Thèse de doctorat

Hachage vers les courbes elliptiques et cryptanalyse de schémas RSA.

présentée et soutenue publiquement le 23 septembre 2011 par

Mehdi Tibouchi

pour obtenir le grade de

Docteur en informatique de l'Université Paris Diderot et de l'Université du Luxembourg

devant le jury composé de

Dr. Jean-Sébastien Coron, directeur de thèse
Assistant Professeur, Université du Luxembourg

Dr. David Naccache, co-directeur de thèse
Professeur, Université Paris II & ENS

Dr. Pierrick Gaudry
Directeur de recherche, CNRS & LORIA

Dr. Jean-François Mestre, président suppléant
Professeur, Université Paris 7

Dr. Igor Shparlinski
Professeur, Macquarie University

Dr. Jacques Stern, président
Professeur, ENS

Dr. Pierre-Alain Fouque, expert associé
Maître de conférence, ENS

Dr. Tatsuaki Okamoto, expert associé
Directeur de laboratoire, NTT Laboratories

Remerciements

Merci tout d'abord à Pierrick et Igor qui ont accepté la lourde tâche de relire ce manuscrit, et à tous les membres du jury, qui ont bien voulu se libérer à cet horaire inhabituel, et qui pour certains viennent de fort loin !

Merci encore à mes directeurs de thèse, David et Jean-Sébastien, qui m'ont accueilli alors que j'étais un matheux naïf et non sans préjugés, et m'ont fait découvrir combien la cryptographie est un domaine vaste, où l'on trouve des questions passionnantes aussi bien théoriques que très proches des applications. Merci aussi pour nos réunions de recherches itinérantes et toujours intéressantes, et pour leur soutien dans toutes sortes de projets, scientifiques ou non.

Merci également à l'équipe de cryptologie de l'ENS, qui offre aussi bien un environnement scientifique idéal où l'on jouit d'une grande liberté que la possibilité d'aller découvrir le monde. Merci notamment à Damien, Pierre-Alain, Phong, Michel et Vadim, tant pour nos discussions autour d'un tableau noir que pour celles, pas moins importantes, autour d'une pizza à Taormina ou des fraises au chocolat de CRYPTO. Merci à tous les chercheurs de l'équipe, permanents, post-docs, doctorants ou plus jeunes, pour l'ambiance toujours agréable au labo, et aux administratifs pour faire en sorte que tout cela fonctionne (pensée particulière pour Joëlle, Valérie et Michelle, pour avoir supporté mon côté tête-en-l'air).

Merci aussi aux collègues qui m'ont accueilli à Ingenico au début de cette thèse, en particulier Éric, avec qui discuter de mathématiques est toujours un plaisir, et Thomas, avec qui j'espère avoir encore l'occasion de prendre un verre sur Orchard Road ou à Shibuya.

Merci aux collègues de NTT, auprès de qui j'ai passé six excellents mois de stage et que j'aurai plaisir à rejoindre par la suite. Merci à Okamoto-san pour son accueil si aimable, à Uchida-san pour sa prévenance et les discussions en français que nous avions sur le Japon (ou inversement), à Yamamoto-san, Abe-san, Fujioka-san et bien d'autres pour des discussions scientifiques toujours intéressantes, à Nishimaki-san et Sasaki-san pour leur aide à la résidence de Sayamagaoka, et à Berkant et Daegun avec qui j'ai pu partager l'expérience de la vie d'étranger au Japon.

Au cours de cette thèse, j'ai eu l'occasion de collaborer avec de nombreux co-auteurs. Merci à tous : ce travail est le vôtre autant que le mien.

Merci à tous ceux qui ont permis que j'aie de quoi vivre pendant cette thèse : David N., David P. et Damien, qui se sont démenés pour cela, Ingenico, NTT, l'ENS et le

projet ANR PACE, ainsi que le projet européen ECRYPT II et le ministère des Affaires étrangères qui ont pris en charge certains de mes déplacements. Merci aussi à ceux qui m'ont encouragé à entreprendre cette thèse ou m'ont soutenu au travers de débuts administrativement délicats : David et Damien, Philippe Clarisse et les collègues de Saint-Louis, Johan Yebbou, Louis Vogel et Chantal Ladoux, David Madore, Yazid Sabeg et d'autres sans doute qui sont intervenus sans que j'en aie eu connaissance.

Et parce qu'il y a une vie en dehors du labo, merci enfin à ceux qui m'ont supporté au quotidien pendant ces trois ans : mes parents bien sûr (et ça n'a pas dû être facile !) ; Chantal Ladoux ; Viêt-Linh, JB et Jean-Rémy ; yaforum ; et mes compagnons de virées du côté de la rue de la Michodière ou de Higashi-Ikebukuro.

はるかまで旅してゐたり昼寝覚

森　澄雄

Sommaire

Chapitre 1

Introduction

1.1 Introduction à la cryptologie

L'usage de codes secrets, principalement dans les communications diplomatiques et militaires, remonte à l'Antiquité (on en trouve déjà des traces dans l'Égypte du moyen empire, puis en Grèce antique), où il était encore artisanal. Peu à peu, une compétition s'installe entre les utilisateurs de codes secrets et ceux qui tentent de les percer : à partir du Moyen-Âge, plusieurs traités sont ainsi écrits sur des techniques permettant de casser les codes secrets (notamment l'analyse de fréquence à l'encontre des chiffrements par substitution simple). L'étude des messages secrets se constitue progressivement comme une science, la *cryptologie*, rassemblant ces deux aspects apparemment antagonistes : la *cryptographie* d'une part, visant à élaborer des méthodes pour sécuriser les communications, et la *cryptanalyse* d'autre part, visant à déceler des faiblesses dans ces méthodes.

Depuis quelques décennies, l'essor des télécommunications et le développement et la miniaturisation des moyens de calcul ont permis à la cryptographie de dépasser le champ militaire et d'investir notre vie quotidienne : des dispositifs aussi variés qu'un téléphone mobile, une carte bancaire, un passeport biométrique et un navigateur web effectuent des calculs cryptographiques, afin de garantir diverses propriétés de sécurité de leurs communications.

Parmi les propriétés de sécurité auxquelles s'intéresse la cryptologie, on peut citer l'*intégrité* des messages (lorsque l'on veut s'assurer qu'ils n'ont pas été altérés sur le canal de transmission : par exemple quand on télécharge un fichier), leur *authenticité* (lorsque l'on veut s'assurer que l'expéditeur est bien celui qu'il prétend : par exemple qu'une transaction bancaire est bien réalisée par une carte de paiement autorisée) ou encore leur *confidentialité* (lorsque l'on veut les garder secrets en dépit des personnes pouvant espionner la communication : par exemple lorsque l'on consulte son courrier électronique à distance). Le cryptographe propose des algorithmes à exécuter par les différentes parties pour satisfaire ces propriétés de sécurité, et le cryptanalyste y cherche des faiblesses.

Si par le passé il a pu être courant de chercher à dissimuler ou garder secret les

algorithmes cryptographiques afin de compliquer la tâche du cryptanalyste, on reconnaît désormais que ce genre de secret, qu'il faut nécessairement partager avec ses correspondants, est peu robuste, et que l'impression de « sécurité par l'obscurité » est largement illusoire. Selon le principe énoncé par Kerckhoffs en 1883, il convient qu'un système cryptographique utilise des algorithme publics, qui n'utilisent eux-mêmes qu'une petite quantité d'information gardée secrète : la *clef*. Si la clef est compromise, elle peut être remplacée sans avoir à concevoir tout un nouveau système. De plus, cela permet à l'algorithme d'être évalué publiquement : si un algorithme cryptographique public a résisté longtemps aux tentatives d'attaques des cryptanalystes, on a davantage confiance en sa sécurité qu'en celle d'un algorithme secret qu'aucun analyste n'a eu l'occasion d'étudier.

1.2 Cryptographie moderne

1.2.1 Clef symétrique et clef publique

Jusque dans les années 1970, il était entendu que la clef cryptographique était un secret partagé entre l'expéditeur et le destinataire : les systèmes cryptographiques étaient alors toujours *symétriques*, au sens où la même clef servait au deux parties.

Pour assurer la confidentialité, par exemple, on utilisait des systèmes de chiffrement symétriques par blocs ; un tel système est la donné deux algorithmes, E_k et D_k, qui constituent des familles de permutations inverses l'une de l'autre des chaînes de bits d'une certaine longueur fixée (par exemple 64 ou 128 bits), indicées par la clef symétrique k. L'expéditeur, Alice, désirant faire parvenir à Bob, avec qui elle partage une clef secrète k, un certain message m d'une façon qui préserve la confidentialité, envoie alors le *chiffré* (ou *cryptogramme*) $c = E_k(m)$, et Bob, à la réception, retrouve m en déchiffrant c : $m = D_k(c)$.

Pourvu que les algorithmes E_k et D_k aient de bonnes propriétés de sécurité, cette approche fonctionne très bien et est encore d'un usage courant aujourd'hui. Cependant, elle présente certains défauts, notamment du point de vue de la distribution de clef. Pour limiter l'impact de la compromission d'une clef, il est important que, dans une organisation où les individus s'échangent des messages chiffrés, chaque personne utilise une clef différente pour communiquer avec n'importe quelle autre. En cryptographie symétrique, cela impose d'utiliser une clef différente pour chaque paire d'individus : dans une organisation comptant plusieurs milliers de personnes, le nombre de clef à maintenir se compte en millions, et devient donc rapidement ingérable.

D'autre part, il y a des primitives de sécurité intéressantes qui ne sont pas réalisables en cryptographie symétrique. Par exemple, deux personnes partageant une clef commune k peuvent s'assurer mutuellement de l'authenticité des messages qu'elle s'échangent (on peut pour cela utiliser la primitive cryptographique appelée *code d'authentification de message*, ou MAC). En revanche, elles ne peuvent pas convaincre un tiers de cette authenticité, puisque lui-même ne dispose pas du secret k. On ne peut donc pas espérer apposer une signature publiquement vérifiable sur un message en utilisant de la cryptographie symétrique.

Ces limites ont pu être dépassées avec l'avènement de la *cryptographie à clef publique*, proposée par Diffie et Hellman en 1976 [Hel76], et dont la première réalisation, peu de temps après, est due à Rivest, Shamir et Adleman [RSA78].

En cryptographie à clef publique, l'expéditeur et le destinataire n'ont pas besoin de partager de secret. Si Alice souhaite envoyer un message m à Bob de manière confidentielle, elle lui demande sa *clef publique* pk, une information qui n'a pas besoin d'être gardée secrète, et applique un algorithme de chiffrement utilisant cette clef publique, pour obtenir un chiffré $c = \mathsf{Encrypt}(pk, m)$. La clef publique, en revanche, ne permet pas de retrouver m à partir de c. Il faut pour cela disposer de la *clef privée* sk de Bob, que celui-ci garde secrète ; il est donc le seul à pouvoir retrouver m à partir de c, sous la forme $m = \mathsf{Decrypt}(sk, m)$.

La cryptographie à clef publique simplifie ainsi grandement le problème de la distribution de clefs : dans une grande organisation, il suffit d'une paire de clef (sk, pk) par individu, et non plus d'une clef secrète par paire d'individus désirant communiquer : plus besoin de millions de clefs pour quelques milliers de personnes !

De plus, comme l'avaient observé Diffie et Hellman, la cryptographie à clef publique permet l'existence de signatures numériques. Dans ce contexte, Alice possède une clef de signature secrète sk et une clef de vérification publique pk. La clef de signature permet de construire une signature $\sigma = \mathsf{Sign}(sk, m)$ publiquement vérifiable à l'aide de pk : $\mathsf{Verify}(pk, \sigma, m) = \texttt{true}$ si et seulement si σ est de la forme précédente. En revanche, la connaissance de pk ne permet pas de retrouver sk, ni de produire des signatures. On a donc un système dans lequel une signature est « opposable aux tiers » ; depuis une dizaine d'années, elles ont d'ailleurs acquis la même valeur juridique en droit français que les signatures papier (Code civil, art. 1316–4).

Malgré ces nombreux avantages, la cryptographie à clef publique possède un inconvénient de taille : celui d'être, en pratique, sensiblement moins efficace que la cryptographie symétrique. C'est pourquoi de nos jours on utilise souvent l'une et l'autre en conjonction : pour chiffrer un long message, par exemple, on va souvent chiffrer une clef symétrique aléatoire (*clef de session*) avec une clef publique, et chiffrer le long message lui-même avec la clef de session, en utilisant un algorithme de chiffrement symétrique (c'est ce que l'on appelle le chiffrement hybride).

Les travaux présentés dans ce manuscrit relèvent tous de la cryptologie à clef publique.

1.2.2 Problèmes difficiles

Un algorithme de chiffrement à clef publique est une *fonction à sens unique* : la fonction qui à un message m associe le chiffré $c = \mathsf{Encrypt}(pk, m)$ doit être efficacement calculable pour que le chiffrement soit utilisable, mais elle doit également être difficile à inverser, puisqu'il ne doit pas être possible de retrouver publiquement m à partir de c. Or on ignore s'il existe réellement des fonctions à sens unique : leur existence implique $\mathsf{P} \neq \mathsf{NP}$ (en tout cas si l'on interprète, comme on le fait habituellement, « efficace » comme signifiant « polynomial »), ce qui est peut-être le plus célèbre problème ouvert en informatique théorique.

Par conséquent, toute la cryptographie à clef publique est en fait heuristique ou

conjecturale. Pour construire des primitives cryptographiques à clef publique, on doit en général *supposer* que certains problèmes NP sont difficiles (non résolubles en temps polynomial), sans que l'on sache pour le moment établir un tel résultat. Citons quelques un des problèmes supposés difficiles les plus couramment utilisés dans la conception de systèmes cryptographiques à clef publique.

Factorisation et RSA. Le premier algorithme de chiffrement à clef publique, proposé par Rivest, Shamir et Adleman [RSA78], avait la forme suivante (que l'on appelle aujourd'hui « textbook RSA »).

La génération de clef tire au hasard deux grands nombres premiers p et q, et les multiplie pour obtenir le *module public* $N = pq$. Il est par ailleurs choisi (généralement pas au hasard) un entier e premier à $\varphi(N) = (p-1)(q-1)$ (et en particulier impair), appelé *exposant public*. La clef publique pk est le couple (N, e). La clef privée sk est un entier d inverse de e modulo $\varphi(N)$.

Le chiffrement d'un message $m \in \mathbb{Z}_N^*$ s'obtient alors simplement en élevant à la puissance e : $c = m^e \bmod N$; et le déchiffrement est l'élévation à la puissance d : $m = c^d \bmod N$ en vertu du théorème d'Euler sur l'ordre multiplicatif des entiers modulo N.

Retrouver la clef privée d en connaissant la clef publique (N, e) est essentiellement équivalent à factoriser N. En effet, si l'on connait p et q, on peut calculer $\varphi(N)$ et donc l'inverse de e modulo $\varphi(N)$. Réciproquement, connaissant d et e, on peut arriver à retrouver $\varphi(N)$, ce qui fournit ensuite la somme $p + q = N - \varphi(N) + 1$ et le produit $pq = N$ de p et q, et donc révèle la factorisation.

Or jusqu'à présent, on ne connaît pas d'algorithme efficace (disons polynomial) pour factoriser le produit de deux grands nombres premiers : c'est l'un des problèmes supposés difficiles les plus importants en cryptographie. Le meilleur algorithme connu actuellement pour la factorisation de tels entiers date des années 1990 et est nettement sous-exponentiel : il s'agit du crible algébrique [LJMP90].

Notons que le caractère « à sens unique » du textbook RSA, c'est-à-dire l'impossibilité de retrouver m à partir de c (i.e. de calculer des racines e-ièmes modulo N) est a priori un problème plus facile que la factorisation, que l'on appelle *problème RSA* : si l'on sait factoriser N, on peut calculer des racines e-ièmes, mais on ne voit pas comment factoriser N à l'aide d'un calcul de racines e-ièmes. Néanmoins, on ne connaît pas non plus d'algorithme meilleur que la factorisation pour calculer des racines e-ièmes : l'équivalence ou non du problème RSA avec la factorisation est un problème ouvert.

Le chiffrement textbook RSA n'est pas considéré comme sûr (le fait qu'il soit déterministe, par exemple, l'empêche de satisfaire la propriété de sécurité importante appelée indistinguabilité des chiffrés), mais l'on sait effectivement construire des systèmes de chiffrement, de signature, et beaucoup d'autres primitives cryptographiques qui sont sûrs si l'on suppose le problème RSA difficile, et ces primitives restent les plus utilisées aujourd'hui dans les applications.

On peut également effectuer la plupart de ces constructions en supposant seulement la factorisation difficile, en se fondant sur le système de Rabin (qui consiste essentiellement à choisir $e = 2$ dans ce qui précède), au prix d'une certaine perte de simplicité ou

d'efficacité.

Toute la seconde partie de cette thèse est consacrée à l'analyse de systèmes cryptographiques (principalement des schémas de chiffrement ou de signature) dérivés de RSA ou de certaines de ses variantes.

Logarithme discret et problèmes Diffie-Hellman. Dans leur article marquant le départ de la cryptographie à clef publique [Hel76], Diffie et Hellman n'ont pas construit de système de chiffrement, mais ils ont proposé une construction pour une primitive différente : l'échange de clef. Il s'agit pour Alice et Bob, communiquant sur un canal susceptible d'être écouté, de dériver un secret commun (par exemple une clef de session) d'une façon qui ne permette pas à un adversaire ayant espionné l'ensemble de la communication de découvrir ce secret.

La méthode proposé par Diffie et Hellman est la suivante. On suppose fixés une fois pour toute un grand nombre premier q, et un générateur g d'une grand sous-groupe $\mathbb{G}$ d'ordre premier p de $\mathbb{Z}_q^*$. Alors Alice tire au hasard $x \in \mathbb{Z}_p$ et envoie à Bob l'élément g^x de $\mathbb{G}$. De même Bob tire au hasard $y \in \mathbb{Z}_p$ et envoie à Alice g^y. Ils peuvent alors tous les deux calculer la valeur commune $g^{xy} = (g^x)^y = (g^y)^x$.

L'adversaire espionnant la communication, Ève, peut alors casser le protocole si elle est capable de calculer g^{xy} à partir de g^x et g^y. Ce problème s'appelle CDH, ou *problème Diffie-Hellman calculatoire*. On ne connaît pas de meilleure attaque que celle consistant à retrouver x à partir de g^x et de même pour y, ce qui constitue le *problème du logarithme discret* (DL). L'algorithme le plus efficace connu pour le calcul du logarithme discret dans $\mathbb{Z}_q^*$ est d'une complexité équivalente à celle de la factorisation : c'est également un algorithme de crible algébrique.

De nombreuses primitives cryptographiques ont été construites sur la base du logarithme discret, du problème CDH, de sa variante décisionnelle DDH (i.e. distinguer le triplet (g^x, g^y, g^{xy}) de (g^x, g^y, g^z)) ou de variantes plus exotiques : outre l'échange de clef, de nombreux schémas de chiffrement, de signature ou d'identification sont basés sur ces problèmes difficiles.

Bien que le groupe $\mathbb{G}$ considéré originellement par Diffie et Hellman est dans $\mathbb{Z}_q^*$, le problème du logarithme discret et ses variantes ont un sens dans un groupe cyclique arbitraire, et la plupart des constructions se transposent immédiatement à un autre groupe sous réserve que ces problèmes y soient encore difficiles. Ce n'est pas toujours le cas (par exemple le logarithme discret est très facile dans un groupe tel que le groupe additif $\mathbb{Z}_p$ lui-même), ou parfois ce n'est pas plus intéressant que $\mathbb{Z}_q^*$ lui-même (par exemple travailler dans un sous-groupe cyclique d'un groupe algébrique linéaire n'apporte généralement rien en terme de sécurité). Mais le cas où $\mathbb{G}$ est un sous-groupe cyclique du groupe des points d'une courbe elliptique sur un corps fini, envisagé dès 1985 par Koblitz [Kob87] et Miller [Mil85], présente un intérêt tout particulier : on appelle *cryptographie par courbes elliptiques* la cryptographie construite sur les problèmes difficiles précédents dans un tel groupe.

La cryptographie par courbes elliptiques a d'abord été considérée avec une relative méfiance, ou en tout cas prudence, par la communauté cryptologique. Des attaques

Niveau de sécurité (bits)	RSA/Corps finis	Courbes elliptiques
80	1248	160
96	1776	192
112	2432	224
128	3248	256
256	15424	512

Table 1.1 : Comparaison des tailles de clefs pour RSA ou DL dans les corps finis d'une part, et DL dans les courbes elliptiques d'autre part, pour divers niveaux de sécurité [S+10a].

comme celle de Menezes-Okamoto-Vanstone [MVO91, MOV93], réduisant le problème du logarithme discret sur certaines courbes elliptiques à celui dans le groupe multiplicatif d'un corps fini, ont suscité une certaine circonspection. Cependant, à mesure que les efforts pour casser le logarithme discret sur les courbes ordinaires se sont révélés infructueux, et que l'arithmétique sur ces courbes s'est faite plus efficace, les courbes elliptiques ont gagné la faveur des cryptographes. Mais c'est sans doute l'avènement de la cryptographie *à base de couplages*, utilisant la structure bilinéaire du groupe des points de certaines courbes, et permettant des constructions cryptographiques riches et totalement nouvelles (comme le chiffrement à base d'identité de Boneh-Franklin [BF01] ou l'échange de clef tripartite de Joux [Jou00]) qui a imposé les courbes elliptiques dans la communauté académique.

Depuis lors, comme le logarithme discret et les problèmes apparentés semblent toujours avoir une complexité exponentielle sur les courbes elliptiques là où elle est sous-exponentielle dans les corps finis (comme la factorisation et RSA), les courbes elliptiques ont suscité un vif intérêt industriel, en étant susceptible d'améliorer notablement les performances des applications tant logicielles que matérielles grâce à des clefs plus courtes (voir Tab. 1.1). L'usage des courbes elliptiques est désormais recommandé par des agences comme la NSA [NSA05].

La première partie de cette thèse est consacrée à diverses contributions à la cryptographie par courbes elliptiques.

Autres problèmes. De nombreux autres problèmes difficiles ont été envisagés pour la construction de systèmes cryptographiques à clef publique, depuis les problèmes de sac à dos (Merkle-Hellman [MH78]) ou des problèmes de décodage pour les codes linéaires (McEliece [McE78]) jusqu'à la résolutions de grands systèmes polynomiaux (cryptographie multivariée). Toutefois, la plupart de ces systèmes restent peu usités en pratique.

Une famille de problèmes qui s'est avérée particulièrement fructueuse pour la construction de primitives cryptographiques ces dernières années concerne les *réseaux* (les sous-groupes discrets de $\mathbb{R}^n$), où il peut s'agir par exemple de trouver un plus court vecteur non nul, ou bien un plus proche vecteur à un point donné de l'espace vectoriel ambiant, éventuellement à un certain facteur d'approximation près. Des constructions basées sur

ces problèmes ont été proposées pour la plupart des primitives dont on connaissait une instanciation à base de couplages (chiffrement à base d'identité...), ainsi que pour de toutes nouvelles primitives, notamment le chiffrement totalement homomorphe. Ces constructions, même si elles sont parfois « asymptotiquement efficaces », utilisent en général des paramètres trop grands pour envisager leur utilisation pratique, en particulier dans les applications embarquées, mais elles revêtent une importance théorique certaine. Par ailleurs, il existe un schéma de chiffrement à base de réseaux, NTRUEncrypt, ayant reçu une normalisation ANSI pour des usages embarqués [ANSI X9.98].

Nous avons consacré quelques travaux à la cryptographie à base de réseaux, non présentés dans ce manuscrit. On en mentionne certains au paragraphe §2.3.1.

1.2.3 Sécurité heuristique et sécurité prouvée

Quand peut-on dire qu'une certaine primitive cryptographique, par exemple un schéma de chiffrement ou de signature, est sûre ? Aux débuts de la cryptographie à clef publique, la réponse à cette question n'était pas claire. Par exemple, s'il est vrai que, sous l'hypothèse que le problème RSA est difficile, il n'est pas possible en général de décrypter[1] un chiffré textbook RSA ; cependant, comme on l'a indiqué plus haut, ce schéma n'est pas considéré comme sûr : il est ainsi possible de décrypter les chiffrés des messages courts, ou encore de déterminer si deux chiffrés correspondent au même message clair.

En cherchant à porter une réponse assez générale à cette question, les cryptographes des années 1980 ont été amenés à formaliser la cryptographie à clef publique en donnant des définitions rigoureuses des propriétés de sécurité que l'on cherche à atteindre pour différentes primitives, et en tentant de proposer des schémas satisfaisant ces propriétés. Ainsi, pour le chiffrement, Goldwasser et Micali [GM82, GM84] ont dégagé la notion de *sécurité sémantique*, ou d'*indistinguabilité des chiffrés* : un schéma de chiffrement satisfait cette propriété lorsqu'un adversaire efficace ne peut obtenir aucun bit d'information sur un message à partir d'un chiffré (ou de façon équivalente, étant donné un chiffré d'un message parmi deux de son choix, il ne peut pas deviner duquel des deux messages c'est le chiffré significativement mieux qu'en répondant au hasard : il est clair dit comme cela qu'un chiffrement sémantiquement sûr ne peut pas être déterministe). Pour les signatures, Goldwasser, Micali et Rivest [GMR84, GMR88] ont défini l'*infalsifiabilité existentielle* : un adversaire ne possédant pas la clef de signature ne peut produire de signature sur aucun nouveau message. Ces notions ont été largement reconnues par la suite comme « les bonnes »,[2] pourvu qu'il soit donné à l'adversaire des *moyens* suffisamment puissant (l'accès adaptatif à un oracle de déchiffrement ou à un oracle de signature selon le

[1] Rappelons que « déchiffrer » un cryptogramme consiste à retrouver le message clair quand on possède la clef privée, tandis que le « décrypter » consiste à faire de même sans la clef. Un algorithme de déchiffrement relève de la cryptographie, tandis qu'un algorithme de décryptement relève de la cryptanalyse.

[2] À tel point que les cryptographes théoriciens considèrent parfois qu'une attaque qui ne met pas en défaut ces propriétés de sécurité « n'est pas une attaque », dans la mesure où elle ne fait pas partie du « cahier des charges » de la primitive correspondante, et ce même si l'attaque en question peut être un problème de sécurité dans un cas courant d'utilisation de la primitive. Une discussion un peu polémique de cette question a récemment été donnée par Koblitz et Menezes [KM11].

cas). Et des études similaires ont été conduites pour toute sorte d'autres primitives cryptographiques.

Une fois définies rigoureusement les notions à atteindre, il a été possible de construire des schémas et de *prouver* qu'ils les atteignaient, sous réserve qu'un certain problème soit difficile. La forme que prennent ces démonstrations est celle d'une *réduction de sécurité* : on suppose qu'il existe un adversaire efficace (polynomial) cassant la propriété voulue du schéma avec probabilité non négligeable, et on montre que cet adversaire peut être utilisé comme « sous-programme » pour résoudre efficacement le problème supposé difficile. Mais comme le problème est supposé difficile, il ne peut être résolu efficacement, et l'adversaire ne peut donc pas exister ! De cette manière, les articles cités précédemment exhibent des schémas dont ils prouvent qu'ils satisfont les propriétés de sécurité voulues.

Cependant, la plupart des schémas « prouvés sûrs » proposés jusque dans les années 1990 (et un bon nombre de ceux proposés depuis lors également) étaient d'une efficacité limitée, ou en tout cas insuffisante pour les applications. L'industrie et le grand public se contentaient donc souvent de schémas *ad hoc*, c'est-à-dire conçus pour résister à un certain nombre d'attaques connues, mais dont il n'était pas pour autant établi qu'il satisfaisait les propriétés de sécurité souhaitables. Par exemple, pour construire des schémas de chiffrement ou de signature RSA palliant les attaques simples décrites plus haut et quelques autres de ce type, il a été proposé diverses constructions ad hoc appliquant la fonction RSA non pas au message lui-même, mais à son image par *fonction de remplissage* ou *padding*, pouvant par exemple consister à faire précéder le message d'un certain nombre de bits d'aléa. Ces techniques sont beaucoup plus efficaces que celles utilisés dans les schémas prouvés, mais il n'est pas démontré qu'elles sont sûres, et on y a souvent découvert des faiblesses (c'est d'ailleurs l'un des buts de cette thèse).

Cette division entre une cryptographie théorique prouvée sûre mais inefficace et une cryptographie pratique rapide mais à la sécurité incertaine s'est un peu réduite dans les années 1990, notamment grâce à l'introduction par Bellare et Rogaway du *modèle de l'oracle aléatoire* [BR93], dans lequel les fonctions de hachage cryptographiques sont modélisées par des fonctionnalités idéales répondant uniformément au hasard à chaque nouvelle requête. Des constructions bien plus efficaces qu'auparavant (et souvent aussi efficaces que les constructions ad hoc) ont pu être prouvées sûres dans ce modèle. Une telle preuve n'est pas aussi « solide » qu'une preuve dans le modèle standard, puisqu'il n'existe pas de véritable oracle aléatoire, et qu'instancier un oracle aléatoire par une fonction de hachage cryptographique réelle ne préserve la sécurité que de façon heuristique. Cependant, un schéma prouvé sur dans ce modèle est certainement bien préférable à un schéma ad hoc, et de fait, on n'a pas constaté d'attaque sérieuse sur ces nouvelles constructions contrairement à celles qui les ont précédées.

Jusque dans les années 1990, les systèmes cryptographiques à clef publique qui ont été consacrés par une agence de normalisation, ce qui est en général un préalable nécessaire à une large adoption industrielle, étaient pour ainsi dire tous des schémas ad hoc. S'agissant de RSA, on peut citer par exemple les paddings ad hoc [PKCS#1 v1.5], [ISO9796–1] et [ISO9796–2]. Depuis les années 2000, des schémas prouvés dans le modèle de l'oracle aléatoire ont également été normalisés (c'est le cas des paddings RSA comme

OAEP [BR94, ISO18033–2] pour le chiffrement et PSS [BR96, PKCS#1 v2.1] pour les signatures), et on peut espérer que cela ouvre la voie à leur déploiement dans les applications. Malgré tout, aujourd'hui encore, les schémas ad hoc demeurent les plus utilisés, du fait des cycles de renouvellement assez lents de beaucoup d'applications embarquées (comme les terminaux de paiement) et d'une certaine réticence des industriels à faire évoluer leur code cryptographique (ce qui est coûteux) en l'absence d'attaque directe sérieuse sur leur produit (et parfois même dans ce cas !).

Chapitre 2

Présentation des travaux

Les travaux présentés dans ce manuscrit appliquent diverses techniques d'algèbre et d'arithmétiques aux deux versants de la cryptologie, construction et analyse, dans deux directions distinctes qui correspondent également aux deux familles de cryptosystèmes à clef publique sans doute les plus utilisés de nos jours : ceux fondés sur RSA d'un côté et sur le logarithme discret dans les courbes elliptiques (ou les jacobiennes de courbes algébriques) de l'autre.

Les travaux de construction, d'une part, concernent la cryptographie à base de courbes algébriques, et plus particulièrement, pour l'essentiel, le problème précis de l'encodage et du hachage à valeurs dans les courbes elliptiques ou les jacobiennes de courbes hyperelliptiques. Nous avons notamment étudié certaines propriétés quantitatives de diverses fonctions d'encodage vers les courbes, notamment la taille de leur image ou leur probabilité de collision, et obtenu une construction satisfaisante (c'est-à-dire indifférentiable d'un oracle aléatoire) de fonctions de hachage à partir de ces encodages. Ces résultats utilisent principalement des méthodes d'arithmétique des corps de fonctions, de géométrie des courbes et des surfaces, et de sommes de caractères. Toujours en cryptographie à base de courbes elliptiques, nous avons également travaillé sur une question d'implémentation : l'obtention de formules d'addition et de doublement sûres et efficaces.

Les travaux de cryptanalyse, d'autre part, ont porté sur divers cryptosystèmes fondés sur RSA, principalement des schémas de chiffrement et de signature. Nous avons en particulier obtenu et mis en œuvre une attaque pratique à messages choisis sur la norme ISO/IEC 9796-2 :2002 de signatures RSA, utilisée notamment par les cartes et terminaux de paiement EMV. Sur cette même norme, nous avons également étudié et implémenté une attaque physique par fautes. Plus généralement, nous avons travaillé sur un nouveau type d'attaque par fautes sur les implémentations de schémas de signatures RSA utilisant les restes chinois. Par ailleurs, nous avons obtenu de nouveaux résultats sur la sécurité de la norme PKCS#1 v1.5 de chiffrement RSA, et une meilleure attaque sur les schémas RSA utilisant de petits sous-groupes. Les outils employés vont des techniques de calcul d'indice ou de réduction de réseaux à l'algorithmique efficace des polynômes de grand degré.

Le présent manuscrit rassemble ces différents travaux en deux parties selon cette double distinction : cryptographie par courbes elliptiques, puis cryptanalyse de schémas RSA. À ces deux familles de travaux s'ajoutent quelques résultats non présentés dans ce manuscrit mais mentionnés à la fin de ce chapitre, et portant sur des questions telles que le chiffrement totalement homomorphe.

2.1 Contributions à la cryptographie par courbes elliptiques

La première partie de ce manuscrit rassemble les travaux sur la cryptographie par courbes elliptiques et hyperelliptiques, notamment sur l'encodage et le hachage vers les courbes.

2.1.1 Hachage en temps constant vers les courbes (hyper)elliptiques

Le problème principal considéré dans cette partie, et présenté plus longuement au chapitre 3, est celui du hachage en temps constant vers les courbes elliptiques et hyperelliptiques. Comme mentionné plus haut, un certain nombre de protocoles cryptographiques utilisant des courbes elliptiques, notamment basés sur les couplages, nécessitent de pouvoir hacher dans le groupe $\mathbb{G}$ des points de la courbe : ils mettent en jeu une ou plusieurs fonctions de hachage $\mathfrak{H} : \{0,1\}^* \to \mathbb{G}$ envoyant des chaînes de bits quelconques vers la courbe. Il en va ainsi du schéma de chiffrement basé sur l'identité de Boneh-Franklin [BF01], des signatures courtes de Boneh-Lynn-Schacham [BLS01] et beaucoup d'autres schémas de chiffrement, de signatures ou d'identification prouvés sûrs dans le modèle de l'oracle aléatoire pour ces fonctions $\mathfrak{H}$.

Cependant, s'il est souvent raisonnable d'instancier un oracle aléatoire à valeurs dans les chaînes de bits de longueur fixée par une fonction de hachage cryptographique, et que l'on peut aisément en déduire des oracles aléatoires à valeurs dans des groupes tels que $\mathbb{Z}_p^*$ ou $\mathbb{F}_q^*$, obtenir un oracle aléatoire à valeurs dans le groupe des points d'une courbe elliptique est en revanche plus délicat, et beaucoup de constructions naïves cassent totalement la sécurité des protocoles où elles sont utilisées.

Une construction dont on peut prouver qu'elle fonctionne effectivement est l'algorithme dit *try-and-increment* : partant d'une courbe elliptique E sur $\mathbb{F}_q$ donnée par son équation de Weierstrass

$$E\colon y^2 = x^3 + ax + b$$

on se donne un oracle aléatoire $\mathfrak{h}\colon \mathbb{F}_q \times \{0,1\}$ (ce que l'on peut instancier raisonnablement en pratique) et l'on procède comme suit pour calculer l'image $\mathfrak{H}(m)$ d'un message m. On commence par faire précéder m d'un compteur c de longueur fixe initialisé à zéro, et l'on évalue $(x, b) = \mathfrak{h}(c\|m)$. Alors si x est l'abscisse d'un point dans $\mathbb{G} = E(\mathbb{F}_q)$, on retourne (x, y) où y est l'une des deux ordonnées correspondantes choisie en fonction du bit b ; sinon, on incrémente le compteur c et on recommence. Si la longueur du compteur est choisie suffisamment grande (supérieure à $\log_2 k$ où k est le paramètre de sécurité), la fonction $\mathfrak{H}$ se comporte comme un oracle aléatoire, et peut donc être utilisée dans les protocoles nécessitant de hacher vers les courbes elliptiques.

Toutefois, la méthode try-and-increment a l'inconvénient que le temps de calcul de l'algorithme dépend du message à hacher. Cela peut permettre à un attaquant physique d'acquérir de l'information sur le message par chronométrage : c'est par exemple un problème particulièrement sérieux pour les protocoles d'authentification à base de mot de passe implémentés dans des dispositifs comme les passeports électroniques, où une telle attaque peut révéler complètement un mot de passe après tout juste quelques exécutions du protocole.

Pour cette raison, et parce que c'est sans doute plus efficace, on cherche à obtenir une construction de fonctions de hachages vers les courbes elliptiques qui s'exécute naturellement en temps constant. Une première étape dans cette direction consiste à construire des « encodages » en temps constant, c'est-à-dire des fonctions $\mathbb{F}_q \to E(\mathbb{F}_q)$ d'image grande et pouvant s'évaluer en temps constant (tous les cas connus de tels fonctions sont en fait des fonctions algébriques, ou au moins « algébriques par morceaux », de petit degré). Pour certaines courbes particulières, notamment supersingulières, il est assez facile d'exhiber de tels encodages, mais traiter le cas général est un problème qui, après avoir essentiellement été formulé par Schoof en 1985 [Sch85], est resté ouvert pendant une vingtaine d'années, jusqu'aux travaux de Skałba [Ska05] et surtout Shallue et van de Woestijne [SvdW06]. À la suite de ces articles, une succession assez rapide de résultats ([Ula07, Ica09, SH09, KLR10] et d'autres encore) a fourni une solution raisonnablement efficace au problème de l'encodage vers toutes les courbes elliptiques, et l'a étendu à un certain nombres de courbes de genre supérieur.

Partant d'un encodage $f\colon \mathbb{F}_q \to E(\mathbb{F}_q)$ et d'un oracle aléatoire $\mathfrak{h}\colon \{0,1\}^* \to \mathbb{F}_q$, on obtient alors une fonction de hachage $\mathfrak{H}(m) = f\big(\mathfrak{h}(m)\big)$ dont on peut montrer dans beaucoup de protocoles particuliers qu'elle peut remplacer un oracle aléatoire à valeurs dans $\mathbb{G} = E(\mathbb{F}_q)$ en préservant la sécurité. Néanmoins, on peut trouver des protocoles pour lesquels ce n'est pas le cas, et il est de fait assez facile pour la plupart des encodages f de distinguer efficacement la fonction $\mathfrak{H}$ d'un véritable oracle aléatoire, ce qui rend difficile l'obtention d'un résultat général de composition.

Plusieurs résultats de cette thèse, mentionnés dans les quatre sections qui suivent, ont pour objet d'éclairer ces questions et de combler certaines lacunes de ce programme.

2.1.2 Estimation de la taille de l'image des encodages en temps constant [FT10b]

Un premier résultat sur l'étude des encodages en temps constant, présenté au chapitre 4 de ce manuscrit, porte sur le calcul du nombre de points dans leur image. Ce nombre de point permet d'une part de constater qu'il existe des distingueurs simples entre la construction $\mathfrak{H}(m) = f\big(\mathfrak{h}(m)\big)$ visée plus haut et un oracle aléatoire, et joue par ailleurs un rôle dans la qualité de la réduction de sécurité dans des constructions plus élaborées.

L'une des fonctions d'encodage en temps constant les mieux connues est par exemple celle d'Icart [Ica09]. Dans cet article, Icart montre que cette fonction f, lorsqu'elle est définie vers une courbe elliptique E sur $\mathbb{F}_q$, a une image de taille au moins $\#f(\mathbb{F}_q) \geqslant q/4$. Cependant, cette borne inférieure est assez grossière, et il donne un argument heuristique

lui suggérant de conjecturer l'estimation bien plus précise :

$$\left|\#f(\mathbb{F}_q) - \frac{5q}{8}\right| \leqslant \lambda\sqrt{q}$$

pour une certaine constante λ universelle. On donne dans cette thèse une preuve de ce résultat, qui repose sur l'interprétation de la fonction d'Icart comme une correspondance algébrique entre E et la droite ainsi que sur le théorème de Chebotarev.

On montre également comment la méthode se généralise de manière naturelle aux autres encodages (y compris ceux dérivés des travaux de Shallue, van de Woestijne et Ulas, dont la structure géométrique est un peu plus compliquée puisqu'elle fait intervenir deux correspondances algébriques au lieu d'une), et ramène le calcul de la taille de l'image au calcul du groupe de Galois du corps de fonctions associé à l'encodage considéré (le corps de fonctions de la courbe donnant la correspondance) vu comme extension du corps de fonctions de la courbe elliptique. On effectue en particulier ce calcul pour la variante de la fonction d'Icart en caractéristique 2, et pour l'encodage « SWU simplifié » introduit en §3.5.4.

2.1.3 Hachage indifférentiable vers les courbes elliptiques [BCI+10a]

Comme mentionné plus haut, la construction $\mathfrak{H}(m) = f(\mathfrak{h}(m))$, où f est un encodage en temps constant vers le groupe $\mathbb{G}$ des points d'une courbe elliptique, ne peut pas *génériquement* remplacer un oracle aléatoire à valeurs dans $\mathbb{G}$ en préservant la sécurité dans le modèle de l'oracle aléatoire. Bien que cette construction soit suffisante dans un certain nombre de protocoles, on peut en trouver d'autres où la sécurité est perdue lorsque l'on utilise cette construction (nous donnons un tel exemple en §5.4.2).

Aussi avons-nous étudié en détails les conditions sur une fonction F garantissant que la construction $\mathfrak{H}(m) = F(\mathfrak{h}(m))$ soit *indifférentiable* d'un oracle aléatoire au sens de Maurer [MRH04], ce qui fournit aussitôt un résultat générique de composition (dans les limites précisées par Ristenpart et al. [RSS11]). Nous avons dégagé la notion d'encodage *admissible* comme condition suffisante et pratique à établir pour obtenir l'indifférentiabilité : un encodage $F\colon S \to \mathbb{G}$ est admissible lorsqu'il est calculable en temps polynomial déterministe, efficacement échantillonnable (c'est-à-dire qu'il existe un algorithme efficace donnant pour tout $\mathbf{P} \in \mathbb{G}$ un antécédent par F presque uniformément distribuée dans $F^{-1}(\mathbf{P})$) et régulier (c'est-à-dire que si s est uniforme dans S, $F(s)$ est presque uniformément distribué dans $\mathbb{G}$).

Les encodages en temps constant $f\colon \mathbb{F}_q \to \mathbb{G}$ vers les courbes elliptiques ne sont pas eux-mêmes des encodages admissibles (sauf quelques exemples très particuliers, comme l'encodage de Boneh-Franklin vers une certaine courbe supersingulière), mais nous donnons deux méthodes permettant de construire un encodage admissible à partir d'un tel f.

Une première méthode consiste à considérer $F\colon \mathbb{F}_q \times \mathbb{Z}/N\mathbb{Z} \to \mathbb{G}$ avec $N = \#\mathbb{G}$ donnée par $F(u, v) = \mathbf{P} + [v] \cdot \mathbf{G}$ où $\mathbf{G}$ est un générateur de $\mathbb{G}$ (supposé cyclique). L'admissibilité de F est assez aisée à établir pour tous les encodages f connus, mais la fonction de

hachage qui en résulte :

$$\mathfrak{H}(m) = f(\mathfrak{h}_1(m)) + [\mathfrak{h}_2(m)] \cdot \mathbf{G}$$

(où $\mathfrak{h}_1, \mathfrak{h}_2$ sont des oracles aléatoires indépendants vers $\mathbb{F}_q$ et $\mathbb{Z}/N\mathbb{Z}$) est assez peu efficace, du fait de la multiplication de longueur pleine dans le groupe $\mathbb{G}$, qui est en général au moins un ordre de grandeur plus lente que l'évaluation de f.

Une seconde méthode plus efficace consiste à prendre $F\colon \mathbb{F}_q \times \mathbb{F}_q \to \mathbb{G}$ donnée par $F(u, v) = f(u) + f(v)$. La fonction de hachage qui en résulte est rapide, mais son indifférentiabilité (l'admissibilité de F) est plus délicate à établir. La difficulté principale consiste à montrer que la fonction F est régulière. Dans un premier premier temps, nous n'avons pu obtenir cette régularité que pour la fonction d'Icart et sa variante en caractéristique 2. L'idée de base, assez simple, consiste à dire que l'ensemble des antécédents (u, v) d'un point $\mathbf{P}$ forme une courbe dans le plan, en général irréductible et de genre borné, et donc de cardinal à peu près constant en dehors de quelques points exceptionnels d'après la borne de Hasse-Weil ; cependant, faire effectivement fonctionner cette idée de preuve nécessite de résoudre un certain nombre d'obstacles techniques assez délicat qui rendent l'extension à d'autres fonctions que celle d'Icart malaisée.

En tout état de cause, nous avons obtenu à ce stade une construction vraiment satisfaisante (sûre et efficace) de fonctions de hachage vers une large classe de courbes elliptiques :

$$\mathfrak{H}(m) = f(\mathfrak{h}_1(m)) + f(\mathfrak{h}_2(m))$$

avec f l'encodage d'Icart et $\mathfrak{h}_1, \mathfrak{h}_2$ des oracles aléatoires indépendants à valeurs dans $\mathbb{F}_q$. Notre approche démontre que cette construction peut remplacer un oracle aléatoire à valeurs dans $\mathbb{G}$ dans essentiellement tout protocole le nécessitant.

2.1.4 Encodages bien distribués [FFS+11]

Nous avons ensuite cherché à étendre cette construction satisfaisante à d'autres fonctions f que celle d'Icart, et dans la mesure du possible à des courbes de genre supérieur.

Comme la méthode de preuve géométrique considérée précédemment s'avère impraticable, nous avons introduit une nouvelle approche de nature arithmétique, décrite au chapitre 6, permettant d'étudier les constructions de fonctions de hachage de la forme plus générale :

$$\mathfrak{H}(m) = f(\mathfrak{h}_1(m)) + \cdots + f(\mathfrak{h}_s(m))$$

où f est l'un quelconque des encodages en temps constant proposés jusqu'ici, à valeurs dans les courbes elliptiques et même hyperelliptiques (dans ce dernier cas, on plonge la courbe dans sa jacobienne et l'addition est celle de la jacobienne). Nous sommes en mesure d'établir que cette construction se comporte convenablement (i.e. est indifférentiable d'un oracle aléatoire quand les $\mathfrak{h}_i$ sont des oracles aléatoires à valeurs dans $\mathbb{F}_q$) dès que s est supérieur au genre de la courbe d'arrivée (autrement dit, $s \geqslant 2$ pour une courbe elliptique, $s \geqslant 3$ pour une courbe de genre 2, etc.). Nous retrouvons notamment les résultats précédents sur la fonction d'Icart comme cas particulier et avec de meilleures

constantes dans les inégalités, et ils s'étendent à tous les encodage en temps constant connus, y compris vers les courbes de genre supérieur.

Afin d'établir ces résultats, nous introduisons la notion d'encodage *bien distribué*, définie en termes d'un nouveau type de somme de caractères sur le groupe des points de la jacobienne de la courbe d'arrivée. Si f est un encodage bien distribué, on obtient de manière formelle des bornes de régularité sur les fonctions de la forme $(u_1, \ldots, u_s) \mapsto f(u_1) + \cdots + f(u_s)$. Il suffit alors de montrer que les encodages en temps constant que l'on considère sont bien distribués, ce qui se ramène, à l'aide d'estimations classiques dues à Weil et Bombieri [Wei95, Bom66], à une propriété géométrique relativement simple (bien plus que celle considérée dans l'approche antérieure, car elle ne met en jeu que la courbe associée à l'encodage, et non des variétés de dimension plus grande).

Au final, nous obtenons des fonctions de hachages efficaces vers toutes les courbes elliptiques, et vers les jacobiennes de toutes les courbes hyperelliptiques pour lesquelles une fonction d'encodage est connue, avec en outre des bornes fines sur la qualité des réductions de sécurité.

2.1.5 Hachage et encodage vers les courbes hyperelliptiques impaires [FT10a]

Un autre champ d'investigation dans le domaine du hachage vers les courbes est celui de la construction des encodages en temps constant eux-mêmes. Nous y avons contribué en passant dans le travail décrit au chapitre 5, en proposant plusieurs encodages efficaces vers les courbes elliptiques sur les corps de caractéristique 3, et de façon plus substantielle dans celui présenté au chapitre 7.

Ce dernier travail porte sur la construction d'une nouvelle famille de fonctions d'encodages particulièrement simple et efficace vers une large classe de courbes hyperelliptiques de genres arbitrairement grand, à savoir les courbes de la forme :

$$H : y^2 = g(x)$$

où g est un polynôme impair. Nous appelons ces courbes H *courbes hyperelliptiques impaires*. On en trouve de fréquents exemples dans la littérature s'agissant du calcul efficace de leur fonction zêta ou de calculs efficaces de couplages, ce qui en fait de bons candidats à diverses applications cryptographiques.

Notre nouvel encodage est presque une bijection de $\mathbb{F}_q$ dans $H(\mathbb{F}_q)$, ce qui rend très simple l'obtention du hachage indifférentiable dans la jacobienne, en utilisant la propriété de bonne distribution évoquée précédemment. Cela permet en outre de construire une fonction d'encodage *injective* d'image grande dans la jacobienne, ce qui est utile pour des applications telles que le chiffrement El Gamal.

L'encodage a également l'avantage de se calculer très efficacement, sans division ni branchement, avec une seule exponentiation et quelques multiplications dans $\mathbb{F}_q$. Cela en fait sans doute l'encodage le plus rapide existant. De plus, il étend de manière significative la famille des courbes hyperelliptiques pour lesquelles un encodage est connu, qui se limitait jusqu'alors à un certain nombre de courbes de genre 2 et seulement quelques courbes isolées en genre plus grand.

2.1.6 Le modèle de Huff [JTV10]

Outre les questions de constructions de fonctions de hachage évoquées jusqu'ici, et qui relèvent de la sécurité des protocoles, nous avons examiné au cours de cette thèse un problème de cryptographie par courbes elliptiques relevant plus directement de préoccupations d'implémentation : celui de l'obtention de formules d'addition et de doublement rapides.

Les courbes elliptiques ont l'avantage sur les groupes multiplicatifs des corps finis ou les groupes RSA d'avoir des tailles de paramètres plus petites à niveau de sécurité égal. Cependant, l'arithmétique y est plus complexe : ainsi, pour une courbe en forme de Weierstrass dont les points sont représentés en coordonnées de Jacobi, très utilisées en pratique, une addition sur la courbe nécessite 16 multiplications dans le corps de base, ce qui relativise le gain d'efficacité obtenu. De ce fait, de nombreuses « formes » de courbes elliptiques ont été examinées dans le but de réduire ce nombre de multiplications et d'obtenir une arithmétique plus efficace pour les applications cryptographiques. On trouve ainsi dans la littérature des formules d'addition efficaces sur les courbes elliptiques en forme d'intersection de Jacobi, de courbes de Hesse, de de Montgomery, de Doche-Icart-Kohel, d'Edwards, etc.

Dans cette thèse, nous avons examiné un autre modèle introduit en 1948 par le mathématicien Gerald Huff [Huf48], à l'origine pour l'étude d'un problème diophantien. Nous avons pu déterminer l'ensemble des courbes elliptiques pouvant se mettre sous cette forme (il s'agit des courbes ayant un sous-groupe isomorphe à $\mathbb{Z}/4\mathbb{Z} \times \mathbb{Z}/2\mathbb{Z}$ sur le corps de base) et proposer des formules d'addition assez compétitives (en 11 multiplications dans le corps de base, ce qui est proche du record détenu par les courbes d'Edwards), possédant de plus des propriétés d'unification et d'indépendance en les paramètres de la courbe utiles contre les attaques par canaux auxiliaires. En outre, nous avons obtenu des formules relativement efficaces pour le calcul de couplages.

Ces résultats, ainsi que leurs extensions à des modèles analogues un peu plus généraux, sont présentés au chapitre 8.

2.1.7 Problèmes ouverts et perspectives

On peut considérer la question de l'encodage et du hachage vers la plupart des courbes elliptiques comme réglée de façon assez satisfaisante. Elle a d'ailleurs déjà trouvé certaines applications industrielles, notamment dans le domaine des passeports électroniques [CT11, CGIP11, BCI10b]. Il subsiste néanmoins un champ assez vaste de questions à explorer, tant pratiques que théoriques.

Sur le plan pratique, on souhaiterait disposer de fonctions d'encodage aussi efficaces que celle d'Icart ou que SWU simplifié pour toutes les courbes elliptiques, et notamment des courbes de Barreto-Naehrig, très utiles dans les applications mettant en jeu des couplages : le seul encodage connu vers ce type de courbes, celui plus général de Shallue et van de Woestijne, est en effet sensiblement plus exigeant en calculs. De même, il serait souhaitable de disposer d'algorithmes explicites efficaces et sans division, comme il en existe pour la fonction d'Icart et pour notre encodage vers les courbes hyperelliptiques

impaires, pour le calcul de chacun des différents encodages en temps constant—afin de s'assurer qu'ils se calculent effectivement en temps constant ! Comme la résistance aux attaques physiques est l'un des buts de ces encodages, il est également souhaitable que les algorithmes ne contiennent pas de branchement conditionnel.

D'un point de vue beaucoup plus théorique, un problème intéressant serait de formuler et d'établir des résultats d'impossibilité dans un groupe générique : il ne devrait pas être possible de construire des fonctions de hachage comme on le fait dans cette thèse sans utiliser la représentation particulière du groupe d'arrivée. On peut par exemple raisonnablement conjecturer qu'il n'existe pas d'encodage admissible de $\{0,1\}^{\mathrm{poly}(N)}$ dans un groupe générique $\mathbb{G}$ de cardinal N. Un tel résultat pourrait généraliser le résultat de Shoup sur la difficulté générique du logarithme discret [Sho97].

À mi-chemin entre le très pratique et le très théorique, on peut constater que les constructions de fonctions d'encodage dont on dispose semblent assez disparates et très ad hoc. On aimerait pouvoir dégager des principes généraux derrière la profusion de formules (ce qui permettrait peut-être en outre de construire des encodages vers de plus larges classes de courbes hyperelliptiques). Un premier pas dans cette direction a récemment été franchi par Couveignes et Kammerer [CK11], qui ont pu donner une description géométrique élégante et unifiée de trois fonctions d'encodage semblables à celle d'Icart, mais qui ne semblaient pas jusqu'alors naturellement liées. Cependant, cette approche ne semble pas se généraliser davantage, et en particulier elle n'admet pas d'analogue évident en genre supérieur. Une autre piste pourrait passer par le constat, dû à Ritzenthaler, que la correspondance algébrique associée à une fonction d'encodage (au moins de « type Icart ») est, d'après un résultat de Guralnick et al., un revêtement exceptionnel de la droite projective, et que ces revêtements admettent une classification combinatoire plus ou moins explicite.

Enfin, une autre voie à explorer sur le plan tant théorique que pratique est celui de la construction d'encodages injectifs. Nous en avons donné un exemple à valeurs dans les jacobiennes de courbes hyperelliptiques impaires, mais le problème de construire de tels encodages reste grand ouvert même pour la plupart des classes d'isomorphismes de courbes elliptiques. Le résultat le plus fort à ce jour dans cette direction est dû à Farashahi [Far11], qui obtient une fonction d'encodage injective vers toutes les courbes de Hesse (i.e. possédant un point d'ordre 3) sur les corps $\mathbb{F}_q$ avec $q \equiv 2 \pmod 3$, qui atteint environ la moitié des points. Il serait intéressant d'aller plus loin.

S'agissant par ailleurs des courbes de Huff, on peut noter que leur généralisation en caractéristique 2, présentée dans cette thèse, admet des formules d'addition et de doublement particulièrement efficaces, exhibées par Devigne et Joye [DJ11], qui en font le modèle binaire le plus rapide actuellement. On peut certainement espérer améliorer également les formules que nous donnons pour le calcul de couplages.

Sur une note plus mathématiquement curieuse, le modèle de Huff a ceci d'assez remarquable qu'il manifeste le fait que la courbe modulaire $X_1(4,2)$, qui paramétrise les courbes elliptiques contenant 8 points rationnels formant un sous-groupe isomorphe à $\mathbb{Z}/4\mathbb{Z} \times \mathbb{Z}/2\mathbb{Z}$, est de genre 0. En effet, il existe un isomorphisme évident entre $\mathbb{P}^1$ et la famille des courbes de Huff, et comme ces courbes ont bien les 8 points rationnels requis,

on en déduit un morphisme non trivial $\mathbb{P}^1 \to X_1(4,2)$. Il serait intéressant d'obtenir un modèle similaire (et si possible d'un niveau comparable de simplicité) associé à toutes les courbes modulaires $X_1(m,n)$ de genre 0. Le problème est, semble-t-il, partiellement ouvert, et pourrait avoir des applications pratiques à des questions comme l'optimisation pratique de l'algorithme de factorisation ECM.

2.2 Cryptanalyse de schémas fondés sur RSA

La seconde partie de ce manuscrit rassemble les travaux de cryptanalyse à l'encontre de divers schémas de signature et de chiffrement RSA, dont plusieurs paddings normalisés. Ces attaques sont pour certaines « en boîte noire » et pour d'autres des attaques physiques ou supposant un accès à un oracle spécifique.

2.2.1 Cryptanalyse pratique des signatures ISO/IEC 9796-2 et EMV [CNTW09]

Le premier de ces travaux, présenté au chapitre 9, concerne le padding ISO/IEC 9796-2 pour les signatures RSA avec récupération partielle de messages, utilisé notamment dans les cartes et terminaux de paiement EMV. Il se fonde sur une attaque antérieure en falsification existentielle par messages choisis, due à Coron, Naccache et Stern [CNS99], sur une version précédente de la norme [ISO9796–2], et qui avait conduit à sa révision.

Nous avons proposé un certain nombre d'améliorations algorithmiques importantes à la technique de Coron-Naccache-Stern, qui nous ont permis d'élaborer et de mener à bien une attaque pratique sur la version alors valide de la norme [ISO9796–2 :2002].

Concrètement, notre attaque, comme celle de Coron-Naccache-Stern, est basée sur l'observation suivante de Desmedt et Odlyzko [DO85]. Considérons un padding RSA μ encodant des messages m en des entiers $\mu(m)$ relativement courts et d'apparence aléatoire (par exemple $\mu = \text{SHA-1}$), et montrons comment on peut espérer falsifier les signatures RSA utilisant μ :

1. On se fixe une borne B et on pose $\mathfrak{P} = \{p_1, \ldots, p_\ell\}$ l'ensemble des nombres premiers inférieurs à B.

2. En essayant des messages m_i au hasard, on fait en sorte d'en trouver $\tau \geqslant \ell + 1$ tels que $\mu(m_i)$ ait tous ses facteurs premiers dans $\mathfrak{P}$ (c'est-à-dire qu'il soit B-friable).

3. En résolvant le système linéaire dans $\mathbb{Z}/e\mathbb{Z}$ (e étant l'exposant public RSA) donné par les exposants des m_i dans la décomposition en facteur premier sur la base $\mathfrak{P}$, on peut alors exprimer l'un des $\mu(m_j)$ comme combinaison multiplicative des $\mu(m_i)$ pour $i \neq j$: c'est analogue à l'étape d'algèbre linéaire des algorithmes de calcul d'indice ou de crible algébrique.

4. Cela fait, on peut obtenir par requête à un oracle les signatures des messages m_i pour $i \neq j$, et en déduire la signature de m_j.

La complexité de cette attaque dépend principalement de la probabilité pour qu'un nombre de la taille de $\mu(m)$ soit B-friable, et de la complexité du test de friabilité : c'est pourquoi si $\mu(m)$ est trop grand, l'attaque peut être nettement plus lente que la factorisation du module RSA lui-même ! Par exemple, la probabilité qu'un nombre de 1024 bits soit 2^{25}-friable est d'environ 2^{-247}, donc l'attaque est inutile contre le Full Domain Hash avec un module de 1024 bits. En revanche, la probabilité qu'un nombre de 100 bits soit 2^{25}-friable est d'environ 0.5%, de sorte que l'on peut aisément trouver $\ell \approx 2^{25}/\log(2^{25}) \approx 2^{21}$ tels nombres et mener l'attaque si μ se comporte comme un oracle aléatoire de longueur 100 bits !

Le padding μ défini par la norme ISO/IEC 9796-2 est de longueur pleine, donc l'attaque de Desmedt-Odlyzko ne peut pas s'appliquer directement. En revanche, Coron, Naccache et Stern ont observé que pour certains messages m_i bien choisis, les entiers $t_i = (2^8 \cdot \mu(m_i)) \bmod N$ étaient assez courts (de taille $k_h + 16$ bits, où k_h est la taille de sortie d'une fonction de hachage utilisé dans le padding), et il est facile d'adapter la technique précédente pour qu'elle s'applique aux t_i plutôt qu'aux $\mu(m_i)$. Cela suffisait pour traiter le cas $k_h = 128$ prévue par la version d'origine de la norme. En revanche, la version révisée, imposant $k_h \geqslant 160$, était hors de portée d'une attaque pratique.

Notre travail a consisté à améliorer cette approche pour réussir à attaquer la version révisée. Les techniques employées sont les suivantes :

- l'utilisation d'un algorithme « par lots » dû à Bernstein [Ber04a] pour le test de friabilité, beaucoup plus efficace en pratique que l'algorithme naïf ;
- l'utilisation d'une variante « avec grand premier » pour la friabilité, comme dans les implémentations modernes des algorithmes de factorisation ou de logarithme discret ;
- un choix plus judicieux de constante à la place du 2^8 dans la définition des t_i, qui permet de réduire leur taille de $k_h + 16$ à $k_h + 8$ environ ;
- une recherche exhaustive sur les bits de poids faible et forts des valeurs de hachage mis en jeu, qui permet d'équilibrer les vitesses respectives du calcul de hachage et du test de friabilité.

On obtient avec ces différentes méthodes une accélération d'un facteur de plusieurs milliers, qui a permis d'attaquer [ISO9796–2 :2002] en environ 1000 heures CPU (et deux jours réels) sur le « nuage » Amazon EC2, pour un coût total de 800 USD. Nous estimons qu'une attaque sur les signatures formatées selon les spécifications EMV, avec la même technique, aurait un coût de 45 000 USD sur les mêmes machines Amazon.

Cette nouvelle attaque a conduit à la publication d'une nouvelle révision de la norme ISO/IEC 9796-2 fin 2010 [ISO9796–2 :2010].

2.2.2 Attaques par fautes sur les signatures EMV [CNT10]

Nous avons également étudié, dans un travail présenté au chapitre 10, la sécurité de la norme ISO/IEC 9796-2 et des spécifications EMV du point de vue des attaques physiques,

et plus précisément des attaques par fautes. Dans ce modèle, l'adversaire a accès au dispositif physique de signature (tel qu'une carte à puce), et est en mesure de le manipuler, à l'aide par exemple de pics électriques ou de flashes laser, pour provoquer des erreurs dans le calcul d'une signature.

Les attaques par fautes ont été introduites par Boneh, DeMillo et Lipton en 1997 [BDL01]. Ils ont montré notamment que les signatures RSA, en particulier lorsqu'elles sont calculées en utilisant le théorème des restes chinois (ce qui est presque toujours le cas en pratique, a fortiori dans les applications embarquées), sont très souvent vulnérables à ce type d'attaques. Une unique signature fautive produite par un dispositif non protégé peut suffire à révéler la clef secrète ! Depuis lors, les attaques par fautes sur les signatures RSA et le contremesures à ces attaques ont constitué un sujet de recherche actif.

Cependant, l'attaque de Boneh et al. n'est pas directement applicable aux signatures ISO/IEC 9796-2 ou EMV, car une partie de l'information utilisée pour calculer ces signatures n'est pas accessible à l'adversaire (soit parce qu'elle est choisie aléatoirement au moment de la génération de signatures, soit parce qu'elle est retrouvée au moment de la vérification—sous réserve que la signature soit correcte !).

À CHES 2009, Coron, Joux, Kizhvatov, Naccache et Paillier [CJK+09] ont montré comment l'attaque de Boneh et al. pouvait malgré tout se généraliser aux signatures ISO/IEC 9796-2, pourvu que la partie du message encodé inconnue de l'adversaire soit suffisamment courte. L'idée est de retrouver les chaînes de bits inconnues comme des petites racines modulaires d'un polynôme multivarié, à l'aide de variantes multivariées de l'algorithme de Coppersmith [Cop97]. Toutefois, les restrictions de taille sur ces chaînes de bits inconnues sont très fortes, ce qui limite significativement la portée de cette attaque en pratique.

Notre contribution a consisté à proposer une approche très différente pour retrouver ces chaînes de bits inconnues, en utilisant plusieurs signatures fautives simultanément, et en en déduisant les parties inconnues avec techniques de réseaux orthogonaux introduites par Nguyen et Stern [NS97, NS98]. Nous montrons que cette approche permet de traiter aisément et très efficacement des chaînes inconnues d'une longueur pouvant aller jusqu'à la moitié de la taille du module.

Sur un exemple pratique de format de signature EMV dépassant de très loin ce qui pouvait être traité avec l'attaque précédente, nous montrons qu'avec notre nouvelle attaque, une dizaine de signatures fautives suffisent à retrouver la clef secrète en une fraction de seconde.

2.2.3 Attaques par fautes sur le module contre les signatures RSA [BNNT11a, BNNT11b]

Toujours en matière d'attaques par fautes contre les signatures RSA utilisant le théorème des restes chinois, nous avons proposé un nouveau type d'attaque, qui s'écarte de la tradition des attaques dérivées de celle de Boneh et al.. Ce travail est présenté au chapitre 11.

L'attaque par fautes traditionnelle de Boneh et al. consiste à perturber l'exponentiation

modulo l'un des deux facteurs du module RSA lors de la génération de signature, et en réponse à cette attaque, de nombreuses contremesures protègent l'exposant RSA privé. Nous montrons dans ce travail que l'on peut mettre en œuvre une attaque par fautes récupérant la clef privé en perturbant plutôt le *module RSA* public juste avant l'interpolation par les restes chinois. C'est une approche très différente, nécessite des contremesures a priori différentes.

Notre nouvelle attaque est, comme celle décrite précédemment, basée sur les techniques de réseaux orthogonaux de Nguyen et Stern, et nous montrons qu'elle est très efficace en pratique. Selon le modèle de fautes considéré, 5 à 45 signatures fautives suffisent à retrouver la clef privée en quelques secondes au plus. Dans sa version la plus simple, l'attaque nécessite que l'adversaire connaisse le module fautif, mais des versions plus sophistiquées fonctionnent même lorsque ce module fautif est inconnu, pourvu que les fautes respectent une forme raisonnable.

Tant l'attaque dans ses diverses variantes que les modèles de fautes considérés ont fait l'objet d'une validation expérimentale complète, à l'aide d'injections de fautes par laser sur un microcontrôleur 8 bits très commun.

2.2.4 Sur la sécurité du chiffrement PKCS#1 v1.5 [BCN+10]

Outre les signatures RSA, nous nous sommes également intéressés à la sécurité de certains paddings de *chiffrement* RSA. En particulier, dans le travail présenté au chapitre 12, nous avons proposé plusieurs nouvelles attaques sur le chiffrement PKCS#1 v1.5, un schéma dont l'utilisation est aujourd'hui déconseillée mais qui reste l'un des plus répandus en pratique. Nous obtenus deux résultats principaux.

Le premier peut être vu comme un prolongement théorique de l'attaque de Bleichenbacher [Ble98], qui permettait de décrypter n'importe quel chiffré PKCS#1 v1.5 à l'aide d'un nombre important (de l'ordre du million) de requêtes adaptatives à un oracle un peu particulier permettant de déterminer si un entier donné était un chiffré PKCS#1 v1.5 valide. De telles attaques « à oracle » sont parfois considérées comme peu problématiques en pratique, mais Bleichenbacher a montré que beaucoup d'implémentations du protocole SSL à l'époque se comportaient précisément comme un tel oracle, en délivrant un message d'erreur en cas de chiffré invalide, ce qui permettait de retrouver en pratique une clef de session SSL en quelques centaines de milliers ou quelques millions d'interaction avec le serveur. Notre nouvelle attaque utilise le même type d'oracle de validité, mais pour casser l'indistinguabilité des chiffrés plutôt que le caractère à sens unique de PKCS#1 v1.5. C'est en un sens plus faible, mais nous montrons qu'il suffit pour cela d'une unique requête à l'oracle plutôt que d'un million.

L'autre résultat cryptanalytique, assez différent, que nous obtenons, concerne la sécurité *broadcast* du chiffrement PKCS#1 v1.5, c'est-à-dire lorsqu'un même message est chiffré à destination de plusieurs utilisateurs. Ce type de modèle d'attaque avait été considéré par Håstad [Hås88] dans les années 1980 pour des paddings qui sont des fonctions polynômes déterministes du message à chiffrer. Nous étendons cette attaque au cas du padding probabiliste PKCS#1 v1.5. L'idée est de retrouver, connaissant plusieurs chiffrés d'un même message pour des destinataires différents, les aléas utilisés pour chacun

des chiffrés en même temps que le message. Nous utilisons pour cela une généralisation multivariée des techniques de Coppersmith, et particulièrement la méthode de Jochemsz et May [JM06] pour la construction du réseau à réduire. En particulier, nous pouvons au moins en principe retrouver un message clair de 936 bits en connaissant un chiffré PKCS#1 v1.5 pour chacun de 4 destinataires différents utilisant des clefs de 1024 bits.

2.2.5 Cryptanalyse de l'hypothèse RSA dans un sous-groupe [CJM+11]

Le dernier résultat exposé dans ce manuscrit, au chapitre 13, s'intéresse non pas à un schéma de signature ou de chiffrement RSA particulier, mais plutôt à la réfutation d'une hypothèse de sécurité liée à RSA et sur la base de laquelle Groth a proposé en 2005 la construction de diverses primitives cryptographiques [Gro05].

Groth considère des modules RSA particuliers de la forme :

$$N = p \cdot q = (2p'r + 1) \cdot (2q's + 1)$$

où p, q, p' et q' sont premiers, avec p', q' relativement petits. Il existe alors un sous-groupe $\mathbb{G}$ de $\mathbb{Z}_N^*$ d'ordre $p'q'$ dont on publie un générateur g avec le module N. Tout le reste, y compris l'ordre $p'q'$ de g, est gardé secret. L'hypothèse est alors que la fonction RSA est difficile à inverser dans $\mathbb{G}$.

La meilleure attaque considérée par Groth (outre la factorisation de N) considérée sur cette hypothèse a pour complexité $O(p')$, ce qui lui permet de proposer un choix de paramètres dans lequel p' et q' sont de l'ordre de 100 bits.

Nous montrons qu'il existe en réalité une attaque permettant de factoriser N connaissant le couple (N, g) en temps $\widetilde{O}(\sqrt{p'})$, ce qui casse le choix de paramètres précédents avec un nombre d'opération de l'ordre de 2^{50} : cela rend ce choix de paramètres potentiellement peu sûr, et a conduit Groth a réviser notablement à la hausse ses prescriptions.

Cette nouvelle attaque consiste essentiellement à appliquer l'algorithme « pas de bébé, pas de géant » dans le groupe $\mathbb{G}$, en utilisant un calcul de PGCD comme test d'égalité. Nous en fournissons une implémentation asymptotiquement efficace, utilisant notamment les techniques de Bostan et Schost pour l'évaluation et l'interpolation de polynômes de grand degré [BS05].

Il faut toutefois noter que, comme il se doit pour une variante du « pas de bébé, pas de géant », la complexité en espace de l'algorithme est aussi en $O(\sqrt{p'})$, ce qui limite la portée pratique de l'attaque. Imaginer une variante en espace constant, à la manière de l'algorithme lambda de Pollard, est un problème ouvert intéressant.

2.2.6 Factorisation de nombres RSA déséquilibrés partiellement connus [BNT09]

Un travail supplémentaire non présenté en détails dans ce manuscrit a concerné la factorisation des entiers RSA déséquilibrés (ayant un facteur beaucoup plus grand que l'autre), popularisés par un célèbre article de Shamir dans *CryptoBytes* [Sha95].

Plus précisément, il s'est agi de déterminer dans quelle mesure la connaissance d'une sous-chaîne de bits du grand facteur premier p permet de factoriser un nombre RSA

déséquilibré $N = pq$ en temps polynomial. On peut interpréter cette question tant comme un problème théorique sur la complexité de la factorisation avec oracle que comme une préoccupation pratique sur la sécurité de RSA dans des implémentations physiques présentant des fuites ou pouvant faire l'objet d'attaques de type *cold boot*.

Nous établissons que la connaissance de $2\lceil \log_2 q \rceil$ bits consécutifs de p suffisent à factoriser $N = pq$ en temps polynomial, quelle que soit la position de cette chaîne de bits. Par ailleurs, nous pouvons montrer que selon la position de la chaîne, un nombre plus faible de bits peut suffire.

Ces résultats sont à nouveau obtenus par une technique de type Coppersmith multivarié. Plus précisément, nous montrons que la factorisation peut se ramener à la résolution d'une équation quadratique de la même forme que celle de Boneh et Durfee [BD00], dont on sait prouvablement retrouver les petites racines d'après les résultats de Bauer et Joux [BJ07].

2.2.7 Falsification de signatures RSA à padding affine [CNT11a]

Un autre résultat un peu particulier, et non repris en détails dans ce manuscrit, a concerné l'étude des paddings affines pour les signatures RSA. Ces paddings, sans doute les plus simples qu'on puisse imaginer, encodent un message m à signer par RSA sous la forme $\mu(m) = \omega \cdot m + a$ pour certaines constantes ω, a données.

Parce qu'ils sont si simples, ils ont fait l'objet d'une étude assez approfondie depuis les années 1980. Il s'avère que si la taille permise pour m est une trop grande fraction de la taille du module, on peut construire en temps polynomial des relations multiplicatives entre les paddings de plusieurs messages, et donc construire des falsifications. Une série de publications [dJC85, GM97, BCCN01] a progressivement réduit cette fraction de 2/3 à 1/2 puis 1/3, meilleure borne connue aujourd'hui. Cependant, on conjecture que cette borne devrait pouvoir être encore améliorée, par exemple à 1/4 de la taille du module. Ce problème reste ouvert depuis une décennie.

Sans fournir une solution à ce problème ouvert, nous donnons un certain nombre de résultats nouveaux tentant d'avancer dans cette direction en tirant de nouvelles conséquences des méthodes de fractions continuées de Brier et al. [BCCN01].

Nous montrons tout d'abord qu'il est possible de falsifier en temps polynomial des signatures à padding affine sur des messages dont l'*entropie* est 1/4 de la taille du module, bien qu'ils soient de longueur plus grande que cette fraction. Nous montrons également comment on peut construire en pratique (en temps superpolynomial mais bien plus rapide que celui nécessaire à factoriser le module) une relation multiplicative entre quatre messages dont les longueurs sont respectivement 1/4, 1/4, 1/4 et 3/8 de la taille du module.

Nous montrons également que des falsifications à 1/4 peuvent être obtenues en temps polynomial dans des scénarios particuliers, notamment lorsqu'il est possible de signer avec deux paddings affines indépendants, ou encore lorsque les bits de poids fort du module sont choisis de façon malveillante.

2.2.8 Problèmes ouverts et perspectives

Bien que la communauté cryptologique s'accorde à ne plus recommander que l'utilisation de paddings RSA prouvés, leur adoption industrielle reste modeste. Par conséquent, la cryptanalyse des paddings RSA ad hoc, bien qu'elle soit un sujet de recherche moins actif qu'il y a quelques années, demeure très pertinente, et nombre de schémas à la sécurité qu'on peut considérer « suspecte » n'ont pas encore fait l'objet d'attaques pratiques. On peut par exemple citer l'exemple des *signatures* PKCS#1 v1.5, qui à l'inverse du chiffrement PKCS#1 v1.5 ou des signatures ISO/IEC 9796-2 n'ont pas encore été cassées. Il est concevable que certaines des techniques utilisées dans cette thèse puissent s'y appliquer, même si l'angle d'attaque semblent pour le moment nous échapper.

S'agissant des attaques par fautes, les attaques sur le module que nous avons introduites ne peuvent avoir qu'un impact pratique limité sous leur forme actuelle car elles ne s'appliquent pas lorsque l'interpolation par les restes chinois s'effectue par la formule de Garner, ce qui est très souvent le cas dans les applications. Il serait très intéressant d'essayer de généraliser ce nouveau type d'attaques à une implémentation des signatures RSA protégée selon l'état de l'art. D'autre part, l'idée de perturber le module semble pouvoir s'appliquer de manière fructueuse à d'autres contextes, comme celui du logarithme discret aussi bien dans un corps fini que sur une courbe elliptique : il y a là aussi des pistes intéressantes à explorer.

2.3 Autres travaux

Nous avons par ailleurs mené quelques travaux qui ne s'inscrivent pas dans l'un des deux grands axes esquissés jusqu'ici, principalement parce qu'ils ne relèvent pas de l'un des deux problèmes difficiles que sont RSA et le logarithme discret dans les courbes elliptiques. Ces travaux ne sont pas présentés en détails dans ce manuscrit, mais nous en donnons les grandes lignes ci-après.

2.3.1 Chiffrement totalement homomorphe sur les entiers [CMNT11, CNT11b]

L'une des avancées théoriques les plus marquantes en cryptographie ces dernières années a certainement été la construction par Gentry en 2009 du premier schéma de chiffrement totalement homomorphe [Gen09].

Un schéma de chiffrement à clef publique est dit *homomorphe* lorsqu'il permet de calculer publiquement sur des données chiffrées. Par exemple le chiffrement « textbook RSA » est *multiplicativement homomorphe* car connaissant les chiffrés de deux entiers, on peut calculer un chiffré de leur produit (à savoir le produit des chiffrés) sans avoir à déchiffrer ni connaître la clef privée. De la même façon, le chiffrement de Paillier est *additivement homomorphe*, et ainsi de suite.

Dès 1978, alors que RSA avait à peine été inventé, Rivest, Adleman et Dertouzos [RAD78] ont imaginé qu'il puisse exister un schéma de chiffrement *totalement homomorphe*, c'est-à-dire pour lequel on puisse appliquer aux chiffrés non pas juste des additions

ou juste des multiplications, mais n'importe quelle fonction efficacement calculable. Un tel schéma aurait de nombreuses applications pratiques, permettant notamment de sous-traiter des calculs à des serveurs distants sans que ces serveurs n'acquièrent d'information sur les données traitées, qui peuvent rester confidentielles. Un tel scénario semblait attirant en 1978, et l'est encore davantage de nos jours à l'heure du *cloud computing*. Toutefois, Rivest et al. ne sont pas parvenus à exhiber de schéma sûr satisfaisant cette propriété, et le problème est resté ouvert pendant plus de 30 ans jusqu'aux travaux de Gentry.

Depuis, plusieurs schémas ont été proposés : celui original de Gentry, dont la sécurité est basée sur des problèmes de réseaux idéaux (ICP), celui de van Dijk, Gentry, Halevi et Vaikuntanathan [vDGHV10] reposant sur le problème du diviseur commun approché, et plus récemment quelques autres, notamment ceux dûs à Brakerski et Vaikuntanathan basés sur des problèmes de réseaux mieux compris comme LWE [BV11].

Cependant, aucun de ces schémas n'est utilisable en pratique. Les tailles de clefs nécessaires pour assurer un niveau de sécurité raisonnable, notamment s'agissant des deux premiers schémas, sont gigantesques (plus de 2^{60} bits au minimum pour le schéma de van Dijk et al., par exemple) et la complexité des opérations homomorphes sur les chiffrés sont du même ordre.

Compte tenu de l'importance à la fois théorique et pratique du chiffrement totalement homomorphe, beaucoup d'efforts ont été menés pour améliorer l'efficacité de ces schémas. S'agissant du schéma original de Gentry, on peut citer les travaux de Smart et Vercauteren [SV10] et de Stehlé et Steinfeld [SS10], qui ont abouti à la première implémentation véritable du schéma par Gentry et Halevi [GH11]. Cette implémentation est encore loin d'être « pratique », avec une clef publique de plusieurs giga-octets et environ une demi-heure pour multiplier les chiffrés de deux bits, mais elle représente une réelle avancée, en ce qu'elle concrétise pour la première fois le chiffrement totalement homomorphe.

Nos contributions à ce champ de recherche comprennent une amélioration importante du schéma de van Dijk et al. sur les entiers (réduisant en particulier considérable la taille des clefs) et sa première implémentation. Pour cela, il nous a fallu modifier l'algorithme de chiffrement pour en quelque sorte combiner multiplicativement des éléments de la clef publique et ainsi se contenter d'en avoir un nombre plus faible dans la clef. La preuve de sécurité de ce schéma modifié met en jeu des résultats sur le nombre de points des hypersurfaces algébriques de grande dimension sur les corps finis.

Certains de nos travaux en cours plus récents concernent également le chiffrement totalement homomorphe sur les entiers : nous obtenons une nouvelle réduction importante de la taille de clef, et nous sommes en mesure d'adapter à ce contexte les nouvelles techniques de Brakerski, Gentry et Vaikuntanathan sur le chiffrement homomorphe « sans bootstrapping » [BGV11].

2.3.2 Génération efficace et quasi-uniforme de nombres premiers [FT11]

Dans une direction très différente, nous nous sommes intéressés à la génération aléatoire efficace de nombres premiers. On peut imaginer diverses mesures d'un algorithme de génération de nombres premiers, telles que sa rapidité, sa précision (la probabilité qu'un

nombre produit soit en fait composé), ses propriétés statistiques (la régularité de sa distribution de sortie), la quantité d'aléa qu'il consomme pour produire un nombre premier (car de l'aléa de qualité n'est pas aisé à obtenir), etc.

On trouve dans la littérature un certain nombre de travaux consacrés à l'accélération de la génération de nombres premiers [BDL91, BD92, JPV00, JP06] ou à la construction d'algorithmes de génération fournissant en outre une preuve de primalité [Mau89, Mau95, Mih94].

Parmi ces articles, certains démontrent des bornes inférieures à l'entropie des distributions de sortie des algorithmes qu'ils considèrent, en utilisant généralement des conjectures très fortes sur la distribution des nombres premiers, comme la conjecture des r-uples de nombres premiers de Hardy-Littlewood. Toutefois, de telles bornes ne suffisent pas à assurer que la distribution obtenue soit statistiquement proche de la distribution uniforme sur les nombres premiers de la taille considérée ; et l'on peut en fait généralement montrer avec l'aide des mêmes conjectures que la distance statistique entre ces distributions n'est pas négligeable (par exemple qu'elle est minorée par une constante : nous obtenons un résultat de ce type dans notre article).

Or certains protocoles cryptographiques particuliers, notamment basés sur l'hypothèse Strong RSA, requièrent explicitement des nombres premiers uniformes pour que les preuves de sécurité fonctionnent. D'autre part, pour une application plus commune comme la génération de module RSA, il semble intuitivement préférable d'avoir des facteurs premiers choisis sans biais, même si l'existence d'un biais statistique ou d'un distingueur explicite entre la distribution des nombres premiers et l'uniforme ne fournit pas immédiatement d'attaque sur la factorisation.

L'algorithme le plus simple imaginable pour générer des nombres premiers uniformes consiste simplement à tirer des entiers (disons impairs) de taille voulue uniformément au hasard, de tester leur primalité, et de recommencer si l'on a tiré un nombre composé. La distribution de sortie est exactement uniforme, mais cette méthode est relativement peu efficace et a surtout le défaut de consommer une quantité considérable d'aléa. À l'inverse, l'algorithme PRIMEINC étudié par Brandt et Damgård [BD92] (essentiellement, prendre un nombre au hasard et incrémenter jusqu'à tomber sur un premier) ou l'algorithme de Joye, Paillier et Vaudenay [JPV00, JP06] consomment peu d'aléa (seulement au début de l'algorithme) mais ont l'inconvénient d'avoir une distribution assez éloignée de l'uniforme.

Dans ce travail, nous introduisons un nouvel algorithme de génération de premiers tentant de combiner les avantages de ces deux approches opposées : il consomme relativement peu d'aléa, et la distance statistique à la distribution uniforme est prouvablement contrôlée (en admettant l'hypothèse de Riemann étendue, ce qui est moins audacieux que la conjecture de Hardy-Littlewood). Il a par ailleurs l'avantage d'être au moins aussi rapide que l'algorithme de Joye-Paillier.

La majoration de la distance statistique utilise des majorations de sommes de caractères de façon très analogue à nos travaux sur les encodages bien distribués (chapitre 6) : il est intéressant de voir de telles techniques s'appliquer à des contextes aussi différents.

2.4 Liste des publications

2.4.1 Articles de revues

[BNNT11b] Modulus Fault Attacks Against RSA Signatures. É. Brier, D. Naccache, P. Q. Nguyen, M. Tibouchi. (JCEN)

[FFS+11] Indifferentiable Deterministic Hashing to Elliptic and Hyperelliptic Curves. R. R. Farashahi, P.-A. Fouque, I. E. Shparlinski, M. Tibouchi, J. F. Voloch. (Math. Comp.)

2.4.2 Articles de conférences

[CNTW09] Practical Cryptanalysis of ISO/IEC 9796-2 and EMV Signatures. J.-S. Coron, D. Naccache, M. Tibouchi, R.-P. Weinmann. (CRYPTO 2009)

[BNT09] Factoring Unbalanced Moduli with Known Bits. É. Brier, D. Naccache, M. Tibouchi. (ICISC 2009)

[CNT10] Fault Attacks Against EMV Signatures. J.-S. Coron, D. Naccache, M. Tibouchi. (CT-RSA 2010)

[BCN+10] On the Broadcast and Validity-Checking Security of PKCS#1 v1.5. A. Bauer, J.-S. Coron, D. Naccache, M. Tibouchi, D. Vergnaud. (ACNS 2010, *best student paper*)

[JTV10] Huff's Model for Elliptic Curves. M. Joye, M. Tibouchi, D. Vergnaud. (ANTS-IX)

[FT10b] Estimating the Size of the Image of Deterministic Hash Functions to Elliptic Curves. P.-A. Fouque, M. Tibouchi. (LATINCRYPT 2010)

[BCI+10a] Efficient Indifferentiable Hashing into Ordinary Elliptic Curves. É. Brier, J.-S. Coron, T. Icart, D. Madore, H. Randriam, M. Tibouchi. (CRYPTO 2010)

[FT10a] Deterministic Encoding and Hashing to Odd Hyperelliptic Curves. P.-A. Fouque, M. Tibouchi. (Pairing 2010)

[CJM+11] Cryptanalysis of the RSA Subgroup Assumption from TCC 2005. J.-S. Coron, A. Joux, A. Mandal, D. Naccache, M. Tibouchi. (PKC 2011)

[CMNT11] Fully Homomorphic Encryption over the Integers with Shorter Public Keys. J.-S. Coron, A. Mandal, D. Naccache, M. Tibouchi. (CRYPTO 2011)

[BNNT11a] Modulus Fault Attacks Against RSA Signatures. É. Brier, D. Naccache, P. Q. Nguyen, M. Tibouchi. (CHES 2011)

2.4.3 Articles d'exposition et chapitres de livres

[CT11] Securing E-passports with Elliptic Curves. H. Chabanne, M. Tibouchi. (IEEE Security & Privacy)

[NT11] Lattice-Based Fault Attacks on Signatures. P. Q. Nguyen, M. Tibouchi. (Fault Analysis in Cryptography)

[Tib11] A Nagell Algorithm in Any Characteristic. M. Tibouchi. (Festschrift J.-J. Quisquater)

2.4.4 Prépublications

[CNT11a] Another Look at RSA Signatures With Affine Padding. J.-S. Coron, D. Naccache, M. Tibouchi.

[CNT11b] Optimization of Fully Homomorphic Encryption. J.-S. Coron, D. Naccache, M. Tibouchi.

[FT11] Close to Uniform Prime Number Generation With Fewer Random Bits. P.-A. Fouque, M. Tibouchi.

Part One

Contributions to Elliptic Curve Cryptography

Overview

In 1985, Koblitz [Kob87] and Miller [Mil85] independently proposed the use of elliptic curves in public-key cryptography. The main advantage of elliptic curve systems stems from the absence of a subexponential-time algorithm to compute discrete logarithms on general elliptic curves over finite fields. Consequently, one can use an elliptic curve group that is smaller in size compared with systems based on either integer factorization or the discrete log problem in the multiplicative group of a finite field, while maintaining the same (heuristic) level of security.

Despite initial misgivings about the potential for unforeseen, mathematically sophisticated attacks, elliptic curve cryptography became widely accepted by the cryptographic community in the early 2000s [KKM11], particularly owing to the advent of pairing-based cryptography: protocols such as Joux's tripartite key agreement [Jou00] and especially the first efficient construction by Boneh and Franklin of an identity-based encryption scheme [BF01] quickly won cryptographers over. Since then, elliptic curves have also made important inroads in the industry, with the standardization of multiple elliptic curve-based cryptographic primitives [ISO18033–2, ANSI X9.62, ANSI X9.63] (ECDSA [FIPS186–3] in particular has been widely adopted), agencies weighing in favor of their use [NSA05], and new embedded applications such as secure e-passports mandating them [ICAO10].

This part of the thesis presents contributions to two areas of elliptic curve cryptography. On the one hand, we have used a range of tools from algebraic geometry and algebraic number theory to construct and analyze so-called encoding functions to elliptic curves, which are building blocks in a number of elliptic curve-based protocols (signatures, password-based authenticated key exchange, encryption and more). Chapters 3–7 are devoted to our work on this topic, some of which has found quick concrete applications. On the other hand, we have studied the more basic problem of finding representations

of elliptic curves with efficient arithmetic operations, specifically looking into an elliptic curve model introduced by G.B. Huff in 1948. This work is covered in Chapter 8.

Contents

Chapter 3

Constant-Time Hashing to Elliptic and Hyperelliptic Curves

3.1 Introduction

This chapter introduces the general problem of constant-time hashing to elliptic and hyperelliptic curves, and discusses on several examples why this problem is of interest to elliptic curve-based cryptographic protocols. It also presents a round-up of the various constant-time encoding functions to elliptic and hyperelliptic curves that have been proposed to this day to obtain hash functions.

3.1.1 Background

Many elliptic curve-based cryptographic protocols require hashing to the elliptic curve group $\mathbb{G}$: they involve one or more hash functions $\mathfrak{H} : \{0,1\}^* \to \mathbb{G}$ mapping arbitrary values to points on the elliptic curve.

For example, in the Boneh-Franklin identity-based encryption scheme [BF01], the public key for identity $\mathrm{id} \in \{0,1\}^*$ is a point $\mathbf{Q}_{\mathrm{id}} = \mathfrak{H}(\mathrm{id})$ on the curve. This is also the case in many other pairing-based cryptosystems including IBE and HIBE schemes [BZ04, GS02, HL02], signature and identity-based signature schemes [Bol03, BGLS03, BLS01, CC03, ZK02] and identity-based signcryption schemes [Boy03, LQ04].

Hashing into elliptic curves is also required for some passwords-based authentication protocols such as the SPEKE [Jab96] and PAK [BMP00] protocols, as well as various signature schemes based on the hardness of the discrete logarithm problem, like [CM05], when they are instantiated over elliptic curves.

In all of those cases, the hash functions are modeled as random oracles [BR93] in security proofs. However, it is not clear how such a hash function can be instantiated in practice. Indeed, random oracles to groups like $\mathbb{Z}_p^*$ can be easily constructed from random oracles to fixed-length bit strings, for which conventional cryptographic hash functions usually provide acceptable substitutes. On the other hand, constructing random oracles

to an elliptic curves even from random oracles to bit strings appears difficult in general, and some of the more obvious instantiations actually break security completely.

3.1.2 Outline

We first present in §3.2 a naive construction of an elliptic curve-valued hash function and show on a concrete example how this naive construction breaks security completely. We then introduce in §3.3 a better solution, the so-called "try-and-increment" hashing, that has satisfactory black-box security properties (it preserves random oracle proofs of security). However, it has the drawback of not running in constant time, which, as we shall see, can be a security concern for physical implementations. We then discuss in §3.4 another strategy for constructing elliptic curve-valued hash functions, based on simpler building blocks called encodings. In §3.5, we turn to the problem of constructing elliptic and hyperelliptic curve encodings. Finally, in §3.6, we discuss the extent to which those constructions actually solve our original hashing problem, as well as some practical applications that those constructions have found in the industry.

3.2 The Trivial Encoding: Totally Insecure

To gain a sense of why the construction of hash functions to elliptic curves requires some care, we first show how a naive construction can completely break the security of a protocol that uses it.

3.2.1 A naive construction

We would like to construct a hash function $\mathfrak{H} : \{0,1\}^* \to \mathbb{G}$ to an elliptic curve group $\mathbb{G}$, which we can assume is cyclic of order N and generated by a given point $\mathbf{G}$. The simplest, most naive way to do so is probably to start from an integer-valued hash function $\mathfrak{h} : \{0,1\}^* \to \mathbb{Z}_N$ (for which reasonable instantiations are easy to come by) and to define $\mathfrak{H}$ as:

$$\mathfrak{H}(m) = [\mathfrak{h}(m)] \cdot \mathbf{G}. \tag{3.1}$$

This is, however, a bad idea on multiple levels.

On the one hand, it is easy to see why this will typically break security *proofs* in the random oracle model. Indeed, at some point in a random oracle model security reduction, the simulator will typically want to "program" the random oracle by setting some of its outputs to specific values. In this case, it will want to set the value $\mathfrak{H}(m)$ for some input m to a certain elliptic curve point $\mathbf{P}$. However, if $\mathfrak{H}$ is defined as in (3.1), the simulator should actually program the integer-valued random oracle $\mathfrak{h}$ to satisfy $[\mathfrak{h}(m)] \cdot \mathbf{G} = \mathbf{P}$. In other words, it should set $\mathfrak{h}(m)$ to the discrete logarithm of $\mathbf{P}$ with respect to $\mathbf{G}$. But this discrete logarithm isn't usually known to the simulator, and it cannot be computed efficiently: therefore, the security reduction breaks down.

On the other hand, it is often not clear how this problem translates into an actual security weakness for a protocol using the hash function $\mathfrak{H}$: one could think that it is

- $\mathsf{KeyGen}()$: Pick $x \xleftarrow{\$} \mathbb{Z}_p$ as the private key, and $\mathbf{P} \leftarrow [x] \cdot \mathbf{G}$ as the public key.
- $\mathsf{Sign}(m, x)$: Compute the signature as $\mathbf{S} \leftarrow [x] \cdot \mathfrak{H}(m)$.
- $\mathsf{Verify}(m, \mathbf{S}, \mathbf{P})$: accept if and only if $e(\mathfrak{H}(m), \mathbf{P}) = e(\mathbf{S}, \mathbf{G})$.

Figure 3.1: The BLS signature scheme.

mostly an artifact of the security proof. Nevertheless, a construction like (3.1) makes it possible for an adversary to compute the discrete logarithm of $\mathfrak{H}(m)$ whenever m is known, which certainly feels uncomfortable from a security standpoint. We demonstrate below that this discomfort is entirely warranted, by showing that the Boneh-Lynn-Shacham signature scheme [BLS01]—certainly the best-known signature scheme that involves hashing to elliptic curves—becomes completely insecure if the hash function involved is instantiated as in (3.1).

3.2.2 BLS signatures

Proposed by Boneh, Lynn and Shacham in 2001 [BLS01], the BLS signature scheme remains the efficient scheme which achieves the shortest signature length to this day: about 160 bits at the 80-bit security level[1].

It is also quite simple to describe. The public parameters are a cyclic group $\mathbb{G}$ of prime order p endowed with a symmetric[2] non degenerate bilinear pairing $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ and a hash function $\mathfrak{H} : \{0,1\}^* \to \mathbb{G}$. A generator $\mathbf{G}$ of the group is also fixed.

In practical instances, $\mathbb{G}$ is a prime order subgroup in the group of rational points of a supersingular elliptic curve over a finite field and e is the modified Weil pairing. Therefore, we denote the group law of $\mathbb{G}$ additively, and that of $\mathbb{G}_T$ multiplicatively. The signature scheme is then as described in Figure 3.1. Boneh, Lynn and Shacham prove that if the Computational Diffie-Hellman problem is hard in $\mathbb{G}$, then this scheme is secure (in the usual sense of existential unforgeability under chosen message attacks) when $\mathfrak{H}$ is modeled as a random oracle.

Now consider the case when $\mathfrak{H}$ is instantiated as in (3.1). Then, the signature on a message m can be written as:

$$\mathbf{S} = [x] \cdot \mathfrak{H}(m) = \big[x\mathfrak{h}(m)\big] \cdot \mathbf{G} = [\mathfrak{h}(m)] \cdot \mathbf{P}$$

[1] The security reduction for the original scheme is not tight (and cannot be made tight, much like that of the Full Domain Hash [Cor00]), so one may want to increase signature size accordingly for really conservative security. However, Katz and Wang [KW03] showed how a slight modification of the scheme makes it possible to obtain tight security without increasing the signature size.

[2] As was noted in the journal version of the BLS paper [BLS04b], the construction extends in a natural way to an asymmetric pairing, and hence to non supersingular pairing-friendly curves, such as Barreto-Naehrig elliptic curves [BN05]. See also [SV07] for a discussion of how this affects the precise statement of the security result.

and hence, one can forge a signature on any message using only publicly available data! There is no security left at all when using the trivial hash function construction.

A slightly less naive variant of the trivial construction consists in defining $\mathfrak{H}$ as:

$$\mathfrak{H}(m) = [\mathfrak{h}(m)] \cdot \mathbf{Q} \tag{3.2}$$

where $\mathbf{Q}$ is an auxiliary public point distinct from the generator $\mathbf{G}$ and whose discrete logarithm α with respect to $\mathbf{G}$ is not published. Using this alternate construction for $\mathfrak{H}$ thwarts the key-only attack described above against BLS signatures. However, the scheme remains far from secure. Indeed, the signature on a message m can be written as:

$$\mathbf{S} = [x\mathfrak{h}(m)] \cdot \mathbf{Q} = [\alpha x \mathfrak{h}(m)] \cdot \mathbf{G} = [\mathfrak{h}(m)] \cdot \alpha\mathbf{P}.$$

Now suppose an attacker knows a valid signature $\mathbf{S}_0$ on some message m_0. Then the signature $\mathbf{S}$ on an arbitrary m is simply:

$$\mathbf{S} = \left[\frac{\mathfrak{h}(m)}{\mathfrak{h}(m_0)}\right] \cdot [\mathfrak{h}(m_0)] \cdot [\alpha]\mathbf{P} = \left[\frac{\mathfrak{h}(m)}{\mathfrak{h}(m_0)}\right] \cdot \mathbf{S}_0$$

where the division is computed in $\mathbb{Z}_p$. Thus, even with this slightly less naive construction, knowing a single valid signature is enough to produce forgeries on arbitrary messages: again, a complete security break down.

3.3 Try-and-Increment: Why Constant Time Matters

A classical construction of a hash function to elliptic curves which does work (and one variant of which is suggested by Boneh, Lynn and Shacham in the original short signatures paper [BLS01]) is the so-called "try-and-increment" algorithm. We present this algorithm here, as well as some of the limitations that explain why different constructions may be preferable.

3.3.1 The try-and-increment algorithm

Consider an elliptic curve E over a finite field $\mathbb{F}_q$ of odd characteristic. It admits a Weierstrass equation of the form:

$$E\colon y^2 = x^3 + ax^2 + bx + c \tag{3.3}$$

for some $a, b, c \in \mathbb{F}_q$. A probabilistic way of constructing points on $E(\mathbb{F}_q)$ is then to pick a random $x \in \mathbb{F}_q$, check whether $t = x^3 + ax^2 + bx + c$ is a square in $\mathbb{F}_q$, and if so, let $y = \pm\sqrt{t}$ and return (x, y). If t is not a square, then x is not the abscissa of a point on the curve: then, one can pick another x and try again, and if so, let $y = \pm\sqrt{t}$ and return (x, y). If t is not a square, then x is not the abscissa of a point on the curve: then, one can pick another x and try again.

It is an easy consequence of the Hasse bound [Has36] on the number of points on $E(\mathbb{F}_q)$ (namely, $|\#E(\mathbb{F}_q) - q - 1| \leqslant 2\sqrt{q}$) that the success probability of a single try is

very close to 1/2. Indeed, if we denote by χ_q the non trivial quadratic character of $\mathbb{F}_q^*$, extended by 0 to $\mathbb{F}_q$ as usual, we have:

$$\#E(\mathbb{F}_q) = 1 + \sum_{x \in \mathbb{F}_q} (1 + \chi_q(x^3 + ax^2 + bx + c)) = q + 1 + \sum_{x \in \mathbb{F}_q} \chi_q(x^3 + ax^2 + bx + c)$$

On the other hand, the success probability ϖ of a single iteration of this point construction algorithm is the proportion of $x \in \mathbb{F}_q$ such that $\chi_q(x^3 + ax^2 + bx + c) = 1$ or 0, namely:

$$\varpi = \frac{\alpha}{2q} + \frac{1}{q} \sum_{x \in \mathbb{F}_q} \frac{1 + \chi_q(x^3 + ax^2 + bx + c)}{2}$$

where $\alpha \in \{0, 1, 2, 3\}$ is the number of roots of the polynomial $x^3 + ax^2 + bx + c$ in $\mathbb{F}_q$. This gives:

$$\varpi = \frac{1}{2} + \frac{\#E(\mathbb{F}_q) - q - 1 + \alpha}{2q} = \frac{1}{2} + O\left(\frac{1}{\sqrt{q}}\right)$$

Now this point construction algorithm can be turned into a hash function based on an $\mathbb{F}_q$-valued random oracle $\mathfrak{h} : \{0,1\}^* \to \mathbb{F}_q$. To hash a message m, the idea is to pick the x-coordinate as, essentially, $\mathfrak{h}(m)$ (which amounts to picking it at random once) and carry out the point construction above. However, since one should also be able to retry in case the first x-coordinate that is tried out is not the abscissa of an actual curve point, we rather let $x \leftarrow \mathfrak{h}(c\|m)$, where c is a fixed length counter initially set to 0 and incremented in case of a failure. Since there is a choice of sign to make when taking the square root of $t = x^3 + ax^2 + bx + c$, we also modify $\mathfrak{h}$ to output an extra bit for that purpose: $\mathfrak{h} : \{0,1\}^* \to \mathbb{F}_q \times \{0,1\}$. This is the try-and-increment algorithm, described more precisely in Algorithm 3.1 (and called MapToGroup in [BLS01][3]). The failure probability after up to k iterations is about 2^{-k} by the previous computations, so choosing the length of the counter c to be large enough for up to $k \approx 128$ iterations, say, is enough to ensure that the algorithm succeeds except with negligible probability.

Boneh, Lynn and Shacham prove that this construction can replace the random oracle $\mathfrak{H} \colon \{0,1\}^* \to E(\mathbb{F}_q)$ in BLS signatures without compromising security. In fact, it is not hard to see that it is *indifferentiable* from such a random oracle, in the sense of Maurer, Renner and Holenstein [MRH04]: this ensures that this construction can be plugged in many protocols[4] requiring a random oracle $\mathfrak{H} \colon \{0,1\}^* \to E(\mathbb{F}_q)$ while preserving random oracle security proofs—this will be discussed in more details in Chapter 5.

Nevertheless, there are various reasons why Algorithm 3.1 is not a completely satisfying construction for hash functions to elliptic curves. There is arguably a certain lack of

[3]Boneh et al. were in fact concerned with hashing to a supersingular curve of characteristic 3 of the form $y^2 = x^3 + 2x \pm 1$. In this case, it was later observed by Barreto and Kim [BK01] that picking y at random and solving the resulting Artin-Schreier equation for x was actually much more efficient, as that equation can be seen as a linear system over $\mathbb{F}_3$. But the basic principle of trying a value and incrementing a counter in case of failure remains the same.

[4]Not necessarily *all* protocols, as conventional wisdom would have it until recently, but at least all protocols with single-stage security games, as clarified by Ristenpart, Shacham and Shrimpton [RSS11].

Algorithm 3.1 The try-and-increment algorithm.

```
1: procedure TryAndIncrementHash(m)      ▷ hash to E: y^2 = x^3 + ax^2 + bx + c
2:     c ← 0                              ▷ c is represented as a ⌈log_2 k⌉-bit bit string
3:     (x, b) ← 𝔥(c‖m)                    ▷ 𝔥 is a random oracle to F_q × {0, 1}
4:     t ← x^3 + ax^2 + bx + c
5:     if t is a square in F_q then
6:         y ← (−1)^b · √t      ▷ define √· as the smaller square root wrt some ordering
7:         return (x, y)
8:     else
9:         c ← c + 1
10:        if c < k then
11:            goto step 3
12:        end if
13:    end if
14:    return ⊥
15: end procedure
```

mathematical elegance in the underlying idea of picking x-coordinates at random until a correct one is found, especially as the length of the counter, and hence the maximum number of trials, has to be fixed (to prevent collisions). More importantly, this may have adverse consequences for the security of physical devices implementing a protocol using this construction: for example, since the number of iterations in the algorithm depends on the input m, an adversary can obtain information on m by measuring the running time or the power consumption of a physical implementation.

3.3.2 Timing attacks on key agreement protocols

A concrete situation in which this varying running time can be a serious issue is the case of embedded devices (especially e-passports) implementing an elliptic curve-based Password-Authenticated Key Exchange (PAKE) protocol.

PAKE is a method for two parties sharing a common low-entropy secret (such as a four-digit PIN, or a self-picked alphabetic password) to derive a high-entropy session key for secure communication in an authenticated way. One of the main security requirements is, informally, that an attacker should not be able to gain any information about the password, except through a brute force online dictionary attack (i.e. impersonating one of the parties in the protocol and attempting to authenticate with each password, one password at a time), which can be prevented in practice by latency, smart card blocking and other operational measures. In particular, a PAKE protocol should be considered broken if a *passive* adversary can learn any information about the password.

Now consider the PAKE protocol described in Figure 3.2, which is essentially Jablon's Simple Password-base Exponential Key Exchange (SPEKE) [Jab96] implemented over an elliptic curve, except with a random salt as suggested in [Jab97]. The public parameters are an elliptic curve group $\mathbb{G}$ of prime order p and a hash function $\mathfrak{H}\colon \{0,1\}^* \to \mathbb{G}$.

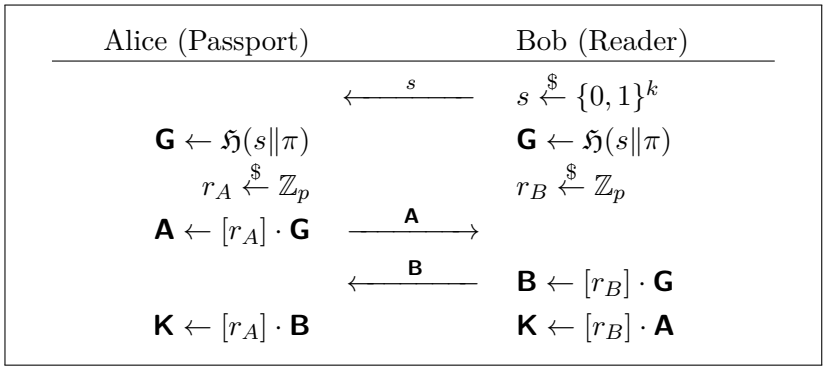


Figure 3.2: A randomized variant of the SPEKE protocol.

The two parties share a common password π, and derive a high-entropy $\mathbf{K} \in \mathbb{G}$ using Diffie-Hellman key agreement in $\mathbb{G}$ but with a variable generator $\mathbf{G} \in \mathbb{G}$ computed by hashing the password.

But if the hash function $\mathfrak{H}$ is instantiated by the try-and-increment construction and an eavesdropper is able to measure the running time of one of the parties, he will find different running times or different power traces depending on how many trials it takes to find a suitable x-coordinate in the computation of $\mathfrak{H}(s\|\pi)$. Since it takes a single iteration with probability close to $1/2$, an execution of the protocol provides at least one bit of information about π to the adversary (and about $-\sum_{k\geqslant 1} 2^{-k} \log_2(2^{-k}) = 2$ bits on average).

This leads to a so-called "partition attack", conceptually similar to those described by Boyd et al. in [BMN01]: the adversary can count the number of iterations needed to compute $\mathfrak{H}(s\|\pi_0)$ for each password π_0 in the password dictionary, keeping only the π_0's for which this number of iterations matches the side-channel measurement. This reduces the search space by a factor of at least 2 (and more typically 4) for each execution of the protocol, as the running times for different values of s are independent. As a result, the eavesdropper can typically reduce his search space to a single password after at most a few dozen executions of the protocol!

A rather inefficient countermeasure that can be considered is to run all k iterations of the try-and-increment algorithm every time. However, even that is probably insufficient to thwart the attack: indeed, the usual algorithm (using quadratic reciprocity) for testing whether an element of $\mathbb{F}_q$ is a square, as is done in Step 5 of Algorithm 3.1, also has different running times depending on its input. This can be provide information to the adversary as well, unless this part is somehow tweaked to run in constant time, which seems difficult to do short of computing the quadratic character with an exponentiation and making the algorithm prohibitively slow with k exponentiations every time. In principle, padding the quadratic reciprocity-based algorithm with dummy operations might provide a less computationally expensive solution, but implementing such a countermeasure securely seems quite daunting. A construction that naturally runs in constant time would certainly be preferable.

3.4 Encoding to Elliptic Curves

3.4.1 Main idea

A natural way to construct a constant-time hash function to an elliptic curve E would be use, as a building block, a suitable function $f\colon \mathbb{F}_q \to E(\mathbb{F}_q)$ that can be efficiently computed in constant time[5]. Then, combining f with a hash function $\mathfrak{h}\colon \{0,1\}^* \to \mathbb{F}_q$, we can hope to obtain a well-behaved hash function to $E(\mathbb{F}_q)$.

Of course, not all such functions f are appropriate: for example, when $q = p$ is prime, the trivial encoding described in §3.2 is essentially of that form, with $f\colon u \mapsto [\hat{u}] \cdot \mathbf{G}$ (and $u \mapsto \hat{u}$ any lifting of $\mathbb{F}_p$ to $\mathbb{Z}$).

On the other hand, if f is a bijection between $\mathbb{F}_q$ and $E(\mathbb{F}_q)$ whose inverse is also efficiently computable, then the following construction:

$$\mathfrak{H}(m) = f\big(\mathfrak{h}(m)\big) \tag{3.4}$$

is well-behaved, in the sense that if $\mathfrak{h}$ is modeled as a random oracle to $\mathbb{F}_q$, then $\mathfrak{H}$ can replace a random oracle to $E(\mathbb{F}_q)$ in any protocol while preserving proofs of security in the random oracle model. Indeed, contrary to what happens in the case of the trivial encoding (where programming the random oracle would require computing discrete logarithm), a simulator can easily choose a value $\mathfrak{H}(m_0) = \mathbf{P}_0$ by setting $\mathfrak{h}(m_0) = f^{-1}(\mathbf{P}_0)$. More generally, such a construction is, again, indifferentiable from a random oracle to $E(\mathbb{F}_q)$ (and even *reset indifferentiable* in the sense of Ristenpart et al. [RSS11]).

The same holds if f induces a bijection from $\mathbb{F}_q \setminus T$ to $E(\mathbb{F}_q) \setminus W$ for some finite or negligibly small sets of points T, W.

More generally, we will be considering cases where f is not necessarily an efficiently invertible bijection but only a so-called *samplable* mapping, in the sense that for each $\mathbf{P} \in E(\mathbb{F}_q)$, one can compute a random element of $f^{-1}(\mathbf{P})$ in probabilistic polynomial time.

3.4.2 A simple example

It was actually one of the first papers requiring hashing to elliptic curves, namely Boneh and Franklin's construction [BF01] of identity-based encryption from the Weil pairing, that introduced the first practical example of a hash function of the form (3.4). Boneh and Franklin used elliptic curves of a very special form:

$$E\colon y^2 = x^3 + b \tag{3.5}$$

over a field $\mathbb{F}_q$ such that $q \equiv 2 \pmod 3$. In $\mathbb{F}_q$, $u \mapsto u^3$ is clearly a bijection, and thus each element has a unique cube root. This makes it possible, following Boneh and

[5] It is probably superfluous to point out that, by constant time, we mean "whose running time does not depend on the input" (once the choice of parameters like E and $\mathbb{F}_q$ is fixed), and not $O(1)$ time in the sense of complexity theory.

Franklin, to define a function f as:

$$f\colon \mathbb{F}_q \to E(\mathbb{F}_q) \tag{3.6}$$

$$u \mapsto \Big((u^2 - b)^{1/3}, u\Big). \tag{3.7}$$

In other words, instead of picking the x-coordinate and try to deduce the y-coordinate by taking a square root (which may not exist) as before, we first choose the y-coordinate and deduce the x-coordinate by taking a cube root (which always exists).

Obviously, the function f is a bijection from $\mathbb{F}_q$ to all the finite points of $E(\mathbb{F}_q)$. In particular, this implies that $\#E(\mathbb{F}_q) = 1 + \#\mathbb{F}_q = q + 1$; thus, E is supersingular (and hence comes with a computable symmetric pairing). This also means that f satisfies the conditions mentioned in the previous section; therefore, construction (3.4) can replace the random oracle $\mathfrak{H}$ required by the Boneh-Franklin IBE scheme, or any other protocol proved secure in the random oracle model. And it can also easily be computed in constant time: it suffices to compute the cube root as an exponentiation to a fixed power α such that $3\alpha \equiv 1 \pmod{q-1}$.

Note that in fact, the group $\mathbb{G}$ considered by Boneh and Franklin isn't $E(\mathbb{F}_q)$ itself, but a subgroup $\mathbb{G} \subset E(\mathbb{F}_q)$ of prime order. More precisely, the cardinality q of the base field is chosen of the form $6p - 1$ for some prime $p \neq 2, 3$. Then $E(\mathbb{F}_q)$ has a unique subgroup $\mathbb{G}$ of order p (and index 6), which is the group actually used in the scheme. Hashing to $\mathbb{G}$ rather than $E(\mathbb{F}_q)$ is then easy:

$$\mathfrak{H}(m) = f'(\mathfrak{h}(m)) \quad \text{where} \quad f'(u) = [6] \cdot f(u). \tag{3.8}$$

The encoding f' defined in that way isn't injective but it is samplable: indeed, to compute a random preimage of some point $\mathbf{P} \in \mathbb{G}$, we can simply compute the six points $\mathbf{Q}_i$ such that $[6] \cdot \mathbf{Q}_i = \mathbf{P}$, and return $f^{-1}(\mathbf{Q}_i)$ for a random index i. Using that observation, Boneh and Franklin prove that construction (3.8) can replace the random oracle to $\mathbb{G}$ in their IBE scheme. More generally, it is easy to see that it is indifferentiable from a random oracle in the sense of Chapter 5.

3.4.3 Beyond supersingular curves

The previous example suggests that a sensible first step towards constructing well-behaved constant-time hash functions to elliptic curves is to first obtain mappings $f\colon \mathbb{F}_q \to E(\mathbb{F}_q)$ that are computable in deterministic polynomial time and samplable, and admit constant-time implementations. We will refer to such mappings as *encoding functions* or simply *encodings*[6]. Note that despite what the name might suggest, there is no assumption of injectivity for those mappings.

It turns out that constructing encodings to elliptic curves beyond special cases such as 3.6 is far from an easy task. In fact, Schoof mentioned the presumably easier problem

[6]We do not attempt to formulate a very precise definition for those terms, as we will be using them rather informally. Precise statements will be made about more restricted (and properly defined) classes of encodings in later chapters.

of constructing a *single* non-identity point on a general elliptic curve over a finite field (the Hasse bound ensures that there is always such a point over fields of cardinality at least 5) as open in his 1985 paper on point counting [Sch85], and little progress was made on this problem before the 2000s.

The first significant result in that direction was obtained by Schinzel and Skałba in 2004 [SS04]. They exhibited one non-identity point on the elliptic curve (3.5) without any assumption on the cardinality of the base field. A part of their result is as follows.

Theorem 3.1 ([SS04, Th. 1])**.** *Let $\mathbb{F}_q$ be a finite field of characteristic at least 5 and b an element of $\mathbb{F}_q$ such that $b^3 + 72^3 \neq 0$. Further set:*

$$\begin{aligned}
y_1 &= -2^{-9}3^{-5}b^3 + 2^{-6}3^{-3}b^2 - 2^{-3}b - 3 \\
y_2 &= 2^{-8}3^{-6}b^3 - 2^{-5}3^{-3}b^2 + 2^{-2}3^{-1}b + 2 \\
y_3 &= \frac{b^6 - 2^5 3^2 b^5 + 2^6 3^6 b^4 - 2^{10}3^6 5 b^3 + 2^{12}3^8 5 b^2 - 2^{16}3^{11}b + 2^{18}3^{12}}{2^8 3^5 (b+72)^3} \\
y_4 &= \frac{b^9 - 2^3 3^2 7 b^8 + 2^9 3^5 b^7 - 2^{13}3^7 b^6 + 2^{13}3^8 29 b^5}{2^{10}3^5(b^2 - 72b + 72^2)^3} \\
&\quad + \frac{-2^{17}3^{12}b^4 + 2^{19}3^{13}7b^3 + 2^{22}3^{14}b^2 + 2^{24}3^{17}b + 2^{27}3^{18}}{2^{10}3^5(b^2 - 72b + 72^2)^3}.
\end{aligned}$$

Then for at least one $j = 1, 2, 3, 4$, $y_j^2 - b$ is a cube x^3 in $\mathbb{F}_q$, and hence (x, y) is a non-identity rational point on the elliptic curve $y^2 = x^3 + b$.

This rather contrived result only shows how to construct a single non trivial $\mathbb{F}_q$-point on the special curve $E\colon y^2 = x^3 + b$ in deterministic polynomial time. However, further research in recent years by Skałba and others led to the construction of more or less practical encoding functions to all elliptic curves in recent years. Several approaches have been proposed. We introduce the main ones in the next section.

3.5 Constructing Encodings to Elliptic Curves and Hyperelliptic Curves

3.5.1 The Shallue-van de Woestijne-Ulas approach

Skałba's theorem. The first construction of a constant-time encoding function to a large class of elliptic curves is due to Skałba in 2005 [Ska05], who was trying to generalize the results of his earlier paper with Schinzel [SS04]. He was able to prove a result of the following form.

Theorem 3.2 ([Ska05, Th. 1])**.** *Let $\mathbb{F}_q$ be a finite field of characteristic at least 5, and $g(x) = x^3 + ax + b \in \mathbb{F}_q[x]$ a polynomial over $\mathbb{F}_q$ with $a \neq 0$. Then there exists non constant rational functions $X_1, X_2, X_3, U \in \mathbb{F}_q(x)$ such that the following identity holds in $\mathbb{F}_q(t)$:*

$$g(X_1(t^2))g(X_2(t^2))g(X_3(t^2)) = U(t)^2. \tag{3.9}$$

The rational functions X_i and U are given explicitly, but since they are quite large (the product $X_1X_2X_3$ is a rational function of degree 26 with coefficients that are several hundred bits long), we omit them here. However, let us recall how, from identity (3.9), one can readily deduce an encoding function to the elliptic curve:

$$E\colon y^2 = x^3 + ax + b. \tag{3.10}$$

For any value $u \in \mathbb{F}_q$ such that u^2 is not a pole of the X_i's, we see that the product $g(X_1(u^2))g(X_2(u^2))g(X_3(u^2))$ is a square in $\mathbb{F}_q$, which implies that at least one of the three values $g\big(X_i(u^2)\big)$ is a quadratic residue. If i is the smallest index for which that is the case, we can then set:

$$(x, y) = \Big(X_i(u^2), \sqrt{g(X_i(u^2))}\Big)$$

and (x, y) is then a non trivial point in $E(\mathbb{F}_q)$.

This defines an encoding to any elliptic curve over finite fields of characteristic $\neq 2, 3$, except those of j-invariant 0, so this is already "almost" a complete solution.

A caveat is that no unconditional polynomial deterministic algorithm is known for computing square roots in general finite fields. However, this is never really a problem in cryptographic applications, where it is usually possible to either make mild congruential assumptions on the cardinality of the field (such that $q \not\equiv 1 \pmod 8$, in which case computing square roots is easy) or assume that a fixed quadratic nonresidue is known (and then use an algorithm such as Tonelli-Shanks [Sha73]). And a later result by van de Woestijne ensures that, in this particular setting, the square root can actually be computed in deterministic polynomial time without any restriction on q (see below).

Still, depending on the particular case under consideration, it may be somewhat difficult to write down a constant-time implementation. And the rational functions X_i are complex enough to make them unattractive for practical applications.

Shallue and van de Woestijne's construction. In a paper presented at ANTS in 2006, Shallue and van de Woestijne [SvdW06] proposed a more general construction along the same lines, that applies to all elliptic curves.

Consider the general Weierstrass equation for an elliptic curve in odd characteristic (possibly including 3):

$$E\colon y^2 = x^3 + ax^2 + bx + c.$$

Let further $g(x) = x^3 + ax^2 + bx + c \in \mathbb{F}_q[x]$. It is possible to construct an encoding function to $E(\mathbb{F}_q)$ like before from a rational curve on the three-dimensional variety:

$$V\colon y^2 = g(x_1)g(x_2)g(x_3)$$

(which, geometrically, is the quotient of $E \times E \times E$ by $(\mathbb{Z}/2\mathbb{Z})^2$, where each non trivial element acts by $[-1]$ on two components and by the identity on the third one). Indeed, if $\phi\colon \mathbb{A}^1 \to V$, $t \mapsto \big(x_1(t), x_2(t), x_3(t), y(t)\big)$ is such a rational curve, then for any $u \in \mathbb{F}_q$ that is not a pole of ϕ, at least one of $g\big(x_i(u)\big)$ for $i = 1, 2, 3$ is a quadratic residue, and we obtain a corresponding point on $E(\mathbb{F}_q)$ like before.

Then, Shallue and van de Woestijne show how to construct such a rational curve ϕ (and in fact a large number of them). They first obtain an explicit rational map $\psi : S \to V$, where S is the surface of equation:

$$S\colon y^2 \cdot (u^2 + uv + v^2 + a(u+v) + b) = -g(u)$$

which can also be written, by completing the square with respect to v, as:

$$\left[y(v + \frac{1}{2}u + \frac{1}{2}a)\right]^2 + \left[\frac{3}{4}u^2 + \frac{1}{2}au + b - \frac{1}{4}a^2\right]y^2 = -g(u).$$

Now observe that for any fixed $u \in \mathbb{F}_q$, the previous equation defines a curve of genus 0 in the (v, y)-plane. More precisely, it can be written as:

$$z^2 + \alpha y^2 = -g(u) \tag{3.11}$$

with $z = y(v + \frac{1}{2}u + \frac{1}{2}a)$ and $\alpha = \frac{3}{4}u^2 + \frac{1}{2}au + b - \frac{1}{4}a^2$. This is a non degenerate conic as soon as α and $g(u)$ are both non zero (which happens for all $u \in \mathbb{F}_q$ except at most 5), and then admits a rational parametrization, yielding a rational curve $\mathbb{A}^1 \to S$. Composing with ψ, we get the required rational curve on V, and hence an encoding, provided that $q > 5$.

Two steps in this algorithm are potentially non deterministic: taking the square root required to map a point of $V(\mathbb{F}_q)$ to $E(\mathbb{F}_q)$ (like in Skałba's case), and finding one point on the non degenerate conic defined by (3.11) (which is necessary to obtain a parametrization). Van de Woestijne actually proves in his thesis that both of these computations can be carried out in deterministic polynomial time: on the one hand, he gives a deterministic variant of the Tonelli-Shanks algorithm which, given $a_1, a_2, a_3, b \in \mathbb{F}_q$ such that $a_1 a_2 a_3 = b^2$, returns an index $i \in \{1, 2, 3\}$ and a square root of a_i; on the other hand, he proves that a solution of $ax^2 + by^2 = c$ can be found in deterministic polynomial time for any $a, b, c \in \mathbb{F}_q^*$.

Combining all of those results, we obtain an encoding function to any elliptic curve over a field of odd characteristic with $q > 5$ elements. Moreover, Shallue and van de Woestijne show that the image of this encoding contains at least $(q - 4)/8$ points in $E(\mathbb{F}_q)$.

They also give a variant of this approach for elliptic curves over binary fields. For example, to obtain an encoding to the binary curves with Weierstrass equation:

$$E\colon y^2 + a_3 y = x^3 + a_4 x + a_6$$

they construct rational curves on the following threefold:

$$V\colon y^2 + a_3 y = g(x_1) + g(x_2) + g(x_3)$$

where $g(x) = x^3 + a_4 x + a_6$. Indeed, since the map $y \mapsto y^2 + a_3 y$ is $\mathbb{F}_2$-linear, elements of $\mathbb{F}_q = \mathbb{F}_{2^m}$ of the form $y^2 + a_3 y$ make up a subgroup of index 2 of the additive group of $\mathbb{F}_q$. Hence, if (x_1, x_2, x_3, y) is a point in $V(\mathbb{F}_q)$, at least one of $g(x_i)$, $i = 1, 2, 3$, belongs

to this subgroup, and hence the corresponding x_i is the abscissa of a point in $E(\mathbb{F}_q)$ as before. A similar reasoning, together with the construction of the corresponding rational curves, can deal with both reduced Weierstrass forms in characteristic two.

In the end, Shallue and van de Woestijne obtain an encoding functions to *every elliptic curve* over any finite field $\mathbb{F}_q$ with $q > 5$. The main drawback of their approach is that the computations are relatively cumbersome, and implementing them in constant time is not straightforward. It also requires finding a point on a conic: this is a one-time pre-computation for any given elliptic curve, so not a serious issue from a cryptographic viewpoint, but it does preclude nice formulas with small coefficients for the encoding function, for example.

The contributions of Ulas. In a paper published the following year [Ula07], Ulas considered again the problem of constructing rational curves on the threefold:

$$V\colon y^2 = g(x_1)g(x_2)g(x_3)$$

where, however, g is of the following somewhat special form (slightly more restricted than the one considered by Skałba):

$$g(x) = x^3 + ax + b \quad \text{with } ab \neq 0.$$

He was actually able to construct an explicit rational parametrization for a *surface* embedded in V, which has the advantage of being much simpler than the rational curves of Skałba and Shallue-van de Woestijne. A special case of his result can be written as follows.

Theorem 3.3 ([Ula07, Th. 2.3(2)])**.** *Let $\mathbb{F}$ be any field, and $g(x) = x^3 + ax + b \in \mathbb{F}[x]$ a polynomial over $\mathbb{F}$ with $ab \neq 0$. Define the following bivariate rational functions over $\mathbb{F}$:*

$$\begin{aligned} X_1(t,u) &= u \\ X_2(t,u) &= -\frac{b}{a} \cdot \frac{t^6 g(u)^3 - 1}{t^6 g(u)^2 - t^2 g(u)} \\ X_3(t,u) &= t^2 g(u) \cdot X_2(t,u) \\ U(t,u) &= t^3 g(u)^2 \cdot g(X_2(t,u)). \end{aligned}$$

Then the following identity holds in $\mathbb{F}(t,u)$:

$$g(X_1(t,u))g(X_2(t,u))g(X_3(t,u)) = U(t,u)^2.$$

Composing this rational parametrization of a surface embedded in V with almost any rational map $\mathbb{A}^1 \to \mathbb{A}^2$ gives rise to a rational curve on V and hence an encoding function to the elliptic curve:

$$E\colon y^2 = x^3 + ax + b, \quad ab \neq 0$$

as before, which is simple enough to write down in full. Any elliptic curve over a finite field of characteristic $\neq 2, 3$ and of j-invariant $\neq 0, 1728$ admits such a Weierstrass equation, making these encoding functions suitable for many cryptographic applications. We call them the SWU encodings (for "Shallue-van de Woestijne-Ulas"). They have actually been considered in certain practical cryptographic settings (particularly for a secure e-passport standard), although the arguably simpler encoding proposed by Icart (§3.5.2) was ultimately preferred.

In [BCI+10a], we proposed a further simplification of these formulas, together with an elementary derivation of them. We also extended them to the case of elliptic curves in characteristic 3. See §3.5.4 for details.

It may be interesting to note that the original paper by Ulas has a somewhat more general scope than suggested by our description. He was able to find rational curves on all varieties of the form:

$$V_k \colon y^2 = g(x_1)g(x_2)\cdots g(x_k)$$

for $k \geqslant 2$. This doesn't have further impact on the construction of elliptic curve encodings, however (but see §3.5.3).

The work of Sato and Hakuta. A different but related approach to construct encodings to elliptic curves was later proposed by Sato and Hakuta [SH09]. They consider again elliptic curves over fields of characteristic $\neq 2, 3$, given in short Weierstrass form:

$$E \colon y^2 = x^3 + ax + b.$$

If we put their basic idea in geometric terms, it is again to parametrize a curve on a higher dimensional variety and map it to the elliptic curve, but instead of considering the threefold $E^3/(\mathbb{Z}/2\mathbb{Z})^2$, they use the abelian surface $A = \mathrm{Res}_{\mathbb{F}_{q^2}/\mathbb{F}_q}(E)$ given by the Weil restriction of E from $\mathbb{F}_{q^2}$ to $\mathbb{F}_q$. The norm $N_{\mathbb{F}_{q^2}/\mathbb{F}_q} \colon \mathsf{P} \mapsto \mathsf{P} + \mathsf{P}^\sigma$ then gives a rational map $A \to E$ which can turn a parametrized curve on A into an encoding to E.

Of course, we cannot hope to find a rational curve directly on A, but Sato and Hakuta show in essence that there is a conic on the Kummer variety $K = A/\{\pm 1\}$ that is non-degenerate provided that $a \neq 0$, and whose $\mathbb{F}_q$-points are entirely contained in the image of $A(\mathbb{F}_q) \to K(\mathbb{F}_q)$. Thus, when $a \neq 0$, they obtain a curve on $A(\mathbb{F}_q)$ with an algebraic parametrization, and composing with the norm map, a non-constant algebraic encoding to $E(\mathbb{F}_q)$.

This approach has the same drawbacks as the original Shallue-van de Woestijne technique (cumbersome formulas and the need to find a point on a conic) without the generality and unconditional deterministicness, but it looks as if it might lend itself to interesting generalizations (e.g. to higher-dimensional abelian varieties).

3.5.2 Icart's approach

Icart's encoding. In [Ica09], Icart introduced an encoding function based on a very different idea—namely, trying to adapt the Boneh-Franklin encoding discussed in §3.4.2

$$f\colon \mathbb{F}_q \longrightarrow E(\mathbb{F}_q)$$
$$u \longmapsto \left(\left(v^2 - b - \frac{u^6}{27}\right)^{1/3} + \frac{u^2}{3}\ ;\ ux + v\right)$$

where $v = (a - 3u^4)/(6u)$. By convention, $f(0) = \mathbf{O}$, the identity element.

Figure 3.3: Icart's encoding to $E\colon y^2 = x^3 + ax + b$ over $\mathbb{F}_q$ with $q \equiv 2 \pmod 3$.

to the case of an ordinary elliptic curve. More precisely, consider again an elliptic curve E given by a short Weierstrass equation:

$$E\colon y^2 = x^3 + ax + b$$

over a field $\mathbb{F}_q$ of odd characteristic with $q \equiv 2 \pmod 3$. The idea is again to reduce the equation to a binomial cubic which can be solved directly in $\mathbb{F}_q$ (where $u \mapsto u^3$ is a bijection).

Unlike the simple case considered by Boneh and Franklin, this cannot be done by picking y as a constant: doing so results in a trinomial cubic which does not always have a root in $\mathbb{F}_q$. Icart's idea is to set $y = ux + v$ for two parameters u, v to be chosen later. This gives:

$$x^3 - u^2x^2 + (a - 2uv)x + b - v^2 = 0$$

and after completing the cube:

$$\left(x - \frac{u^2}{3}\right)^3 + \left(a - 2uv - \frac{u^4}{3}\right)x = v^2 - b - \frac{u^6}{27}.$$

Thus, by setting $v = (a - 3u^4)/(6u)$, it is possible to cancel the term of degree 1 and obtain a binomial cubic equation:

$$\left(x - \frac{u^2}{3}\right)^3 = v^2 - b - \frac{u^6}{27}$$

which is easy to solve for x in $\mathbb{F}_q$. This gives Icart's encoding, described in Figure 3.3.

This encoding applies to a slightly more restricted setting than the SWU encodings, due to the requirement that $q \equiv 2 \pmod 3$, but it has the advantage of being very easy to describe and implement in constant time.

As Icart observed, the exact same technique applies in even characteristic. Consider an elliptic curve E of non zero j-invariant over a binary field $\mathbb{F}_q = \mathbb{F}_{2^n}$ with n odd (in which case $u \mapsto u^3$ is again a bijection). E has a reduced Weierstrass equation of the form:

$$E\colon y^2 + xy = x^3 + ax^2 + b$$

$$f\colon \mathbb{F}_{2^n} \longrightarrow E(\mathbb{F}_{2^n})$$
$$u \longmapsto \left(\left(v^4 + v^3 + b\right)^{1/3} + v \;;\; ux + v^2\right)$$

where $v = a + u + u^2$.

Figure 3.4: Icart's binary encoding to $E\colon y^2 + xy = x^3 + ax^2 + b$ over $\mathbb{F}_{2^n}$ with n odd.

with $b \neq 0$. We can set $y = ux + v$ and solve as before. However, we get slightly simpler formulas with $y = ux + v^2$ instead. This gives:

$$x^3 + (a + u + u^2)x^2 + v^2x + b + v^4 = 0$$

and completing the cube:

$$\left(x + (a + u + u^2)\right)^3 + \left(v^2 - (a + u + u^2)^2\right)x = b + v^4 + (a + u + u^2)^3.$$

Thus, the degree 1 term disappears if we let $v = a + u + u^2$, and we can the solve the resulting binomial cubic equation. This gives Icart's binary encoding, described in Figure 3.4, which is even easier to write down than the odd characteristic variant.

Generalizations. Several variants and extensions of Icart's encoding have been proposed, and various algebraic and geometric interpretations of it have been given afterwards. We now give a quick review of these works.

Farashahi [Far11] showed that Icart's technique applies directly to Hessian curves (these are all elliptic curves with a rational point of order 3):

$$E\colon x^3 + y^3 + 1 = dxy$$

over fields $\mathbb{F}_q$ such that $q \equiv 2 \pmod 3$. Indeed, setting $y = ux + v$ again and completing the cube, the equation becomes:

$$(1 + u^3)\left(x + \frac{u^2v - du}{1 + u^3}\right)^3 + 3(u^2v - du)\cdot\left(1 + \frac{u^2v - du}{1 + u^3}\right)x = -1 - v^3 - \frac{(u^2v - du)^3}{(1 + u^3)^2}$$

and we can cancel the term of degree 1 by setting $v = d/u$, and then solve for x. The encoding function obtained in this way is a bit less general than Icart's since it only applies to Hessian curves, but it has certain interesting properties of its own, such as an inverse that is easier to describe.

Kammerer, Lercier and Renault [KLR10] interpreted (perhaps incorrectly) Icart's approach as an application of Cardano's formulas for the resolution of the cubic equation, and attempted to extended in that direction. More precisely, consider a curve of the form:

$$E\colon x^3 + A(y)x + B(y) = 0 \tag{3.12}$$

where A and B are rational functions of sufficiently low degree to ensure that E is of genus 1 (the reduced Weierstrass form corresponds to the case when A is constant and B is a polynomial of degree 2 with no linear term). Over a field $\mathbb{F}_q$ with $q \equiv 2 \pmod 3$, we can hope to parametrize $E(\mathbb{F}_q)$ by solving (3.12) using Cardano's formula:

$$x = r - \frac{A(y)}{3r} \quad \text{where} \quad r^3 = -\frac{B(y)}{2} + \sqrt{\frac{B(y)^2}{4} + \frac{A(y)^3}{27}}.$$

This works when the square root exists, i.e. when the discriminant $A^3/27 + B^2/4$ is always a square. This can be ensured if the curve of equation:

$$4A(y)^3 + 27B(y)^2 = 4 \cdot 27z^2 \tag{3.13}$$

is unicursal, in which case a rational parametrization $u \mapsto \big(y(u), z(u)\big)$ provides an encoding function to $E(\mathbb{F}_q)$ as:

$$u \mapsto \left(r(u) - \frac{A(y(u))}{3r(u)} \ ; \ y(u)\right) \quad \text{where} \quad r(u) = \left(-\frac{B(y(u))}{2} + z(u)\right)^{1/3}. \tag{3.14}$$

Unfortunately, in the usual Weierstrass case when A is constant and B is a polynomial of degree 2, the curve given by (3.13) is clearly of genus 1, and the method does not apply. However, Kammerer et al. noticed that if A is a polynomial of degree 1 and B is chosen as the product of A by another polynomial of degree 1, then A^2 can be factored out in (3.13) and the curve is birational to a non degenerate conic. Therefore, in that case, they obtain a new encoding function to the curve E, which turns out to be (isomorphic to) a Hessian curve again! Remarkably, this encoding is different from the one found by Farashahi, although they share many properties.

Couveignes and Kammerer [CK11] were later able to give a common geometric interpretation of all of these encodings (Icart's, Farashahi's and Kammerer et al.'s) in terms of unicursal curves passing through the nine cusps of the dual curve of an elliptic curve.

3.5.3 Extensions to hyperelliptic curves

The problem of constructing encoding functions extends naturally to any curve (or indeed any variety) over $\mathbb{F}_q$, and the case of hyperelliptic curves, in particular, has received some attention with the hope that it may be relevant to hyperelliptic curve cryptography.

Indeed, most elliptic curve-base cryptographic protocols, including those that require elliptic curve-valued hash functions (e.g. BLS signatures, Boneh-Franklin IBE, certain PAKE protocols, etc.) admit natural generalizations to a hyperelliptic curve-based setting, possibly with some efficiency gains and lower memory footprint (see e.g. [PWGP03]). A caveat is that in the hyperelliptic setting, the group where computations occur is not the set of point of the hyperelliptic curve itself (which does not have a natural group structure) but the Jacobian variety of the curve, which is a higher-dimensional abelian variety. Thus, hashing should be done to the Jacobian rather than to the curve itself.

However, as we have seen, constructing encoding functions to curves can already be quite tricky, so to the best of our knowledge, no attempt has been made to construct encodings to the Jacobian directly: rather, research has focused on encoding to hyperelliptic curves themselves, and then plugging those encodings into more general constructions to obtain Jacobian-valued hash functions. We give a quick review here of the encodings that have been proposed to date.

Hyperelliptic SWU encoding. The technique used by Ulas in [Ula07] to establish Theorem 3.3 generalizes directly to hyperelliptic curves of the form $y^2 = x^n + ax + b$ or $y^2 = x^n + ax^2 + bx$, for $ab \neq 0$ and any odd $n \geqslant 3$ coprime to the characteristic, and Ulas himself actually stated his results in terms of these curves of arbitrary large genus. In particular, the SWU encoding extends to these particular curves. These form two somewhat restricted families of hyperelliptic curves—families of dimension 1 in the $(2g-1)$-dimensional moduli space of genus g hyperelliptic curves for any $g \geqslant 1$—but like the elliptic curve SWU encoding, this encoding has the advantage of being quite easy to describe. Furthermore, the simplified derivation we proposed in [BCI+10a] applies to this hyperelliptic setting as well.

Kammerer et al.*'s hyperelliptic encodings.* In addition to the Hessian curve encoding described above, Kammerer, Lercier and Renault [KLR10] presented several encodings to higher-genus hyperelliptic curve.

They first try to extend the technique of applying Cardano's formula to some genus 2 curves. They find that, for example, when A is constant and B is a rational function of degree 2, then the curve of equation (3.12) is hyperelliptic of genus 2, isomorphic to:

$$H\colon y^2 = \lambda(x^3 + 3\mu x + 2a)^2 + 4v^2 - 4\lambda u^2 \tag{3.15}$$

for some $\lambda, \mu, a, u, v \in \mathbb{F}_q$, and the curve (3.13) associated to the cubic discriminant is of genus 1. It is thus possible to construct an encoding to $H(\mathbb{F}_q)$ when $q \equiv 2 \pmod 3$ by first mapping to the associated genus 1 curve with Icart's encoding, and then deducing a function to $H(\mathbb{F}_q)$ by solving the cubic as in (3.14). The explicit formulas, given in [KLR10, Fig. 3], are rather daunting. However, a nice feature of this encoding is that the target family of genus 2 curves defined by (3.15) is rather large: it is a family of dimension 2 in the 3-dimensional moduli space of genus 2 hyperelliptic curves. This is the largest family of hyperelliptic curves of genus 2 for which an encoding is known (all over families are of dimension at most 1).

Kammerer et al. also give two examples of simpler encodings to restricted families of hyperelliptic curves of arbitrarily large genus. On the one hand, they describe an encoding to hyperelliptic curves of the form:

$$H\colon y^2 = x^{2d} + ax^d + b$$

where $d \geqslant 3$ is an integer coprime to $q(q-1)$ (and a, b are such that the discriminant of the right-hand side is not zero). The idea is simply to write an explicit rational

parametrization of the non degenerate conic $y^2 = w^2 + aw + b$ and deduce an encoding by setting $x = w^{1/d}$.

On the other hand, they use the solvability of De Moivre polynomials to construct an encoding to hyperelliptic curves of the form:

$$H\colon y^2 = x^d + dax^{d-2} + 2da^2x^{d-4} + 3da^3x^{d-6} + \cdots + 2da^{(d-1)/2-1}x^3 + da^{(d-1)/2}x + b$$

with $a, b \in \mathbb{F}_q$ and $d \geqslant 3$ odd and coprime to the characteristic. Both of these families of hyperelliptic curves are of dimension 1 in the moduli space.

Odd hyperelliptic curves. In [FT10a], we proposed a very simple encoding to all hyperelliptic curves of the form:

$$H\colon y^2 = x^{2g+1} + a_1x^{2g-1} + a_2x^{2g-3} + \cdots + a_gx$$

over finite fields $\mathbb{F}_q$ with $q \equiv 3 \pmod 4$. We called such curves *odd hyperelliptic curves* for obvious reasons. In genus g, they form a family of dimension $g-1$ in the $(2g-1)$-dimensional moduli space of hyperelliptic curves of genus g: for any genus $g \geqslant 3$ it is thus the largest family for which an encoding is known. Another nice feature of this family is that many hyperelliptic curves of cryptographic interest are "odd" in this sense, including those considered in [FKT03, HKT05, Sat09] and more. We give a short description our encoding in the next section, and a more thorough treatment in Chapter 7.

3.5.4 Our contributions

Our works include some contributions to the construction of elliptic and hyperelliptic curve encodings. We present them here.

The simplified SWU encoding. In [BCI+10a], we introduced a simpler, more efficient variant of the SWU encoding, and described an elementary derivation of its formula. It also extends in a straightforward way to the case of Ulas hyperelliptic curves.

To see where it comes from, consider what happens when we try to deduce an encoding function from Theorem 3.3 by fixing the value of u. We know that a product of the form $g(u)g(\tilde{X}_1(t, g(u)))g(\tilde{X}_2(t, g(u)))$ is a square for all t. If $g(u)$ itself is a square, we don't get any interesting information on the values that depend on t, so we cannot construct a non constant encoding. On the other hand, if u is chosen such that $g(u)$ is a quadratic nonresidue, then for all t, either $g(\tilde{X}_1(t, g(u)))$ or $g(\tilde{X}_2(t, g(u)))$ is a square, and thus we get a well-behaved, non constant encoding to $E(\mathbb{F}_q)$. And we can check, moreover, that the same formulas give rise to an encoding even if we replace $g(u)$ by a quadratic nonresidue that isn't in the image of the polynomial g: any choice of a quadratic nonresidue $\xi \in \mathbb{F}_q$ lets us construct an encoding.

Let us see, more formally, how we can derive the formulas for these encodings based on that simple remark. We will consider the hyperelliptic case right away, and try to encode to the set of $\mathbb{F}_q$-points of the curve:

$$H\colon y^2 = g(x) \quad \text{where} \quad g(x) = x^n + ax + b$$

with $n \geqslant 3$ odd and coprime to the characteristic, and $ab \neq 0$.

First observe that, for any $t \in \mathbb{F}_q$ such that $t^n - t \neq 0$, there is a unique $x \in \mathbb{F}_q$ such that $g(tx) = t^n \cdot g(x)$. Indeed, solving this equation for x, we get:

$$\begin{aligned} t^n x^n + atx + b &= t^n x^n + at^n x + bt^n \\ a(t - t^n)x &= b(t^n - 1) \\ x &= -\frac{b}{a} \cdot \frac{t^n - 1}{t^n - t}. \end{aligned}$$

Now write $t = \xi u^2$, where $\xi \in \mathbb{F}_q$ is a fixed quadratic nonresidue, and u is a variable parameter in $\mathbb{F}_q$. Then t^n is a quadratic nonresidue for all u, and hence if x is defined as before (which is possible for all values of u except at most $2n$), either $g(x)$ or $g(tx) = t^n \cdot g(x)$ is a square. This means that either x or tx is the abscissa of a point in $H(\mathbb{F}_q)$, with the ordinate begin $\pm\sqrt{g(x)}$ or $\pm\sqrt{g(tx)}$ accordingly. We thus get the following encoding function:

$$\begin{aligned} f \colon \mathbb{F}_q \setminus S &\longrightarrow H(\mathbb{F}_q) \\ u &\longmapsto \begin{cases} \left(x \ ; \ \sqrt{g(x)}\right) & \text{if } g(x) \text{ is a square} \\ \left(\xi u^2 x \ ; \ -\sqrt{g(\xi u^2 x)}\right) & \text{otherwise} \end{cases} \\ &\text{where} \quad x = -\frac{b}{a} \cdot \frac{\xi^n u^{2n} - 1}{\xi^n u^{2n} - \xi u^2}. \end{aligned}$$

Here, S is the set of cardinality at most $2n$ consisting of all $u \in \mathbb{F}_q$ such that $\xi^n u^{2n} - \xi u^2 = 0$. The minus sign before the second square root ensures that the two "branches" of this encoding function are almost disjoint, making the image somewhat larger.

Note again that this formula is really the same as the one we obtain when deducing an encoding function from Theorem 3.3 by fixing the parameter u, except that the choice of the quadratic nonresidue doesn't have to be in the image of g.

A particularly simple case is when $q \equiv 3 \pmod 4$. Then, we have a very simple nonresidue $\xi = -1$ (and taking square roots is just a matter of raising to the $(q+1)/4$-th power). The resulting simple formula, given in Figure 3.5, is what we call the simplified SWU encoding.

Note also that the same reasoning yields an encoding to hyperelliptic curves of the form:

$$H \colon y^2 = x^n + ax^2 + bx.$$

Simple encodings in characteristic 3. Among the encodings we have described until now, the only one that applies to elliptic curves over fields of characteristic 3 is the original Shallue-van de Woestijne algorithm [SvdW06], which has rather limited efficiency. In [BCI+10a] we have proposed several more efficient constructions.

We only consider elliptic curves of non zero j-invariant over a field $\mathbb{F}_q$ of characteristic 3. They have a reduced Weierstrass equation of the form:

$$E \colon y^2 = x^3 + ax^2 + b \tag{3.16}$$

$$
\begin{aligned}
f\colon \mathbb{F}_q \setminus S &\longrightarrow H(\mathbb{F}_q) \\
u &\longmapsto \begin{cases} \left(x\ ;\ \sqrt{g(x)}\right) & \text{if } g(x) \text{ is a square} \\ \left(-u^2x\ ;\ -\sqrt{g(-u^2x)}\right) & \text{otherwise} \end{cases} \\
&\text{where} \quad x = -\frac{b}{a} \cdot \frac{u^{2n}+1}{u^{2n}-u^2}
\end{aligned}
$$

Here, $S = \{u \in \mathbb{F}_q \mid u^{2n} - u^2 = 0\}$. We can extend f to all of $\mathbb{F}_q$ by setting $f(S) = \{\mathbf{O}\}$, the point at infinity.

Figure 3.5: Simplified SWU encoding to $H\colon y^2 = x^n + ax + b$ over $\mathbb{F}_q$ with $q \equiv 3 \pmod 4$.

and their discriminant is $\Delta = -a^3b$ (hence $ab \neq 0$).

A simple approach is to try and mimic the simplified SWU technique described above. Suppose first that Δ is a square, and write $g(x) = x^3 + ax^2 + b$. For $t \in \mathbb{F}_q \setminus \{0, 1\}$, we can again find $x \in \mathbb{F}_q$ such that $g(tx) = t^3 \cdot g(x)$. Indeed, trying to solve for x we get:

$$
\begin{aligned}
t^3x^3 + at^2x^2 + b &= t^3x^3 + at^3x^2 + bt^3 \\
a(t^2 - t^3)x^2 &= b(t^3 - 1) \\
at^2x^2 &= b(t-1)^2 \\
x^2 &= -\frac{b}{a} \cdot \left(1 - \frac{1}{t}\right)^2
\end{aligned}
$$

but $-b/a = \Delta/a^2$ is a square c^2, so we get a solution $x = c(1 - 1/t)^2$. Then, picking t of the form ξu^2 for some fixed quadratic nonresidue $\xi \in \mathbb{F}_q$ again, we get that one of $g(x)$ and $g(tx)$ must be square. Hence the encoding:

$$
\begin{aligned}
f\colon \mathbb{F}_q^* &\longrightarrow E(\mathbb{F}_q) \\
u &\longmapsto \begin{cases} \left(x\ ;\ \sqrt{g(x)}\right) & \text{if } g(x) \text{ is a square} \\ \left(\xi u^2 x\ ;\ -\sqrt{g(\xi u^2 x)}\right) & \text{otherwise} \end{cases} \\
&\text{where} \quad x = c \cdot \left(1 - \frac{1}{\xi u^2}\right).
\end{aligned}
$$

On the other hand, when Δ is not a square, then the polynomial $g(x)$ has a root $x_0 \in \mathbb{F}_q$ (indeed, if it were irreducible, it would have the non abelian group S_3 as its Galois group, which is not possible over a finite field). Therefore, $g(x - x_0)$ is of the form $x^3 + ax^2 + b'x$, which is amenable to the usual SWU approach. Hence another encoding in this case.

However, as noted by Barreto and Kim in [BK01], we can obtain much more efficient encoding (or in their case, try-and-increment hashing) in characteristic 3 if we can reduce

the problem to solving an Artin-Schreier cubic equation (which is linear over the prime field $\mathbb{F}_3$) instead of taking a square root (which is at least as costly as an exponentiation in $\mathbb{F}_q$). Let us describe how this can be done in our setting.

To put equation (3.16) in Artin-Schreier form (i.e. into a cubic with no term of degree 2), let $w = 1/x$. The equation becomes:

$$\frac{1}{w^3} + \frac{a}{w^2} + (b - y^2) = 0$$

and multiplying by $w^3/(b - y^2)$ to obtain the reciprocal of (3.16), we get:

$$w^3 + \frac{a}{b - y^2} \cdot w = -\frac{1}{b - y^2}. \tag{3.17}$$

This is an Artin-Schreier equation in w; clearly, the map $w \mapsto w^3 + a/(b - y^2) \cdot w$ is linear over $\mathbb{F}_3$, and bijective (hence invertible) if and only if it has a trivial kernel, i.e. if and only if $-a/(b - y^2)$ is not a square. We can ensure that this is the case by fixing a quadratic nonresidue $\xi \in \mathbb{F}_q$, and choosing y such that $b - y^2 = -a\xi z^2$ for some $z \in \mathbb{F}_q$. But $a\xi z^2 + y^2 = b$ is a non degenerate conic over $\mathbb{F}_q$, and hence admits a rational parametrization. For all the values of y given by this parametrization, we can solve equation (3.17) for w and deduce $x = 1/w$. Hence yet another encoding function, which we describe in Algorithm 3.2.

Note that this encoding has some of the drawbacks of the Shallue-van de Woestijne one (e.g. the need to find a point on a conic), but the corresponding performance penalty is easily offset by the efficiency improvement provided by the linear nature of the Artin-Schreier equation.

Algorithm 3.2 Efficient encoding in characteristic 3.

1: **procedure** ArtinSchreierEncoding(u) ▷ encode to $E\colon y^2 = x^3 + ax^2 + b$
2: $z \leftarrow \dfrac{-z_0u^2 + 2y_0u - a\xi z_0}{a\xi - u^2}$ ▷ (z_0, y_0) precomputed solution of $a\xi z^2 + y^2 = b$
3: $y \leftarrow y_0 + u \cdot (z - z_0)$
4: $s \leftarrow 1/(b - y^2)$
5: Find the unique solution w of the Artin-Schreier equation $w^3 + as \cdot w = -s$
6: **return** $(1/w\ ;\ y)$
7: **end procedure**

Encoding to odd hyperelliptic curves. Another contribution, alluded to in the previous section, is the construction of an encoding to so-called odd hyperelliptic curves, of the form:

$$H\colon y^2 = g(x) \quad \text{where } g \text{ is an odd polynomial}$$

over fields $\mathbb{F}_q$ such that $q \equiv 3 \pmod 4$. The idea is really quite simple: since -1 is a quadratic nonresidue, we have that for any $u \in \mathbb{F}_q$, at least one of $g(u)$ and $g(-u) = -g(u)$

$$f\colon \mathbb{F}_q \longrightarrow H(\mathbb{F}_q)$$
$$u \longmapsto \left(\varepsilon(u)\cdot u\ ;\ \varepsilon(u)\sqrt{\varepsilon(u)\cdot g(u)}\right)$$

where $\varepsilon(u) = \chi_q(g(u))$.

Figure 3.6: Encoding to odd hyperelliptic curves $H\colon y^2 = g(x)$ over $\mathbb{F}_q$ with $q \equiv 3 \pmod 4$.

is a square (and indeed, exactly one of them if $g(u) \neq 0$). If we let $\varepsilon(u) = \chi_q(g(u))$, $\varepsilon(u)\cdot u$ is thus always the abscissa of a point in $H(\mathbb{F}_q)$. Hence the encoding function described in Figure 3.6.

This encoding has a number of nice properties: for example, it is "almost" a bijection (more precisely, it induces a bijection between $\mathbb{F}_q$ minus the set of roots of g on the one had, and the non-Weierstrass points in $H(\mathbb{F}_q)$ on the other hand) and has the very simple inverse $(x; y) \mapsto \chi_q(y)\cdot x$. It is also easy to write a constant-time, branch-free and division-free implementation of this encoding, making it possibly the simplest of all encodings with the exception of the Boneh-Franklin supersingular encoding.

As we mentioned earlier, it is also an encoding of some cryptographic interest, since many remarkable hyperelliptic curves are "odd". This includes the genus 2 curves considered by Furukawa et al. [FKT03] and their extension to genus g by Haneda et al. [HKT05], the Type II pairing-friendly curves of genus 2 constructed by Kawazoe and Takahashi [KT08], the genus 2 hyperelliptic curves for which Satoh [Sat09] gave an efficient class group counting algorithm, and more. And even in genus 1, the construction is interesting, since "odd" elliptic curves are the supersingular elliptic curves of Joux [Jou02]: $y^2 = x^3 + ax$.

We present a more thorough discussion of this encoding as well as some constructions based on it in Chapter 7.

3.6 Further Work

3.6.1 Is the problem solved?

We have described a number of constructions of encodings to elliptic and hyperelliptic curves. It is not clear, however, that they solve our initial problem of hashing to elliptic curve groups.

The basic construction of a hash function $\mathfrak{H}\colon \{0,1\}^* \to E(\mathbb{F}_q)$ from an $\mathbb{F}_q$-valued random oracle $\mathfrak{h}\colon \{0,1\}^* \to \mathbb{F}_q$ and an encoding $f\colon \mathbb{F}_q \to E(\mathbb{F}_q)$, as suggested in §3.4, is simply:

$$\mathfrak{H}(m) = f(\mathfrak{h}(m)). \tag{3.18}$$

However, unlike what happens for the Boneh-Franklin encoding, the resulting hash function $\mathfrak{H}$ does not necessarily have strong security properties.

Consider the case when f is Icart's encoding, for example (most other encodings are similar). One can then prove some limited security properties on $\mathfrak{H}$, such as that $\mathfrak{H}$ is one-way if $\mathfrak{h}$ is [Ica09, Lemma 5]. However, unlike the Boneh-Franklin encoding, f is not a surjective or "almost" surjective function to the target group $E(\mathbb{F}_q)$. Indeed, in his original paper [Ica09], Icart could only show that the image $f(\mathbb{F}_q)$ satisfies $\#f(\mathbb{F}_q) \gtrsim (1/4) \cdot \#E(\mathbb{F}_q)$, and conjectured that, in fact,$\#f(\mathbb{F}_q) \approx (5/8) \cdot \#E(\mathbb{F}_q)$ (a conjecture which we will prove in Chapter 4, and extend to other encodings). As a result, the hash function $\mathfrak{H}$ constructed from f using formula (3.18) is easily distinguished from a random oracle!

To see this, note that since f is an algebraic function, we can efficiently compute $f^{-1}(\mathbf{P})$ for any $\mathbf{P} \in E(\mathbb{F}_q)$ by solving a polynomial equation over $\mathbb{F}_q$. In particular, it is possible to decide efficiently whether $\mathbf{P}$ is in the image of f or not. Therefore, we can construct a distinguisher $\mathscr{D}$ between $\mathfrak{H}_0 = \mathfrak{H}$ and a random oracle $\mathfrak{H}_1$ to $E(\mathbb{F}_q)$ as follows. $\mathscr{D}$ is given as input $\mathbf{P} = \mathfrak{H}_b(m) \in E(\mathbb{F}_q)$ for some message m and a random bit $b \in \{0, 1\}$. It answers with a guess of the bit b, as $b = 0$ if $\mathbf{P}$ is in $f(\mathbb{F}_q)$ and $b = 1$ otherwise. Then $\mathscr{D}$ has a constant positive advantage. Indeed, it answers correctly with probability 1 if $\mathbf{P} \notin f(\mathbb{F}_q)$, and with probability $1/2$ otherwise. Hence:

$$\operatorname{Adv} \mathscr{D} \approx (5/8) \cdot (1/2) + (1 - 5/8) \cdot 1 - 1/2 = 3/16.$$

Thus, clearly, construction (3.18) does not behave like a random oracle when f is Icart's encoding (or most other encodings), and cannot replace a random oracle in a generic way.

In many protocols requiring a hash function to an elliptic curve group, this is actually not much of a problem, and an encoding with an image size that is a constant fraction of $\#E(\mathbb{F}_q)$ is often good enough. The reason is that, in a random oracle proof of security, the simulator does want to program the random oracle by setting the hash of some message m to a value $\mathbf{P}$, but that point $\mathbf{P}$ itself can usually be anything depending on some randomness. So the simulator might typically want to set $\mathfrak{H}(m)$ to $\mathbf{P} = [r] \cdot \mathbf{G}$ for some random r, say. Now if $\mathfrak{H}$ is defined in the protocol using a construction like (3.18), the simulator would pick a random r and set $\mathfrak{h}(m)$ to one of the preimages $u \in f^{-1}(\mathbf{P})$ if $\mathbf{P} \in f(\mathbb{F}_q)$. If however $\mathbf{P}$ is not in the image of f, the simulator would pick another random r and try again.

Nevertheless, it seems difficult to give formal sufficient conditions on a protocol for it to remain secure when the elliptic curve-valued random oracle is replaced by a construction like (3.18). One can actually find protocols that are secure in the random oracle model, but in which using that construction instead breaks security completely (we give an example of such a situation in §5.4.2).

Therefore, it would be desirable to obtain from the encodings discussed thus far a construction that does satisfy the *indifferentiability* property mentioned in §3.4, and can thus be used as a plug-in replacement for elliptic curve-valued random oracles in a very large class of protocols. We will see in Chapters 5–6 how we can prove that for the various encodings f described above, the following construction achieves indifferentiability from

a random oracle:

$$\mathfrak{H}(m) = f\big(\mathfrak{h}_1(m)\big) + f\big(\mathfrak{h}_2(m)\big) \tag{3.19}$$

where $\mathfrak{h}_1$ and $\mathfrak{h}_2$ are modeled as random oracles $\{0,1\}^* \to \mathbb{F}_q$ (and the addition is the usual group operation in $E(\mathbb{F}_q)$). As we will show in Chapter 6, a variant of this construction also provides indifferentiable hashing *to the Jacobian* of a hyperelliptic curve even when f is an encoding to the curve itself: this is an important construction in the higher genus case, as a formula like (3.18) alone can never provide secure hashing in that setting[7].

3.6.2 Applications

Constant-time hashing to elliptic curves has already found its way into practical applications and the industry. We discuss some of these applications in a column to *IEEE Security & Privacy* [CT11].

The best-known of them is probably PACE v2 IM [CGIP11], a password-based authenticated key establishment protocol developed by Gemalto and Morpho for e-passports, and standardized as part of the ICAO specifications on Supplemental Access Control for Machine Readable Travel Documents [ICAO10]. This protocol uses a hash function to elliptic curves constructed from Icart's encoding, or alternatively the simplified SWU encoding, using the basic method 3.18.

Bringer, Chabanne and Icart [BCI10b] have also shown how using the indifferentiable hash construction (3.19) instead in a similar password-based key exchange protocol can provide interesting additional anonymity properties for e-passports.

[7] Formula (3.18) does define a hash function to the Jacobian if we fix an embedding of the curve into its Jacobian. However, its image only comprises a negligible proportion of all points on the Jacobian, and thus the re-randomizing trick described in the elliptic case does not apply. More concretely, suppose we were to instantiate BLS signatures over the 160-bit Jacobian of a pairing-friendly hyperelliptic curve of genus 2 using (3.18) as a hash function construction. Then the message hashes would only be 80-bit long, making it possible to find a collision and hence break the scheme in time 2^{40}.

Chapter 4

Estimating the Size of the Image of Constant-Time Encodings

4.1 Introduction

This chapter is devoted to the problem of estimating the size of the image of constant-time encodings. Both Shallue and van de Woestijne [SvdW06] on the one hand, and Icart [Ica09] on the other, gave coarse lower bounds for the image size of their encodings. Obtaining more precise estimates is an interesting mathematical problem, and provides a better understanding of the tightness loss in random oracle security reductions when these encoding functions are used in hash function constructions. A previous version of this work was presented at LATINCRYPT 2010 [FT10b].

4.1.1 Icart's conjecture

To construct a hash function to an elliptic curve group $E(\mathbb{F}_q)$ based on an encoding $f\colon \mathbb{F}_q \to E(\mathbb{F}_q)$ as discussed in §3.4, one would ideally want f to be surjective (possibly up to a negligible fraction of points), or better yet, to satisfy that for a uniformly random $u \in \mathbb{F}_q$, $f(u)$ is "almost" uniformly random (i.e. to be *regular* in the terminology of Chapter 5). The Boneh-Franklin encoding from §3.4.2 satisfies this property, as does the odd hyperelliptic curve encoding we describe in Chapter 7. However, most of the encodings we have described thus far actually factor through a rational map[1] $C(\mathbb{F}_q) \to E(\mathbb{F}_q)$ of degree > 1 for some covering curve C, and such a map is usually *not* surjective. It is interesting to understand how far they fall short of being surjective.

Icart, for example, could only prove that his own encoding f satisfies $\#f(\mathbb{F}_q) \geqslant q/4$. However, he conjectured the following much more precise result, which he left as an open

[1] This is the case for Icart-like encodings. The precise geometric picture is slightly more complicated for SWU-like encodings, where there are typically two covering curves $C_1, C_2 \to E$, and a value $f(u)$ of the encoding f is in the image of one cover or the other depending on the value of the quadratic character $\chi_q(u)$. But the end result is the same: f is usually not surjective.

problem:

$$\left|\#f(\mathbb{F}_q) - \frac{5q}{8}\right| \leqslant \lambda\sqrt{q}$$

for some universal constant λ. It turns out that this result is true: we prove it in this chapter using the Chebotarev density theorem, and show how the methodology naturally adapts to any other encoding by stating and proving the same result for the simplified SWU encoding introduced in §3.5.4, and for the variant of Icart's function over binary fields.

4.1.2 Related work

Independently of our work, Farashahi, Shparlinski and Voloch have obtained a very similar proof of Icart's conjecture [FSV10]. They did not investigate the case of other encoding functions, however.

4.1.3 Outline

In §4.2, we first recall some basic facts about Icart's encoding function, and then give a precise statement of Icart's conjecture. We prove this conjecture in §4.3. Then, we formulate and prove analogues of this conjecture for the binary variant of Icart's encoding (§4.4) and for the simplified SWU encoding (§4.5). Finally, in §4.6, we mention an interesting consequence, already noted by Icart, of the result established herein: namely, the construction of surjective hash functions from Icart's encoding.

4.2 Preliminaries

4.2.1 Icart's encoding

Let $\mathbb{F}_q$ be a finite field of characteristic > 3 and E an elliptic curve over $\mathbb{F}_q$. E can be represented as the union of its neutral element $\mathbf{O}$ and the set of points (x, y) in the affine plane over $\mathbb{F}_q$ such that:

$$y^2 = x^3 + ax + b$$

for some suitable constants $a, b \in \mathbb{F}_q$ satisfying $4a^3 + 27b^2 \neq 0$ (non-singularity).

When $q - 1$ is not divisible by 3, these curves are supersingular for $a = 0$. In all other cases, as discussed in §3.5.2, Icart [Ica09] defines the following function $f\colon \mathbb{F}_q \to E(\mathbb{F}_q)$. He sets $f(0) = \mathbf{O}$ and for all $u \neq 0$, $f(u) = (x, y)$ with:

$$x = \left(v^2 - b - \frac{u^6}{27}\right)^{1/3} + \frac{u^2}{3}$$
$$y = ux + v$$

where $v = (3a - u^4)/(6u)$. This function is shown to be well-defined and easily computed in deterministic polynomial time.

More importantly for our purposes, Icart proves [Ica09, Lemma 3], and it is easy to verify, that there is an algebraic relation between elements $x, y, u \in \mathbb{F}_q$ such that $f(u) = (x, y)$. More precisely, the following holds for all $x, y, u \in \mathbb{F}_q$:

$$f(u) = (x, y) \iff u^4 - 6xu^2 + 6yu - 3a = 0. \tag{4.1}$$

In geometric terms, this equation defines a curve $C \subset \mathbb{P}^1 \times E$ with projections to $\mathbb{P}^1$ and E, and Icart's function is really obtained by composing $C(\mathbb{F}_q) \to E(\mathbb{F}_q)$ with the inverse of $C(\mathbb{F}_q) \to \mathbb{P}^1(\mathbb{F}_q)$ (which is a bijection on $\mathbb{F}_q$-points). See §5.3.2 more a longer discussion of this geometric interpretation.

4.2.2 Icart's conjecture

In [Ica09], Icart conjectures that the image of f contains $(5/8) \cdot \#E(\mathbb{F}_q) + O(q^{1/2})$ points of the curve. In view of relation (4.1), and since the curve itself has $\#E(\mathbb{F}_q) = q + O(q^{1/2})$ points in $\mathbb{F}_q$, this conjecture can be stated as follows.

Conjecture 4.1 (Icart)**.** *Let $\mathbb{K} = \mathbb{F}_q(x, y) = \mathbb{F}_q(x)[y]/(y^2 - x^3 - ax - b)$ be the function field of E, and P the polynomial in $\mathbb{K}[u]$ defined by $P(u) = u^4 - 6xu^2 + 6yu - 3a$. Let further N be the number of points in $E(\mathbb{F}_q)$ at which the reduction of P has a root in $\mathbb{F}_q$. Then*

$$N = \frac{5}{8}q + O(q^{1/2})$$

where the implied constant in the big-O is absolute.

The next section is devoted to the proof of this conjecture.

4.3 Proof of Icart's Conjecture

4.3.1 Genericity of P

Proposition 4.1. *Let again $\mathbb{K} = \mathbb{F}_q(x, y)$ be the function field of E. The polynomial $P(u) = u^4 - 6xu^2 + 6yu - 3a \in \mathbb{K}[u]$ is irreducible over $\mathbb{K}$, and its Galois group is S_4.*

Proof. Introduce the resolvent cubic of P, whose roots in an algebraic closure are $(r_i + r_j)(r_k + r_l)$ for all permutations (i, j, k, l) of $(1, 2, 3, 4)$, with $r_1, \dots, r_4$ the roots of P:[2]

$$\begin{aligned} R(u) &= u^3 + 12xu^2 + (36x^2 + 12a)u + 36y^2 \\ &= u^3 + 12xu^2 + (36x^2 + 12a)u + 36(x^3 + ax + b) \in \mathbb{F}_q(x). \end{aligned}$$

According to classical facts about the quartic equation (see Appendix 4.A), it suffices to prove that P and R are irreducible over $\mathbb{K}$, and that their common discriminant

$$\Delta = -432(9x^6 + 18ax^4 + 90bx^3 - 39a^2x^2 - 54abx + 16a^3 + 81b^2)$$

[2] Some texts use the resolvent whose roots are the $r_i r_j + r_k r_l$. This is of course equivalent, as both sets of roots have the same Galois action.

is not a square in $\mathbb{K}$. Moreover, we can prove these assertions after extending the field of scalars to the algebraic closure $\mathbb{F} = \overline{\mathbb{F}}_q$. Indeed, if they hold over $\mathbb{F}$, they clearly hold *a fortiori* over $\mathbb{F}_q$. The following three lemmas conclude the proof. □

Lemma 4.1. *The resolvent cubic $R(u)$ is irreducible over $\mathbb{F}(x, y)$.*

Proof. This amounts to showing that $R(u)$ has no root in $\mathbb{F}(x, y)$. Note first that it is actually sufficient to prove it has no root in $\mathbb{F}(x)$. Indeed, if it is irreducible in $\mathbb{F}(x)$ but has a root in $\mathbb{F}(x, y)$, the degree of the algebraic extension $\mathbb{F}(x, y)/\mathbb{F}(x)$ must be divisible by $\deg R(u) = 3$. But this extension is quadratic: hence a contradiction.

Let then g/h be a root of R in $\mathbb{F}(x)$, with g and h coprime polynomials. Multiplying the equation $R(g/h) = 0$ by h^3, we get

$$g^3 = h \cdot (- 12xg^2 - (36x^2 + 12a)gh - 36(x^3 + ax + b)h^2)$$

Thus h divides g^3, and since it is coprime to g, it must be constant. Without loss of generality, we thus have $h = 1$ and

$$g^3 + 12xg^2 + (36x^2 + 12a)g + 36(x^3 + ax + b) = 0$$

Let $m = \deg g$. Then the terms in the previous sum are of respective degrees $3m$, $2m + 1$, $m + 2$, 3. If $m \geqslant 2$, the sum is thus of degree $3m$, and if $m \leqslant 0$, it is of degree 3: in neither case can it be 0. The only possibility is thus $m = 1$ and $g = \alpha x + \beta$. We get

$$(\alpha^3 + 12\alpha^2 + 36\alpha + 36)x^3 + 3\beta(\alpha^2 + 8\alpha + 12)x^2 +$$
$$(3\alpha\beta^2 + 12a\alpha + 12\beta^2 + 36a)x + (\beta^3 + 12a\beta + 36b) = 0$$

in $\mathbb{F}(x)$. Suppose $\beta \neq 0$. Since the coefficients of x^3 and x^2 must be zero, this gives $\alpha^3 + 12\alpha^2 + 36\alpha + 36 = \alpha^2 + 8\alpha + 12 = 0$, which is impossible, since the polynomials $X^3 + 12X^2 + 36X + 36$ and $X^2 + 8X + 12$ are coprime. Hence $\beta = 0$, and thus $\alpha^3 + 12\alpha^2 + 36\alpha + 36 = 12a(\alpha + 3) = 0$, which is similarly seen to be impossible (as $a \neq 0$). This completes the proof. □

Lemma 4.2. *The discriminant Δ is not a square in $\mathbb{F}(x, y)$.*

Proof. Again, we will show that it is sufficient to prove that Δ is not a square in $\mathbb{F}(x)$. Indeed, suppose that Δ is not a square in $\mathbb{F}(x)$ but becomes a square in $\mathbb{F}(x, y)$. Since the extension is quadratic, this gives $\mathbb{F}(x, y) = \mathbb{F}(x, \sqrt{\Delta})$. In particular, if λ is a root of $X^3 + aX + b$ in $\mathbb{F}$, the extension $\mathbb{F}(x, \sqrt{\Delta})/\mathbb{F}(x)$ must be ramified at $(x - \lambda)$. In other words, if we specialize $\Delta(x)$ at $x = \lambda$, we must get 0. But

$$\begin{aligned}(\lambda - 3b/a)\Delta(\lambda) &= 16 \cdot 432(\lambda - 3b/a)(3a^2\lambda^2 + 9ab\lambda - a^3) \\ &= 16 \cdot 432\big[3a^2(\lambda^3 + a\lambda + b) - (4a^3 + 27b^2)\lambda\big] \\ &= -16 \cdot 432(4a^3 + 27b^2)\lambda \neq 0\end{aligned}$$

since the characteristic does not divide 6 and $a \neq 0$. Hence a contradiction.

It remains to prove that Δ is not a square in $\mathbb{F}(x)$, or equivalently in $\mathbb{F}[x]$ (since $\mathbb{F}[x]$ is integrally closed). A square root of Δ in $\mathbb{F}[x]$ must have the form $S = \sqrt{-432} \cdot (3x^3 + rx^2 + sx + t)$. The coefficient of x^5 in S^2 must be 0, hence $r = 0$. The coefficient of x^4 must be $18a$, hence $s = 3a$. But then the coefficient of x^2 is equal to both $9a^2$ and $-39a^2$, which is a contradiction since $48a^2 \neq 0$. This completes the proof. □

Lemma 4.3. *The polynomial P is irreducible over $\mathbb{F}(x, y)$.*

Proof. Let σ be the non trivial Galois automorphism of the extension $\mathbb{F}(x, y)/\mathbb{F}(x)$ ($\sigma(y) = -y$). If $P(u)$ decomposes as a product of non constant factors in $\mathbb{F}(x, y)[u]$, then its norm $P_0(u) = P(u)P(u)^\sigma$ is reducible over $\mathbb{F}(x)$. We will show that this is not the case. Note first that $P_0(u)$ can be written as $Q_0(u^2)$, where

$$Q_0(v) = v^4 - 12xv^3 + (36x^2 - 6a)v^2 - 36(x^3 + b)v + 9a^2$$

Now $Q_0(v)$ is easily seen to be an irreducible polynomial of $\mathbb{F}(x)[v]$. Indeed, if it had a root $g/h \in \mathbb{F}(x)$, the rational function g/h would be constant, which is clearly impossible. And if it decomposes as a product of degree 2 factors $Q_0 = (v^2 + rv + s)(v^2 + r'v + s')$, these factors are in $\mathbb{F}[x]$ (integrally closed domain). Since $ss' = 9a^2$, both s and s' are constant. Then, since the coefficient of v^2, $rr' + s + s'$, is of degree 2, r and r' are both of degree at most 2. But then so is $rs' + r's$, which is the coefficient of v in Q_0, namely $-36(x^3 + b)$, hence a contradiction.

Now let w be a root of P_0 in the separable closure of $\mathbb{F}(x)$, and let $\mathbb{L} = \mathbb{F}(x, w)$, $\mathbb{L}' = \mathbb{F}(x, w^2)$. $\mathbb{L}'$ is a subfield of $\mathbb{L}$, and a rupture field of Q_0. In particular $[\mathbb{L} : \mathbb{F}(x)] = [\mathbb{L} : \mathbb{L}'] \cdot [\mathbb{L}' : \mathbb{F}(x)] = 4[\mathbb{L} : \mathbb{L}']$. Since the polynomial P_0 is even, $-w$ is another root of P_0. As $w \notin \mathbb{F}(x)$, $w \mapsto -w$ defines a non trivial $\mathbb{F}(x)$-automorphism of $\mathbb{L}$. This automorphism fixes $\mathbb{L}'$, so $[\mathbb{L} : \mathbb{L}'] \geqslant 2$. This gives $[\mathbb{L} : \mathbb{F}(x)] \geqslant 8$, and thus P_0 must have an irreducible factor of degree $\geqslant 8$. In other words, P_0 is irreducible over $\mathbb{F}(x)$ as required. □

4.3.2 Applying Chebotarev

Now that Proposition 4.1 is established, Conjecture 4.1 readily follows from effective versions of the Chebotarev Density Theorem for function fields. One such version is [FJ05, Proposition 6.4.8], from which one can easily deduce:

Theorem 4.1 (Chebotarev)**.** *Let $\mathbb{K}$ be an extension of $\mathbb{F}_q(x)$ of degree $d < \infty$ and $\mathbb{L}$ a Galois extension of $\mathbb{K}$ of degree $m < \infty$. Assume $\mathbb{F}_q$ is algebraically closed in $\mathbb{L}$, and fix some subset $\mathscr{S}$ of $\mathrm{Gal}(\mathbb{L}/\mathbb{K})$ stable under conjugation. Let $s = \#\mathscr{S}$ and $N(\mathscr{S})$ the number of places v of $\mathbb{K}$ of degree 1, unramified in $\mathbb{L}$, such that the Artin symbol $\left(\frac{\mathbb{L}/\mathbb{K}}{v}\right)$ (defined up to conjugation) is in $\mathscr{S}$. Then*

$$\left| N(\mathscr{S}) - \frac{s}{m} q \right| \leqslant \frac{2s}{m} \left((m + g_{\mathbb{L}}) \cdot q^{1/2} + m(2g_{\mathbb{K}} + 1) \cdot q^{1/4} + g_{\mathbb{L}} + dm \right)$$

where $g_{\mathbb{K}}$ and $g_{\mathbb{L}}$ are the genera of the function fields $\mathbb{K}$ and $\mathbb{L}$.

Proof of Conjecture 4.1. In our case, $\mathbb{K}$ is the function field of E and $\mathbb{L}$ the splitting field of $P(u)$. In particular, $d = 2$, $m = \#S_4 = 24$ and $g_{\mathbb{K}} = 1$. We consider the subset $\mathscr{S} \subset \mathrm{Gal}(\mathbb{L}/\mathbb{K}) = S_4$ consisting of permutations with at least one fixed point—these are the conjugates of (1), (12) and (123), and there are $s = 1 + 6 + 8 = 15$ of them. Hence $s/m = 15/24 = 5/8$.

The places v of $\mathbb{K}$ of degree 1 correspond to points of $E(\mathbb{F}_q)$ (in the projective plane), and for a point $(x_0, y_0) \in E(\mathbb{F}_q)$ not at infinity, saying that $v = (x - x_0)$ has its Artin symbol in $\mathscr{S}$ means that the reduction of $P(u)$ at (x_0, y_0) is a polynomial over $\mathbb{F}_q$ which decomposes into a products of factors at least one of which is of degree 1 (it splits completely if the symbol is (1), decomposes as two linear factors and a quadratic if it is (12) and a product of a linear factor and a cubic if it is (123) up to conjugation).

In other words, $N(\mathscr{S})$ is the same as N in the statement of Conjecture 4.1 up to a constant number accounting for ramified places (at most 12 since Δ is a polynomial of degree 6 in x) and the point at infinity. We then get

$$\left| N - \frac{5}{8}q \right| \leqslant \frac{5}{4}\big((24 + g_{\mathbb{L}}) \cdot q^{1/2} + 72q^{1/4} + g_{\mathbb{L}} + 48\big) + 12 + 1$$

To bound $g_{\mathbb{L}}$, note again that there are at most 12 ramified points, and the ramification index is at most $\deg P_0 = 4$ at each of them. The Riemann-Hurwitz formula thus gives

$$2 - 2g_{\mathbb{L}} \geqslant 24(2 - 2g_{\mathbb{K}}) - 12 \cdot (4 - 1) \quad \text{i.e.} \quad g_{\mathbb{L}} \leqslant 17$$

and thus

$$\left| N - \frac{5}{8}q \right| \leqslant \frac{5}{4}(41q^{1/2} + 72q^{1/4} + 76)$$

In particular, $N = (5/8)q + O(q^{1/2})$. Concretely, for all $q \geqslant 2^{19}$, we have

$$\left| N - \frac{5}{8}q \right| \leqslant 55q^{1/2} \tag{4.2}$$

□

4.4 Analogue in Characteristic 2

As discussed in 3.5.2, Icart also introduced a variant of his function for elliptic curves of non zero j-invariant over finite fields $\mathbb{F}_q$ of even characteristic, i.e. $q = 2^n$. Such an elliptic curve has the form

$$y^2 + xy = x^3 + ax^2 + b$$

with $a, b \in \mathbb{F}_q$, $b \neq 0$. Recall from Figure 3.4 that Icart's function for such a curve E is defined when n is odd as

$$\begin{aligned} f \colon \mathbb{F}_q &\to E(\mathbb{F}_q) \\ u &\mapsto (x, ux + v^2) \end{aligned}$$

where $v = a + u + u^2$ and $x = (v^4 + v^3 + b)^{1/3} + v$. It is shown that $u \in \mathbb{F}_q$ maps to $(x, y) \in E(\mathbb{F}_q)$ if and only if $P(u) = 0$, where $P \in \mathbb{K}[u]$ is defined as

$$P(u) = u^4 + u^2 + xu + (a^2 + y)$$

Using this result, we can prove the following analogue of Icart's conjecture.

Proposition 4.2. *The number of points N in the image of f satisfies:*

$$N = \frac{5}{8}q + O(q^{1/2})$$

where the implied constant in the big-O is universal.

The proof is identical to the one in §4.3.2. The only difference is that the computation of the Galois group is slightly different in even characteristic. However, the group is still S_4. Let us prove this fact now.

Proposition 4.3. *The polynomial $P(u) = u^4 + u^2 + xu + (a^2 + y) \in \mathbb{K}[u]$ is separable and irreducible over $\mathbb{K}$, and its Galois group is S_4.*

Proof. Since $P' = x$ is a unit in $\mathbb{K}[u]$, P is certainly separable. Thus, if we prove that it is irreducible, its Galois group can be determined according to [Con07, Theorem 3.4]: to see that it is S_4, it suffices to show that both its resolvent cubic R and its resolvent *quadratic* Q are irreducible (see Appendix 4.A).

First, we have $R(u) = u^3 + u^2 + x^2$. If this polynomial had a root in $\mathbb{F}_q(x)$, it would be a polynomial of $\mathbb{F}_q[x]$ dividing x^2 by integral closure, which is clearly impossible. Therefore, $R(u)$ is irreducible over $\mathbb{F}_q(x)$, and also over $\mathbb{K}$ by the same degree argument as in the proof of Lemma 4.1: namely, if $R(u)$ had a root in $\mathbb{K}$, $[\mathbb{K} : \mathbb{F}_q(x)] = 2$ would be divisible by $\deg R(u) = 3$, a contradiction.

Additionally, we have $Q(u) = u^2 + x^2u + x^2 + x^4$. If Q is reducible over $\mathbb{F}_q(x)$, we see again that it is split over $\mathbb{F}_q[x]$, and its roots s, t satisfy $\deg s + \deg t = 4$ and $\deg(st) = 2$, hence s and t are quadratics dividing $x^2(1 + x)^2$, i.e. constant multiples of x^2, $x^2 + x$ or $x^2 + 1$, but no such quadratic is a root of Q. Thus, Q is irreducible over $\mathbb{F}_q(x)$. To see that it remains irreducible over $\mathbb{K}$, it suffices to see that $\mathbb{K}$ and the splitting field $\mathbb{F}$ of Q over $\mathbb{F}_q(x)$ are linearly disjoint, and this is certainly the case since $\mathbb{F}$ is a function field of genus 0 whereas $\mathbb{K}$ is of genus 1.

Finally, let us prove that P is irreducible. Let first σ be the non-trivial Galois automorphism of $\mathbb{K}/\mathbb{F}_q(x)$, namely $y \mapsto y + x$, and set $P_0 = PP^\sigma \in \mathbb{F}_q(x)$. It suffices to prove that P_0 is irreducible over $\mathbb{F}_q(x)$. We have

$$P_0 = (u^8 + u^4) + x(u^4 + u^2) + x^2(u^2 + u) + (x^3 + a^2x^2 + a^2x + a^4 + b) = Q_0(u^2 + u)$$

where $Q_0(v) = v^4 + xv^2 + x^2v + (x^3 + a^2x^2 + a^2x + a^4 + b)$.

If Q_0 has a root over $\mathbb{F}_q(x)$, it is in fact in $\mathbb{F}_q[x]$, which is not possible by inspection of the degrees of the four terms in the sum. Similarly, if Q_0 can be written as a product of factors of degree 2, we have $Q_0 = (v^2 + r + s)(v^2 + r + s')$ with r, s and s' are all

in $\mathbb{F}_q[x]$ (with r appearing in both factors by inspection of the degree 3 coefficient of Q_0). We get $\deg(ss') = 3$, so the polynomial $s + s'$ must be of degree at least 2. Since $r(s + s') = x^2$, this implies that r is constant. But then the relation $s + s' + r^2 = x$ gives a contradiction. Therefore Q_0 is irreducible over $\mathbb{F}_q(x)$.

Then, let w be a root of P_0 in the separable closure of $\mathbb{F}_q(x)$, and set $\mathbb{L} = \mathbb{F}_q(x, w)$, $\mathbb{L}' = \mathbb{F}_q(x, w + w^2)$. Like in the proof of Lemma 4.3, we have a tower of extensions $\mathbb{F}_q(x) \subset \mathbb{L}' \subset \mathbb{L}$, and $\mathbb{L}'$ is a rupture field of Q_0, so $[\mathbb{L} : \mathbb{F}_q(x)] = 4[\mathbb{L} : \mathbb{L}']$. Furthermore, since $P_0(u + 1) = P(u)$, $w \mapsto w + 1$ is a non-trivial $\mathbb{L}'$-automorphism of $\mathbb{L}$, which gives $[\mathbb{L} : \mathbb{F}_q(x)] \geqslant 8$ and hence, P_0 is irreducible over $\mathbb{F}_q(x)$, which concludes the proof. □

We can again give concrete bounds. With the notations of §4.3.2, we have $d = 2$, $m = 12$, $s = 8$, $g_{\mathbb{K}} = 1$ and there is exactly one ramified point corresponding to $x = 0$. The Riemann-Hurwitz formula then gives $g_{\mathbb{L}} \leqslant 2$, and thus:

$$\left| N - \frac{3}{4}q \right| \leqslant 21q^{1/2} + 54q^{1/4} + 42$$

In particular, for $q > 2^{16}$ we get

$$\left| N - \frac{3}{4}q \right| \leqslant 25q^{1/2} \tag{4.3}$$

4.5 Analogue for the Simplified Shallue-van de Woestijne-Ulas Encoding

Recall from §3.5.4 the description of the simplified version of the SWU encoding [Ula07] that we gave in [BCI+10a]. We focus here on the case of elliptic curves:

$$E\colon y^2 = x^3 + ax + b$$

with $ab \neq 0$ over fields $\mathbb{F}_q$ with $q \equiv 3 \pmod 4$. Then, -1 is a readily available quadratic nonresidue and we can then use the formula given in Figure 3.5, which simplifies a bit further still as $n = 3$. We finally get the following encoding function:

$$\begin{aligned} f\colon \mathbb{F}_q \setminus \{0, 1, -1\} &\longrightarrow E(\mathbb{F}_q) \\ u &\longmapsto \begin{cases} \left(x\ ;\ \sqrt{g(x)}\right) & \text{if } g(x) \text{ is a square} \\ \left(-u^2 x\ ;\ -\sqrt{g(-u^2 x)}\right) & \text{otherwise} \end{cases} \\ &\text{where} \quad x = -\frac{b}{a} \cdot \left(1 + \frac{1}{u^4 - u^2}\right). \end{aligned}$$

Here, $g(x)$ is again the polynomial $x^3 + ax + b$, and $\sqrt{\cdot}$ denotes the standard square root in $\mathbb{F}_q$, obtained by exponentiation by $(q + 1)/4$.

Using the same notations as in Ulas's Theorem 3.3, define the following rational functions in $\mathbb{F}_q(u)$:

$$X_2(u) = -\frac{b}{a} \cdot \left(1 + \frac{1}{u^4 - u^2}\right) \quad \text{and} \quad X_3(u) = -u^2 \cdot X_2(u).$$

The image of f is then made up of the points in $E(\mathbb{F}_q)$ with an x-coordinate of the form $X_2(u)$ for some u and a *positive*[3] y-coordinate, and of those with an x-coordinate of the form $X_3(u)$ and a *negative*[4] y-coordinate. More formally, $f(\mathbb{F}_q \setminus \{0, 1, -1\})$ is the (almost disjoint) union of the sets I_2 and I_3 defined by

$$I_j = \{(x, y) \in E(\mathbb{F}_q) \mid \exists u \in \mathbb{F}_q,\ x = X_j(u) \text{ and } y = (-1)^j \sqrt{g(x)}\}.$$

Now, disregarding at most three points with zero x-coordinate, I_j consists of half the points on the curve with an x-coordinate of the form $X_j(u)$ for some u. Therefore, if N is the number of points in the image of the algorithm and N_j denotes the number of points with an x-coordinate of the form $X_j(u)$, we get

$$N = \frac{N_2 + N_3}{2} + O(1)$$

and the implied constant is at most 6. We deduce the following result.

Proposition 4.4. *The number of points N in the image of the simplified SWU encoding satisfies:*

$$N = \frac{3}{8}q + O(q^{1/2})$$

where the implied constant in the big-O is universal.

Proof. The proof is again similar to the previous ones. What we actually show is that $N_j = (3/8)q + O(q^{1/2})$ for $j = 2, 3$, using the Chebotarev density theorem again. Note that for all $u \in \mathbb{F}_q \setminus \{-1, 0, 1\}$, we have

$$x = X_2(u) \iff u^4 - u^2 + \frac{1}{\omega} = 0$$
$$x = X_3(u) \iff u^4 - \omega u^2 + \omega = 0$$

where $\omega = \frac{a}{b}x + 1$. Hence, denoting by $\mathbb{K} = \mathbb{F}_q(x, y)$ the function field of E, it suffices to prove that the polynomials $P_2(u) = u^4 - u^2 + 1/\omega$ and $P_3(u) = u^4 - \omega u^2 + \omega$ are irreducible and have Galois group D_8 (the 8-element dihedral group, viewed as a transitive subgroup of S_4) over $\mathbb{K}$. Indeed, D_8 has 8 elements, 3 of which have a fixed point: the same technique as in §4.3.2 then gives the desired estimates for N_2 and N_3.

In view of [KW89, Theorems 2 and 3], a polynomial $P(u) = u^4 - ru^2 + s \in \mathbb{K}[u]$ is irreducible with Galois group D_8 if and only if none of s, $\delta = r^2 - 4s$ or $s\delta$ are

[3] That is, in the image of the square root function $\sqrt{\cdot}$, or equivalently, square.

[4] That is, the opposite of a positive element, i.e. a quadratic nonresidue or zero. Using these definitions, all elements are either positive or negative, except $0 \in \mathbb{F}_q$ which is both.

squares in $\mathbb{K}$. For P_2, we have $(s, \delta, s\delta) = \frac{1}{\omega^2}(\omega, \omega(\omega-4), \omega-4)$, and for P_3, $(s, \delta, s\delta) = (\omega, \omega(\omega-4), \omega^2(\omega-4))$. Thus, all we have to prove is that ω, $\omega-4$ and $\omega(\omega-4)$ are not squares in $\mathbb{K}$. This is obvious in $\mathbb{F}_q(x)$ (since these are polynomials of $\mathbb{F}_q[x]$ which are not square), and extends to $\mathbb{K}$ by a ramification argument as in the proof of Lemma 4.2. □

4.6 Constructing Surjective Hash Functions

In view of the previous results, encodings $\mathbb{F}_q \to E(\mathbb{F}_q)$ are not usually surjective. However, it is possible to use e.g. Icart's encoding to construct simple, efficient surjective hash functions to $E(\mathbb{F}_q)$ as explained in [Ica09, Corollary 2].

Indeed, let f be Icart's function over $\mathbb{F}_q$ (which can be a field of characteristic 2 or > 3) and consider

$$\begin{aligned} F : (\mathbb{F}_q)^2 &\to E(\mathbb{F}_q) \\ (u_1, u_2) &\mapsto f(u_1) + f(u_2) \end{aligned}$$

The following pigeonhole argument shows that F is surjective for large q. Fix a point $P_0 \in E(\mathbb{F}_q)$. Then the sets $S_1 = \{f(u_1) \ / \ u_1 \in \mathbb{F}_q\}$ and $S_2 = \{P_0 - f(u_2) \ / \ u_2 \in \mathbb{F}_q\}$ both consist of $(5/8)q + O(q^{1/2})$ points of E. In particular $\#S_1 + \#S_2 = (5/4)q + O(q^{1/2})$, which is greater than $q + 2\sqrt{q} + 1 \geqslant \#E(\mathbb{F}_q) \geqslant \#S_1 \cup S_2$ for large enough q. Therefore, $S_1 \cap S_2$ is non empty provided that q is large enough, and in that case P_0 is in the image of F.

More precisely, using the explicit bounds (4.2) and (4.3) that we have given for the image size, we obtain that F is surjective as soon as $q > 2^{29}$ in characteristic > 3 (resp. $q > 2^{16}$ in characteristic 2), which is always true in cryptographic applications.

Thus, if we define a hash function $\mathfrak{H}$ as follows:

$$\mathfrak{H}(m) = f\big(\mathfrak{h}_1(m)\big) + f\big(\mathfrak{h}_2(m)\big)$$

where $\mathfrak{h}_1$ and $\mathfrak{h}_2$ are $\mathbb{F}_q$-valued hash functions, then $\mathfrak{H}$ is a surjective hash function to $E(\mathbb{F}_q)$. This is not a very strong statement from a security viewpoint, but we will see in Chapter 5 that, in fact, a much stronger property holds—namely, that $\mathfrak{H}$ is indifferentiable from a random oracle when $\mathfrak{h}_1$ and $\mathfrak{h}_2$ are modeled as random oracles. Hence, $\mathfrak{H}$ can be used as a plug-in replacement for an $E(\mathbb{F}_q)$-valued random oracle in most protocols while preserving random oracle model proofs of security. It is thus a very interesting construction, considering its reasonable level of efficiency.

4.A Galois Groups of Quartics

In this appendix, we recall some classical results regarding the computation of Galois groups of quartic polynomials. The reader is referred to texts like [Cox04, Theorem 13.1.1] and [KW89] for details.

Let $\mathbb{K}$ be any field of odd characteristic, and $P(x) = x^4 + a_1x^3 + a_2x^2 + a_3x + a_4 \in \mathbb{K}[x]$ an irreducible polynomial of degree 4. Let further $\Delta \in \mathbb{K}$ be its discriminant, and

$$R(x) = x^3 - 2a_2x^2 + (a_2^2 + a_1a_3 - 4a_4)x + (a_3^2 + a_1^2a_4 - a_1a_2a_3)$$

its resolvent cubic. Then the Galois group G of P is conjugate to:

- S_4 if R is irreducible and Δ is not a square in $\mathbb{K}$;
- A_4 if R is irreducible and Δ is a square in $\mathbb{K}$;
- $V_4 = \mathbb{Z}/2\mathbb{Z} \times \mathbb{Z}/2\mathbb{Z}$ if R is reducible and Δ is a square in $\mathbb{K}$;
- D_8 or $\mathbb{Z}/4\mathbb{Z}$ otherwise.

If P is a separable polynomial over a field $\mathbb{K}$ of characteristic 2, a similar result holds [Con07, Theorem 3.4], except that the condition that Δ is a square must be replaced by the reducibility over $\mathbb{K}$ of the resolvent quadratic Q of P, defined by

$$Q(x) = x^2 + (a_1^2a_4 + a_1a_2a_3 + a_3^2)x + (a_1^4a_4^2 + a_1^3a_3^3 + a_1^2a_2^3a_4 + a_2^3a_3^2 + a_3^4)$$

Finally, when P is an irreducible biquadratic polynomial (i.e. $a_1 = a_3 = 0$) in odd characteristic, its Galois group can be determined by inspection of its coefficients. It is conjugate to:

- V_4 if a_4 is a square in $\mathbb{K}$;
- $\mathbb{Z}/4\mathbb{Z}$ if $a_4(a_2^2 - 4a_4)$ is a square in $\mathbb{K}$;
- D_8 if neither a_4 nor $a_4(a_2^2 - 4a_4)$ are squares in $\mathbb{K}$.

Chapter 5

Indifferentiable Hashing to Elliptic Curves

5.1 Introduction

In this chapter, we turn to the problem, alluded to in §3.6.1, of constructing hash functions to elliptic curves that are provably well-behaved, in the sense that they can replace an elliptic curve-valued random oracle in a large, explicit class of protocols while preserve proofs of security in the random oracle model. This work was presented at CRYPTO 2010 [BCI+10a].

5.1.1 The random oracle model

Many cryptosystems based on elliptic curves—and all of those that involve the use of a hash function—have been proved secure in the random oracle model (ROM): examples include [BZ04, Bol03, BF01, BGLS03, BLS01, Boy03, BMP00, CC03, GS02, HL02, Jab96, LQ04, ZK02]. In the random oracle model [BR93], a hash function is modeled by a publicly accessible random function (the random oracle); the adversary cannot compute the hash function by himself but must instead query the random oracle.

A proof in the random oracle model is not fully satisfactory, since it does not imply that the scheme will remain secure when the random oracle is replaced by a concrete cryptographic hash function. It is possible to construct ad hoc schemes that are provably secure in the ROM but completely insecure when the random oracle is instantiated with any function family (see [CGH04]). Despite these separation results, a proof in the ROM is believed to indicate that there are no structural flaws in the design of the system, and that no flaw will suddenly appear when a "well-designed" hash function is used instead.

Owing to a large amount of research in the area, we have a good idea of what a well-designed bit string-valued hash function looks like, and it is relatively easy to construct from them "well-behaved" hash functions with values in such algebraic objects as finite fields $\mathbb{F}_q$, finite rings $\mathbb{Z}_N$, and their groups of invertible elements. However, as we mentioned in Chapter 3, it is more difficult to obtain well-behaved hash functions

to elliptic curves even from a bit string-valued random oracle, especially if we want constructions with constant running time.

5.1.2 Constructing good hash functions from elliptic curve encodings

Given a "well-behaved" hash function $\mathfrak{h}\colon \{0,1\}^* \to \mathbb{F}_q$ and an elliptic curve encoding $f\colon \mathbb{F}_q \to E(\mathbb{F}_q)$, the simplest way to construct a hash function $\mathfrak{H}\colon \{0,1\}^* \to E(\mathbb{F}_q)$ is to set:

$$\mathfrak{H}(m) = f(\mathfrak{h}(m)).$$

However, even if $\mathfrak{h}$ is modeled as a random oracle, this construction does not usually behave like a random oracle to $E(\mathbb{F}_q)$. Indeed, except for a very limited number of special curves, known encoding functions f have an image $f(\mathbb{F}_q)$ consisting of a constant fraction < 1 of all points in $E(\mathbb{F}_q)$ (this fraction is $\approx 5/8$ for Icart's encoding, for example, as seen in Chapter 4). This implies that, unlike a random oracle, $\mathfrak{H}$ isn't surjective at all, and this property, together with the fact that encodings are efficiently samplable, is enough to construct an efficient distinguisher between $\mathfrak{H}$ and a random oracle (see §3.6.1).

As a result, current proofs in the random oracle model for $\mathfrak{H}$ do not guarantee the security of the resulting scheme when $\mathfrak{H}$ is instantiated as $\mathfrak{H}(m) = f(\mathfrak{h}(m))$, even if $\mathfrak{h}$ itself assumed to be ideal. In other words, even if a proof in the random oracle for $\mathfrak{H}$ can indicate that there are no structural flaws in the design of the cryptosystem, instantiating $\mathfrak{H}$ in that way could introduce a flaw that would make the resulting cryptosystem completely insecure. We give an example of such a cryptosystem in §5.4.2.

5.1.3 Our goal

Our main goal is to give a generic construction of an elliptic curve-valued hash function $\mathfrak{H}\colon \{0,1\}^* \to E(\mathbb{F}_q)$, based on an elliptic curve encoding f and random oracles $\mathfrak{h}$ for which well-behaved instantiations are known, and that does preserve random oracle proofs of security. To that end, we use the indifferentiability framework of Maurer et al. [MRH04], and try to obtain a hash function construction which is *indifferentiable* from a random oracle. General composition results then ensure that the construction can be used as a plug-in replacement for a random oracle in a large class of cryptographic protocols (namely protocols with single-stage security games, as clarified recently by Ristenpart et al. [RSS11]) while preserving security proofs.

5.1.4 Our results

We first introduce and formalize the notion of *admissible encoding* in §5.2 (which generalizes a similar definition from Boneh and Franklin [BF01]). Being an admissible encoding is a sufficient condition on a family of deterministic algorithms F for the construction:

$$\mathfrak{H}(m) = F(\mathfrak{h}(m))$$

to be indifferentiable from a random oracle when $\mathfrak{h}$ is modeled as a random oracle.

As pointed out earlier, most of the known encoding functions f to elliptic curves cannot give rise to a secure hash function using a construction of that form, so they cannot be admissible encodings themselves. However, we introduce two constructions using them that are indeed indifferentiable.

In §5.3, we prove that if $f\colon \mathbb{F}_q \to E(\mathbb{F}_q)$ is Icart's encoding (both in odd and even characteristic), then the following defines an admissible encoding:

$$F(u, v) = f(u) + f(v)$$

where $+$ denotes the elliptic curve group law. As a result, we obtain a hash function construction which is indifferentiable from a random oracle to $E(\mathbb{F}_q)$:

$$\mathfrak{H}(m) = f\big(\mathfrak{h}_1(m)\big) + f\big(\mathfrak{h}_2(m)\big)$$

when $\mathfrak{h}_1, \mathfrak{h}_2$ are modeled as random oracle to $\mathbb{F}_q$.

The proof relies on a rather careful inspection of the geometric properties of Icart's encoding, but the intuition behind it is in fact quite simple. The main step is to prove that all points $\mathbf{P}$ on the elliptic curve up to a bounded number of exceptions has roughly the same number of preimages under F. But this is true because, since f is an algebraic function, the solutions (u, v) of $f(u) + f(v) = \mathbf{P}$ generically form an irreducible curve of bounded genus in the affine plane over $\mathbb{F}_q$, which consists of $q + O(q^{1/2})$ points by the Hasse-Weil bound.

However, there are some difficulties to overcome to make this idea work, which involve somewhat technical tools from algebraic geometry. This makes the proof difficult to adapt to other encodings with a more complicated geometric description, such as the SWU encoding, and especially encodings to the Jacobians of higher genus curves. Chapter 6 describes a less intuitive but technically much simpler and versatile proof strategy, using arithmetic tools like character sums instead of geometry.

In §5.4, we also describe a different hash function construction for which indifferentiability is very easy to establish. For a very large class of encodings $f\colon \mathbb{F}_q \to E(\mathbb{F}_q)$ which we call *weak encodings* (essentially those whose image size is at least a positive constant fraction of all curve points, which includes all the known elliptic curve encodings described in Chapter 3), the following construction is indifferentiable from a random oracle:

$$\mathfrak{H}(m) = f\big(\mathfrak{h}_1(m)\big) + [\mathfrak{h}_2(m)] \cdot \mathbf{G}.$$

Here, $E(\mathbb{F}_q)$ is assumed to be a cyclic group with generator $\mathbf{G}$, $\mathfrak{h}_1$ is modeled as a random oracle to $\mathbb{F}_q$, and $\mathfrak{h}_2$ as a random oracle to $\mathbb{Z}/N\mathbb{Z}$ with $N = \#E(\mathbb{F}_q)$. This construction is very general and has a very simple security proof, but it is less efficient than the previous one due to the full size scalar multiplication in the elliptic curve group.

Further contributions include a number of composition lemmas for admissible encodings, which we use in §5.5 to construct indifferentiable hash functions to subgroups of elliptic curve groups, or to obtain indifferentiable hashing from bit string-valued random oracles rather than $\mathbb{F}_q$-valued ones.

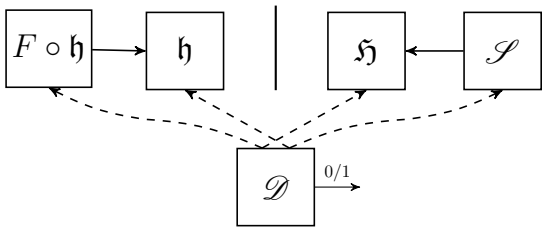


Figure 5.1: The indifferentiability notion, illustrated with construction $C^{\mathfrak{h}} = F \circ \mathfrak{h}$ for some function F, and random oracles $\mathfrak{h}$ and $\mathfrak{H}$.

5.2 Admissible Encodings and Indifferentiability

5.2.1 Preliminaries

Indifferentiability. We recall the notion of indifferentiability introduced by Maurer et al. in [MRH04]. We define an *ideal primitive* as an algorithmic entity which receives inputs from one of the parties and delivers its output immediately to the querying party. A *random oracle* [BR93] to a finite set S is an example of ideal primitive, which provides a random output in S for each new query while identical input queries are given the same answer.

Definition 5.1 (Indifferentiability [MRH04])**.** A Turing machine C with oracle access to an ideal primitive h is said to be $(t_{\mathscr{D}}, t_{\mathscr{S}}, q_{\mathscr{D}}, \varepsilon)$-*indifferentiable* from an ideal primitive H if there exists a simulator $\mathscr{S}$ with oracle access to H and running in time at most $t_{\mathscr{S}}$, such that for any distinguisher $\mathscr{D}$ running in time at most $t_{\mathscr{D}}$ and making at most $q_{\mathscr{D}}$ queries, it holds that:

$$\left|\Pr\left[\mathscr{D}^{C^h,h} = 1\right] - \Pr\left[\mathscr{D}^{H,\mathscr{S}^H} = 1\right]\right| < \varepsilon$$

C^h is said to be indifferentiable from H if ε is a negligible function of the security parameter k, for polynomially bounded $q_{\mathscr{D}}$, $t_{\mathscr{D}}$ and $t_{\mathscr{S}}$.

It is shown in [MRH04] that the indifferentiability notion is a "good" notion for substituting one ideal primitive by a construction based on another ideal primitive. That is, if the construction C^h is indifferentiable from an ideal primitive H, then C^h can replace H in "most" cryptosystems (at least those with single-stage security games), and the resulting cryptosystem is at least as secure in the h model as in the H model; see [MRH04] and [RSS11] for a precise discussion.

Statistical indistinguishability. We also recall the definition of statistically indistinguishable distributions.

Definition 5.2. Let X and Y be two random variables over a set S. The distributions of X and Y are ε-statistically indistinguishable if:

$$\sum_{s \in S} \left| \Pr[X = s] - \Pr[Y = s] \right| \leqslant \varepsilon.$$

The two distributions are statistically indistinguishable if ε is a negligible function of the security parameter.

5.2.2 Admissible encodings

Our goal is to construct a hash function into elliptic curves that is indifferentiable from a random oracle. First, we introduce our new notion of *admissible encoding*. It can be seen as a generalization of the definition used by Boneh and Franklin in [BF01].

Definition 5.3 (Admissible Encoding)**.** A function $F \colon S \to R$ between finite sets is an ε-admissible encoding if it satisfies the following properties:

Computable: F is computable in deterministic polynomial time.

Regular: for s uniformly distributed in S, the distribution of $F(s)$ is ε-statistically indistinguishable from the uniform distribution in R.

Samplable: there is an efficient randomized algorithm $\mathscr{I} \colon R \to S \cup \{\perp\}$ such that for any $r \in R$, $\mathscr{I}(r)$ induces a distribution that is ε-statistically indistinguishable from the uniform distribution in $F^{-1}(r)$.

F is an admissible encoding if ε is a negligible function of the security parameter.

The following theorem shows that if $F \colon S \to R$ is an admissible encoding, then the hash function $\mathfrak{H} \colon \{0,1\}^* \to R$ with:

$$\mathfrak{H}(m) = F\big(\mathfrak{h}(m)\big)$$

is indifferentiable from a random oracle into R when $\mathfrak{h} \colon \{0,1\}^* \to S$ is seen as a random oracle. This shows that the construction $\mathfrak{H}$ can replace a random oracle into R, and the resulting scheme remains secure in the random oracle model for $\mathfrak{h}$.

Theorem 5.1. *Let $F \colon S \to R$ be an ε-admissible encoding. The construction $\mathfrak{H}(m) = F\big(\mathfrak{h}(m)\big)$ is $(t_{\mathscr{D}}, t_{\mathscr{S}}, q_{\mathscr{D}}, \varepsilon')$-indifferentiable from a random oracle, in the random oracle model for $\mathfrak{h} \colon \{0,1\}^* \to S$, with $\varepsilon' = 4q_{\mathscr{D}} \cdot \varepsilon$ and $t_{\mathscr{S}} = 2q_{\mathscr{D}} \cdot t_{\mathscr{I}}$, where $t_{\mathscr{I}}$ is the maximum running time of the sampling algorithm $\mathscr{I}$ for F.*

Proof. We first describe our simulator in Algorithm 5.1. As illustrated in Figure 5.1, the simulator $\mathscr{S}$ has oracle access to $\mathfrak{H}$, and must answer the random oracle queries to $\mathfrak{h}$ made by the distinguisher $\mathscr{D}$. To answer the same queries with the same values, $\mathscr{S}$ is stateful: it maintains a list L of previously answered queries. It uses the sampling algorithm $\mathscr{I}$ of F to answer new queries.

Algorithm 5.1 Simulator $\mathscr{S}$ such that $(\mathfrak{H}, \mathscr{S}^{\mathfrak{H}})$ is indistinguishable from $(F \circ \mathfrak{h}, \mathfrak{h})$.

```
1: procedure 𝒮(m)                                  ▷ hash m ∈ {0,1}* to S
2:     if L contains a pair of the form (m, s) then   ▷ m was queried previously
3:         return s
4:     end if
5:     r ← ℌ(m)                                    ▷ m is new: query ℌ on it
6:     s ← ℐ(r)
7:     Append (m, s) to L
8:     return r
9: end procedure
```

We must show that the systems $(F \circ \mathfrak{h}, \mathfrak{h})$ and $(\mathfrak{H}, \mathscr{S}^{\mathfrak{H}})$ are indistinguishable. We consider a distinguisher $\mathscr{D}$ making at most $q_{\mathscr{D}}$ queries to one of the systems $(F \circ \mathfrak{h}, \mathfrak{h})$ or $(\mathfrak{H}, \mathscr{S}^{\mathfrak{H}})$ and trying to guess which of the two systems it is.

Without loss of generality, we can assume that the distinguisher makes all queries to $\mathfrak{h}(m)$ (or $\mathscr{S}^{\mathfrak{H}}$) for which there was a query to $(F \circ \mathfrak{h})(m)$ (or $\mathfrak{H}(m)$), and conversely; this gives a total of at most $2q_{\mathscr{D}}$ queries. We can then describe the full interaction between the distinguisher and the system as a sequence of triples:

$$\mathsf{View} = (m_i, s_i, r_i)_{1 \leqslant i \leqslant 2q_{\mathscr{D}}}$$

where $s_i = \mathfrak{h}(m_i)$ or $\mathscr{S}^{\mathfrak{H}}(m_i)$, and $r_i = (F \circ \mathfrak{h})(m_i)$ or $\mathfrak{H}(m_i)$. Without loss of generality we can assume that the m_i's are distinct.

If the system is $(F \circ \mathfrak{h}, \mathfrak{h})$ we have $s_i = \mathfrak{h}(m_i)$ for all i. Therefore the s_i's are uniformly and independently distributed in S. Moreover we have $r_i = (F \circ \mathfrak{h})(m_i) = F(s_i)$ for all i: hence, the distribution of each r_i's is ε-statistically indistinguishable from the uniform distribution in R.

On the other hand, if the system is $(\mathfrak{H}, \mathscr{S}^{\mathfrak{H}})$ we have $r_i = \mathfrak{H}(m_i)$ for all i. Therefore the r_i's are uniformly and independently distributed in R. Moreover we have $s_i = \mathscr{I}(r_i)$ for all i. Hence, by Lemma 5.1 below, the distribution of each s_i is 2ε-statistically indistinguishable from the uniform distribution in S. Moreover, from the definition of $\mathscr{I}$, they always satisfy $F(s_i) = r_i$, unless $s_i = \perp$, which happens with probability at most ε.

Overall, the statistical distance between View in system $(F \circ \mathfrak{h}, \mathfrak{h})$ and View in system $(\mathfrak{H}, \mathscr{S}^{\mathfrak{H}})$ is thus at most $q_{\mathscr{D}} \cdot (\varepsilon + 2\varepsilon + \varepsilon) = 4q_{\mathscr{D}} \cdot \varepsilon$. Considering the running time of the simulator, this concludes the proof. □

Lemma 5.1. *For r uniformly distributed in R, the distribution of $s = \mathscr{I}(r)$ is 2ε-statistically indistinguishable from the uniform distribution in S.*

Proof. We let ω be the sequence of random coins used by $\mathscr{I}$. We have to show that $\delta \leqslant 2\varepsilon$, where the statistical distance δ is given by:

$$\delta = \sum_{s \in S} \left| \Pr_{\omega, r}[\mathscr{I}(r) = s] - \frac{1}{\#S} \right|$$

Given $s \in S$, we have $\mathscr{I}(r) = s$ only if $r = F(s)$. Therefore, $\Pr_{\omega,r}[\mathscr{I}(r) = s] = (1/\#R) \cdot \Pr_{\omega}[\mathscr{I}(F(s)) = s]$. This gives:

$$\delta = \sum_{r \in R} \sum_{s \in F^{-1}(r)} \frac{1}{\#R} \left| \Pr_{\omega}[\mathscr{I}(r) = s] - \frac{\#R}{\#S} \right|. \tag{5.1}$$

Now, since F is an ε-admissible encoding, the statistical distance δ_1 between the uniform distribution in R and the distribution of $F(s)$ for a $s \in S$ uniformly random is at most ε, and we can write δ_1 as:

$$\begin{aligned} \delta_1 &= \sum_{r \in R} \left| \Pr_{s}[F(s) = r] - \frac{1}{\#R} \right| \\ &= \sum_{r \in R} \left| \frac{\#F^{-1}(r)}{\#S} - \frac{1}{\#R} \right| \\ &= \sum_{r \in R} \sum_{s \in F^{-1}(r)} \frac{1}{\#R} \left| \frac{\#R}{\#S} - \frac{1}{\#F^{-1}(r)} \right| \end{aligned} \tag{5.2}$$

Moreover, by definition of admissibility again, for each $r \in R$ the distribution of $\mathscr{I}(r)$ is ε-statistically indistinguishable from the uniform distribution in $F^{-1}(r)$ for all $r \in R$. This gives, for all $r \in R$:

$$\sum_{s \in F^{-1}(r)} \left| \Pr_{\omega}[\mathscr{I}(r) = s] - \frac{1}{\#F^{-1}(r)} \right| \leqslant \varepsilon.$$

Hence, summing over $r \in R$, we get:

$$\delta_2 = \sum_{r \in R} \sum_{s \in F^{-1}(r)} \frac{1}{\#R} \left| \Pr_{\omega}[\mathscr{I}(r) = s] - \frac{1}{\#F^{-1}(r)} \right| \leqslant \varepsilon \tag{5.3}$$

From (5.1), (5.2) and (5.3), we obtain $\delta \leqslant \delta_1 + \delta_2 \leqslant \varepsilon + \varepsilon = 2\varepsilon$ as required. □

5.3 Our Main Construction

Let E be an elliptic curve over a finite field $\mathbb{F}_q$ with $q \equiv 2 \pmod 3$. Let $f\colon \mathbb{F}_q \to E(\mathbb{F}_q)$ denote Icart's encoding [Ica09], whose definition is recalled in §3.5.2. As we have note already, it is easy to see that f itself is *not* an admissible encoding to $E(\mathbb{F}_q)$ the image of f consists of only a fraction of the elliptic curve points. Therefore we cannot use the construction $\mathfrak{H}(m) = f(\mathfrak{h}(m))$ for indifferentiable hashing.

In this section, we describe a different construction with the same level of efficiency that does achieve indifferentiability. We consider the following map $F\colon (\mathbb{F}_q)^2 \to E(\mathbb{F}_q)$:

$$F(u_1, u_2) = f(u_1) + f(u_2) \tag{5.4}$$

which had already been considered by Icart for the construction of surjective hashing (see §4.6). We prove that F is an ε-admissible encoding with $\varepsilon = 2^8 \cdot q^{-1/2}$, and deduce the following theorem.

Theorem 5.2. *If $q > 2^{13}$ is any $2k$-bit prime power congruent to* 2 mod 3 *(even or odd), and if the j-invariant of E is not in $\{0; 2592\}$, then the hash function*

$$\mathfrak{H}(m) = f(\mathfrak{h}_1(m)) + f(\mathfrak{h}_2(m))$$

is $(t_D, t_S, q_D, \varepsilon')$-indifferentiable from a random oracle, where $\varepsilon' = 2^{10} \cdot q_D \cdot 2^{-k}$, in the random oracle model for $\mathfrak{h}_1, \mathfrak{h}_2 : \{0,1\}^* \to \mathbb{F}_q$.

In order to prove that F is an admissible encoding, we write, for any $\mathbf{P} \in E(\mathbb{F}_q)$:

$$N(\mathbf{P}) = \#\{(u,v) \in (\mathbb{F}_q)^2 \mid f(u) + f(v) = \mathbf{P}\} = \#F^{-1}(\mathbf{P}).$$

Then the following holds.

Proposition 5.1. *If q is an odd prime power congruent to* 2 mod 3*, and if the j-invariant of E is not in $\{0; 2592\}$, then for every point $\mathbf{P} \in E(\mathbb{F}_q)$ except at most* 144*, we have*

$$|q - N(\mathbf{P})| \leqslant 2^7 \cdot \sqrt{q}$$

and all the remaining points $\mathbf{P}$ satisfy $N(\mathbf{P}) \leqslant 2^5 \cdot q$.

Sections §5.3.2 and §5.3.3 are devoted to the proof of this proposition. Intuitively, the idea of the proof is to show that, for all points $\mathbf{P} \in E(\mathbb{F}_q)$ except a few exceptional ones, $F^{-1}(\mathbf{P})$ is an irreducible algebraic curve of bounded genus in the affine plane $\mathbb{A}^2$ over $\mathbb{F}_q$. The estimate for the number of points then follows from the Hasse-Weil bound. We also show in §5.3.4 that the result extends to Icart's function f in characteristic 2.

Let us now see how Proposition 5.1 implies the regularity of F, and how we can easily deduce that this encoding is admissible, and hence Theorem 5.2.

5.3.1 Admissibility of $F(u,v) = f(u) + f(v)$

To prove that F is ε-admissible, note first that F is clearly computable in deterministic polynomial time. Thus, it suffices to show that it is ε-regular and ε-samplable. Now, for any $\mathbf{P}$ not among the exceptional points, we have

$$\begin{aligned}\left|\frac{\#F^{-1}(\mathbf{P})}{\#(\mathbb{F}_q)^2} - \frac{1}{\#E(\mathbb{F}_q)}\right| &\leqslant \left|\frac{\#F^{-1}(\mathbf{P})}{\#(\mathbb{F}_q)^2} - \frac{1}{q}\right| + \left|\frac{1}{q} - \frac{1}{\#E(\mathbb{F}_q)}\right| \\ &\leqslant \frac{2^7}{q^{3/2}} + \frac{5}{q^{3/2}} \leqslant \frac{2^7+5}{q^{3/2}}\end{aligned}$$

And on the other hand, for exceptional points $\mathbf{P}$:

$$\left|\frac{\#F^{-1}(\mathbf{P})}{\#(\mathbb{F}_q)^2} - \frac{1}{\#E(\mathbb{F}_q)}\right| \leqslant \frac{2^5}{q}$$

Thus, the statistical distance between the distribution of $F(u,v)$ for uniform (u,v) and the uniform distribution on the curve can be bounded as

$$\sum_{\mathbf{P} \in E(\mathbb{F}_q)} \left|\frac{N(\mathbf{P})}{q^2} - \frac{1}{\#E(\mathbb{F}_q)}\right| \leqslant (q + 2\sqrt{q} + 1) \cdot \frac{2^7+5}{q^{3/2}} + 144 \cdot \frac{2^5}{q} \leqslant \frac{2^8}{q^{1/2}}$$

for $q > 2^{13}$, as required. This proves ε-regularity, with $\varepsilon = 2^8 \cdot q^{-1/2}$.

To see that F is ε-samplable, one can consider the following randomized sampling algorithm, where c is some constant to be determined later. Note that computing the set S in step 3 amounts to solving univariate polynomial equations over $\mathbb{F}_q$, which is easily done in polynomial time using an algorithm such as Berlekamp's [Ber68].

Algorithm 5.2 Sampling algorithm for F.

1: **repeat** $\lceil c \cdot \lg q \rceil$ **times**
2: pick $v \in \mathbb{F}_q$ uniformly at random
3: $S \leftarrow f^{-1}(\mathbf{P} - f(v))$
4: **if** $S = \varnothing$ **then**
5: next iteration
6: **else**
7: pick i uniformly at random in $\{1, 2, 3, 4\}$
8: **if** $i \leqslant \#S$ **then**
9: $u \leftarrow i$-th element of S
10: **return** (u, v)
11: **else**
12: next iteration
13: **end if**
14: **end if**
15: **end repeat**
16: **return** $\perp$

Since the image of f contains roughly $(5/8) \cdot \#E(\mathbb{F}_q)$ points (as proved in Chapter 4), a pigeonhole argument shows that $S \neq \varnothing$ with probability at least $2 \cdot 5/8 - 1 = 1/4$. Furthermore, we always have $\#S \leqslant 4$. Thus, each **repeat** iteration succeeds with probability at least $1/16$. Therefore, if we set $c = \frac{1}{2 \lg(16/15)}$, we find that Algorithm 5.2 succeeds with probability greater than $1 - \varepsilon/2$, and steps 7–13 ensure that it yields the uniform distribution on $F^{-1}(\mathbf{P})$. This concludes the proof that F is an admissible encoding.

5.3.2 Geometric interpretation of Icart's encoding

Icart's function f admits a natural extension to the projective line over $\mathbb{F}_q$ by setting $f(\infty) = \mathbf{O}$, the neutral element of the elliptic curve. Then, consider the graph of f:

$$C = \{(u, \mathbf{P}) \in \mathbb{P}^1 \times E \mid f(u) = \mathbf{P}\}$$

As shown in [Ica09, Lemma 3], C is the closed subscheme of $\mathbb{P}^1 \times E$ defined by

$$u^4 - 6xu^2 + 6yu - 3a = 0 \tag{5.5}$$

In other words, Icart's function is the algebraic correspondence between $\mathbb{P}^1$ and E given by (5.5).

Let j be the j-invariant of E:

$$j = 1728 \cdot \frac{4a^3}{4a^3 + 27b^2} \in \mathbb{F}_q.$$

Save for a few exceptional values of j, we can precisely describe the geometry of C.

Lemma 5.2. *If $j \notin \{0; 2592\}$, the subscheme C is a geometrically integral curve on $\mathbb{P}^1 \times E$ with one triple point at infinity and no other singularity. Its normalization $\widetilde{C}$ is a smooth, geometrically integral curve of genus 7. The natural map $h\colon \widetilde{C} \to E$ is a morphism of degree 4 ramified at 12 distinct finite points of $E(\overline{\mathbb{F}}_q)$, with ramification index 2.*

Proof. As usual with Icart's function, we assume that E is not supersingular, i.e. a and j are non-zero. The subscheme C is then clearly reduced, and was shown in Chapter 4 to be geometrically connected (this is Lemma 4.3). It is thus a geometrically integral curve on the surface $\mathbb{P}^1 \times E$.

Let us determine its singular locus. First consider the affine patch of C given by $u \neq \infty$ and $\mathbf{P} \neq \mathbf{O}$. It can be represented as the algebraic set of points (u, x, y) in affine 3-space satisfying $y^2 = x^3 + ax + b$ as well as equation (5.5). It is smooth at all points where the map

$$(u, x, y) \mapsto (y^2 - (x^3 + ax + b), u^4 - 6xu^2 + 6yu - 3a)$$

is of rank 2. The gradient of this map is

$$(u, x, y) \mapsto \begin{pmatrix} 0 & -3x^2 - a & 2y \\ 4u^3 - 12xu + 6y & -6u^2 & 6u \end{pmatrix}.$$

Since E is smooth, this is of rank 2 at a point of the curve unless:

$$\begin{cases} y^2 = x^3 + ax + b \\ u^4 - 6xu^2 + 6yu - 3a = 0 \\ 4u^3 - 12xu + 6y = 0 \\ \begin{vmatrix} -3x^2 - a & 2y \\ -6u^2 & 6u \end{vmatrix} = 6u(2uy - 3x^2 - a) = 0. \end{cases}$$

Eliminating u, x, y between those four equations, we find $-24a^4 - 162ab^3 = -6a(4a^3 + 27b^2) = 0$, which is impossible. Therefore, there is no singular point in this affine patch.

Turning now to points at infinity, we denote by v the local coordinate $1/u$ on $\mathbb{P}^1$ in a neighborhood of ∞, and let $(z, w) = (1/y, x/y)$ be local coordinates on E in a neighborhood of $\mathbf{O}$. First note that there is no point $(\infty, \mathbf{P}) \in C$ with $\mathbf{P} \neq \mathbf{O}$. Indeed, writing equation (5.5) in terms of (v, x, y):

$$3av^4 - 6yv^3 + 6xv^2 - 1 = 0$$

we see that $v = 0$ is never a root.

Similarly, let us find points of the form $(u, \mathbf{O})$ with $u \neq \infty$. In terms of (u, z, w), equation (5.5) becomes

$$zu^4 - 6wu^2 + 6u - 3az = 0.$$

For $z = w = 0$, we get only one root $u = 0$. Thus, the only point on C of the form $(u, \mathbf{O})$ with $u \neq \infty$ is $(0, \mathbf{O})$, and it is easily seen to be regular: the elliptic curve equation becomes $z = bz^3 + awz^2 + w^3$, and hence the map $\mathbb{A}^3 \to \mathbb{A}^2$ defining C in this patch has the following Jacobian matrix:

$$\begin{pmatrix} 0 & -3bz^2 - 2aw + 1 & -a - 3w^2 \\ 4zu^3 - 12wu + 6 & u^4 - 3a & -6u^2 \end{pmatrix}$$

which is of rank 2 for $u = z = w = 0$ (recall that the characteristic does not divide 6).

Finally, the point $(\infty, \mathbf{O})$ lies on C and is a triple point. Indeed, consider the local ring $\mathscr{O} = \mathbb{F}_q[v, z, w]/(bz^3 + awz^2 - z + w^3)_{(v,z,w)}$ of $\mathbb{P}^1 \times E$ at $(\infty, \mathbf{O})$, and write the local equation of C at this point:

$$3azv^4 - 6v^3 + 6wv^2 - z \in \mathscr{O}.$$

Let $\mathfrak{m} = (v, z, w)$ be the maximal ideal of $\mathscr{O}$. Since $z = bz^3 + awz^2 + w^3 \in \mathfrak{m}^3$, we see that the curve equation belongs to $\mathfrak{m}^3$. It isn't in $\mathfrak{m}^4$, however, so the multiplicity of $(\infty, \mathbf{O})$ is exactly 3 (see [Har77, Exercise V.3.4]).

Thus, the normalization $\widetilde{C}$ of C is a smooth geometrically integral curve, and $\widetilde{C} \to C$ is an isomorphism outside $(\infty, \mathbf{O})$, whereas the fiber over $(\infty, \mathbf{O})$ consists of three points.

Consider now the map $h\colon \widetilde{C} \to E$ deduced from the second projection $C \to E$. Since equation (5.5) is of degree 4, h is a morphism of degree 4 as well. The fiber at the origin $\mathbf{O}$ of E contains 4 points, namely $(0, \infty)$ and the 3 points of $\widetilde{C}$ over the singular point of C. In particular, h is unramified at infinity. The ramification points are thus the finite points (x, y) of E where (5.5) has a multiple root, i.e. where the discriminant Δ vanishes.

Recall that Δ is a polynomial of degree 6 in x. It has 6 simple roots over the algebraic closure of $\mathbb{F}_q$ provided that its own discriminant:

$$\operatorname{disc}(\Delta) = 2^{58} \cdot 3^{42} \cdot (4a^3 + 27b^2)^2 \cdot (-4a^3 + 81b^2)^3$$

is nonzero, i.e. $4a^3 \neq 81b^2$, or $j \neq 2592$. We will assume this in what follows.

None of these roots x corresponds to a point (x, y) on E such that $y = 0$. Indeed, eliminating x between $\Delta = 0$ and $x^3 + ax + b = 0$, we find $a(4a^3 + 27b^2) = 0$, which is impossible. Thus, each root of Δ corresponds to exactly two points of E where h is ramified. Hence the 12 distinct ramification points on $E(\overline{\mathbb{F}}_q)$.

Eliminating u, x, y between equation (5.5) and its first and second derivatives as well as $y^2 = x^3 + ax + b$, we find $12(-4a^3 + 81b^2) = 0$, which contradicts our assumption that $j \neq 2592$. It follows that equation (5.5) cannot have a triple root. Thus, all 12 ramification points of h have ramification index 2. This allows us to compute the genus $g_{\widetilde{C}}$ of $\widetilde{C}$ using the Riemann-Hurwitz formula:

$$2g_{\widetilde{C}} - 2 = 4(2 \cdot 1 - 2) + 12 \quad \text{and hence} \quad g_{\widetilde{C}} = 7$$

as required. □

5.3.3 The square correspondence

In this context, the function $(u, v) \mapsto f(u) + f(v)$ occurring in our hash function construction admits the following description. A point (u, v) in the affine plane $\mathbb{A}^2$, or more generally in $\mathbb{P}^1 \times \mathbb{P}^1$, corresponds to $\mathbf{P}$ on the elliptic curve E if and only if there is some point $(\alpha, \beta) \in \widetilde{C} \times \widetilde{C}$ over (u, v) such that $h(\alpha) + h(\beta) = \mathbf{P}$.

Consider the surface $S = \widetilde{C} \times \widetilde{C}$, and define the following two morphisms. The map $p : S \to \mathbb{P}^1 \times \mathbb{P}^1$ is the square of the first projection, and $s : S \to E$ is obtained by composing $h \times h : S \to E \times E$ with the group law $E \times E \to E$. Then the set of points $(u, v) \in \mathbb{P}^1 \times \mathbb{P}^1$ corresponding to a given $\mathbf{P} \in E$ is exactly $p(s^{-1}(\mathbf{P}))$ (and we can take the intersection with $\mathbb{A}^2$ if we are only interested in affine points). This allows us to give a geometric proof of Proposition 5.1.

Let us first describe the geometry of the fibers $s^{-1}(\mathbf{P})$. Denote by $\mathbf{R}_1, \ldots, \mathbf{R}_{12}$ the 12 geometric points of E over which h is ramified, and let $R = \{\mathbf{R}_i + \mathbf{R}_j\}_{1 \leqslant i,j \leqslant 12} \subset E$. The map s is of rank 1 at (α, β) if and only if h is of rank 1 at at least one of α or β, which is certainly the case when $h(\alpha)$ or $h(\beta)$ is not one the $\mathbf{R}_i$. Therefore, s is smooth of relative dimension 1 over the open subscheme $E_0 = E - R$, and all points in E_0 have smooth curves on S as fibers. The following lemma makes this more precise.

Lemma 5.3. *The fibers of s at all geometric points of E_0 are smooth connected curves on $S_{\overline{\mathbb{F}}_q}$ of genus 49.*

Proof. The morphism $s : S \to E$ is projective, and thus proper, so it admits a Stein factorization $S \to E_1 \to E$, where $s_1 : S \to E_1$ has connected geometric fibers, and $E_1 \to E$ is finite [EGA III.1, 4.3.3]. We will show that, in fact, $E_1 = E$.

Consider $\alpha_i \in \widetilde{C}$, $i = 0, \ldots, 3$, the four points such that $h(\alpha_i) = \mathbf{O}$. Then h factors as $\widetilde{C} \to S \to E_1 \to E$ where the first arrow is given by $\beta \mapsto (\alpha_i, \beta)$ for some fixed $i \in \{0, \ldots, 3\}$. In particular, we have surjective morphisms of curves $\widetilde{C} \to E_1 \to E$. The function field $K(E_1)$ of E_1 is thus an intermediate field of the quartic extension $K(\widetilde{C})/K(E)$. But we know by Proposition 4.1 in Chapter 4 that this quartic extension has a normal closure of Galois group S_4. Thus, it doesn't have any non trivial subextension: we must either have $E_1 = \widetilde{C}$ or $E_1 = E$. In the latter case, we are done. Otherwise, $\widetilde{C} \to E_1$ is the identity map, since $K(\widetilde{C})$ doesn't have any non trivial automorphism over $K(E)$. This implies $s_1(\alpha_i, \alpha_j) = \alpha_j$ for all i, j. But by symmetry, we also have $s_1(\alpha_i, \alpha_j) = \alpha_i$, a contradiction.

Hence, $E_1 = E$ and s has connected geometric fibers. In particular, the fiber $Z = s^{-1}(\mathbf{P})$ at any geometric point $\mathbf{P}$ of E_0 is a smooth connected curve. Let us compute its genus g_Z. To do so, observe that the image of Z under $h \times h : S_{\overline{\mathbb{F}}_q} \to (E \times E)_{\overline{\mathbb{F}}_q}$ is the curve E' of points (ς, τ) such that $\varsigma + \tau = \mathbf{P}$. E' is clearly isomorphic to $E_{\overline{\mathbb{F}}_q}$, and is thus of genus 1. Since $h : \widetilde{C} \to E$ is of degree 4, any given point (ς, τ) on E' has $4^2 = 16$ preimages by $Z \to E'$, except when either ς or τ is one of the 12 ramification points $\mathbf{R}_1, \ldots, \mathbf{R}_{12}$ of h (note that ς and τ cannot be both ramification points since $\mathbf{P}$ is outside R). In this latter case, (ς, τ) has $3 \cdot 4 = 12$ preimages. Thus, $Z \to E'$ is a morphism of degree 16 with $2 \cdot 12 = 24$ ramification points in E', each of ramification

type $(2,1,1,2,1,1,2,1,1,2,1,1)$. Applying the Riemann-Hurwitz formula, we get

$$2g_Z - 2 = 16(2 \cdot 1 - 2) + 24 \cdot 4 \quad \text{and hence} \quad g_Z = 49$$

as stated. □

Consider now a fiber Z of s at some $\mathbb{F}_q$-point $\mathbf{P}$ of E not in R. The previous description says that Z is a smooth geometrically integral curve of genus 49 on S. This gives a precise estimate of the number of $\mathbb{F}_q$-points on Z in view of the Hasse-Weil bound:

$$\left|q + 1 - \#Z(\mathbb{F}_q)\right| \leqslant 98\sqrt{q}.$$

What we are interested in, however, is the number of points in $p(Z)$, or more precisely even, in $p(Z) \cap \mathbb{A}^2$. But those numbers are related in a simple way when Icart's function is well-defined, i.e. $q \equiv 2 \pmod 3$.

Lemma 5.4. *Suppose that $q \equiv 2 \pmod 3$, and let N be the number of $\mathbb{F}_q$-points in $p(Z) \cap \mathbb{A}^2$. Then we have*

$$q - 98\sqrt{q} - 23 \leqslant N \leqslant q + 98\sqrt{q} + 1.$$

Proof. Let D be the closed subscheme of points (α, β) on S such that $p(\alpha, \beta)$ has at least one component at infinity in $\mathbb{P}^1 \times \mathbb{P}^1$. Denote further by U be the complementary open subscheme: $U = p^{-1}(\mathbb{A}^2)$. Then p induces a bijection $U(\mathbb{F}_q) \to \mathbb{A}^2(\mathbb{F}_q) \subset (\mathbb{P}^1 \times \mathbb{P}^1)(\mathbb{F}_q)$. This is a direct consequence of the fact that $t \mapsto t^3$ is a bijection in $\mathbb{F}_q$, as explained in the proof of [Ica09, Lemma 3].

In particular, the $\mathbb{F}_q$-points of $p(Z) \cap \mathbb{A}^2$ are in bijection with $(Z \cap U)(\mathbb{F}_q)$. This immediately yields the upper bound:

$$N \leqslant \#Z(\mathbb{F}_q) \leqslant q + 1 + 98\sqrt{q}.$$

To obtain the lower bound, it suffices to estimate the number of $\mathbb{F}_q$-points on Z outside of U, i.e. on $Z \cap D$. This is certainly bounded above by the number of geometric points on $Z \cap D$, which is itself not greater than the intersection number $Z \cdot D$ when Z and D are regarded as 1-cycles on S.

Let l and m denote the divisor classes on $S = \widetilde{C} \times \widetilde{C}$ of $\widetilde{C} \times \{\alpha\}$ and $\{\alpha\} \times \widetilde{C}$ for an arbitrary α. Then $D \equiv n_\infty(l + m)$, where $\equiv$ denotes numerical equivalence and n_∞ is the number of points on $\widetilde{C}$ mapping to ∞ in $\mathbb{P}^1$. As seen in the proof of Lemma 5.2, we have $n_\infty = 3$ (the three points of $\widetilde{C}$ lying over the singular point of C). Hence

$$Z \cdot D = 3Z \cdot (l + m).$$

On the other hand, the canonical divisor K of S satisfies $K \equiv (2g_{\widetilde{C}} - 2)(l + m) = 12(l + m)$ (see e.g. [Har77, Exercises V.1.5(b) and V.1.9(b)]). Now K appears in the adjunction formula of the intersection theory of surfaces [Har77, Proposition V.1.5]:

$$2g_Z - 2 = Z \cdot K + Z^2 = 4Z \cdot D + Z^2.$$

Since Z is a fiber of s_0, its self-intersection number is zero: any two fibers of s_0 are algebraically equivalent and disjoint. Thus $Z^2 = 0$ and we get

$$Z \cdot D = \frac{1}{4} \cdot (2g_Z - 2) = 24.$$

In particular, $\#(Z \cap D)(\mathbb{F}_q) \leqslant 24$ and hence

$$N \geqslant \#Z(\mathbb{F}_q) - 24 \geqslant q - 98\sqrt{q} - 23$$

which concludes the proof. □

The first part of Proposition 5.1 now follows from the previous propositions: under the hypotheses of that theorem, if $\mathbf{P} \in E(\mathbb{F}_q)$ does not belong to R, then $N(\mathbf{P}) = \#\{(u, v) \in (\mathbb{F}_q)^2 \mid f(u) + f(v) = \mathbf{P}\}$ satisfies

$$\left|q - N(\mathbf{P})\right| \leqslant 98\sqrt{q} + 23 \leqslant 2^7 \cdot \sqrt{q}$$

as required. And obviously, there are at most $12^2 = 144$ points in R.

It remains to bound $N(\mathbf{P})$ for an $\mathbb{F}_q$-point $\mathbf{P} \in R \cap E(\mathbb{F}_q)$. To do so, consider again $Z = s^{-1}(\mathbf{P})$ the fiber at such a point, and $E' \subset E \times E$ the image of Z under $h \times h$ (or equivalently, the fiber of the group law of E at $\mathbf{P}$). The morphism $Z \to E'$ is of degree 16, so each point has at most 16 preimages. Hence

$$N(\mathbf{P}) \leqslant 16 \cdot \#E'(\mathbb{F}_q) \leqslant 16\big(q + 1 + 2\sqrt{q}\big) \leqslant 2^5 \cdot q$$

since $q \geqslant 5$. This concludes the proof of Proposition 5.1.

5.3.4 Generalization to even characteristic

The previous technique carries over to Icart's function in characteristic 2 easily (see Figure 3.4 for the definition of Icart's function in characteristic 2). In this case, if E is the elliptic curve $y^2 + xy = x^3 + ax^2 + b$ over a field $\mathbb{F}_q$ of characteristic 2, the curve $C \subset \mathbb{P}^2 \times E$ defining Icart's correspondence has the equation

$$u^4 + u^2 + xu + y + a^2 = 0.$$

It is smooth except at the point $(\infty, \mathbf{O})$ which blows up into 4 regular points in the normalization $\widetilde{C}$. The second projection $h : \widetilde{C} \to E$ is then of degree 4 and only ramified over the single point $\mathbf{R}$ of E such that $x = 0$, with ramification type $(2, 2)$. In particular, $\widetilde{C}$ is a smooth curve of genus 2.

We can then consider the map $s : S = \widetilde{C} \times \widetilde{C} \to E$ again, and find that the fiber $Z = s^{-1}(\mathbf{P})$ at any point $\mathbf{P} \in E - \{[2] \cdot \mathbf{R}\}$ is a smooth geometrically integral curve. The morphism of Z to its image E' in $E \times E$ is of degree 16 and only ramified over $(\mathbf{R}, \mathbf{P} - \mathbf{R})$ and $(\mathbf{P} - \mathbf{R}, \mathbf{R})$, with ramification type $(2, 2, 2, 2, 2, 2, 2, 2)$. Thus, Z is of genus 9.

Using the notations from the proof of Lemma 5.4, the first projection $p : S \to \mathbb{P}^1 \times \mathbb{P}^1$ still induces a bijection $U(\mathbb{F}_q) \to \mathbb{A}^2(\mathbb{F}_q)$ on $\mathbb{F}_q$-points when $q \equiv 2 \pmod 3$ and $U =$

$p^{-1}(\mathbb{A}^2)$. Moreover, if D denotes the complementary divisor on S, we can compute $Z \cdot D = n_\infty Z \cdot (l+m) = 4Z \cdot (l+m)$, while $K \equiv (2g_{\widetilde{C}} - 2)(l+m) = 2(l+m)$. Thus, the adjunction formula gives $2g_Z - 2 = Z \cdot K + Z^2 = \frac{1}{2} Z \cdot D$. Hence:

$$\#(Z \cap D)(\mathbb{F}_q) \leqslant Z \cdot D = 2 \cdot (2g_Z - 2) = 32.$$

Therefore, for any point $\mathbf{P} \in E(\mathbb{F}_q) - \{[2] \cdot \mathbf{R}\}$, we get

$$q - 4\sqrt{q} - 31 \leqslant N(\mathbf{P}) \leqslant q + 4\sqrt{q} + 1.$$

Furthermore, we still have $N([2] \cdot \mathbf{R}) \leqslant 2^5 \cdot q$ as in the previous section.

Those results show that $F : (u, v) \mapsto f(u) + f(v)$ is still an admissible function when f is Icart's function in characteristic 2.

5.4 A More General Construction

Our construction of §5.3 has the advantage of being simple and efficient as it only requires two evaluations of Icart's function. However, the proof involves somewhat technical tools from algebraic geometry, and it is not so simple to adapt to other encoding functions, such as the SWU encoding.

At the cost of a small performance penalty, however, we describe a more general construction that applies to a large class of encoding functions satisfying a few simple axioms. Those encoding functions include Icart's function, a simpler variant of the SWU function, new deterministic encodings in characteristic 3, etc. We call them *weak encodings*. They are defined as follows.

Definition 5.4 (Weak Encoding)**.** A function $f : S \to R$ between finite sets is said to be an α-weak encoding if it satisfies the following properties:

Computable: f is computable in deterministic polynomial time.

α-bounded: for s uniformly distributed in S, the distribution of $f(s)$ is α-bounded in R, i.e. the inequality $\Pr_s[f(s) = r] \leqslant \alpha/\#R$ holds for any $r \in R$.

Samplable: there is an efficient randomized algorithm $\mathscr{I}$ such that $\mathscr{I}(r)$ induces the uniform distribution in $f^{-1}(r)$ for any $r \in R$. Additionally $\mathscr{I}(r)$ returns $N_r = \#f^{-1}(r)$.

The function f is a *weak encoding* if α is a polynomial function of the security parameter.

The main difference with an admissible encoding is that in the second criterion, the distribution of $f(s)$ is only required to be α-bounded instead of being ε-indistinguishable from the uniform distribution. More precisely this criterion requires that, for all $r \in R$:

$$\Pr_s[f(s) = r] = \frac{\#f^{-1}(r)}{\#S} \leqslant \frac{\alpha}{\#R}. \tag{5.6}$$

In view of this inequality, we see that when $\#R/\#S$ is bounded, any invertible function whose preimages have bounded cardinality is a weak encoding; in particular, this is the case for Icart's function, and indeed all the elliptic curve encodings introduced in Chapter 3.

Lemma 5.5. *Suppose $\#R/\#S$ is bounded, and let $f\colon S \to R$ be a polynomially computable function such that $B = \max_{r\in R} \#f^{-1}(r)$ is bounded. Assume that there exists a polynomial-time algorithm Inv that for any $r \in R$ outputs the set $f^{-1}(r)$. Then f is an α-weak encoding, with $\alpha = B \cdot \#R/\#S$. In particular, Icart's function f is an α-weak encoding from $\mathbb{F}_q$ to $E(\mathbb{F}_q)$ with $\alpha = 4 \cdot \#E(\mathbb{F}_q)/q$.*

Proof. We have $\Pr_s[f(s) = r] = \#f^{-1}(r)/\#S \leqslant B/\#S = \alpha/\#R$ by taking $\alpha = B \cdot \#R/\#S$; therefore, the distribution of $f(s)$ for uniform $s \in S$ is α-bounded in R. Given $\mathsf{Inv}(r) = f^{-1}(r)$, the algorithm $\mathscr{I}(r)$ simply generates a random element in the set $f^{-1}(r)$, and lets $N_r = \#f^{-1}(r)$. □

When the output set of a weak encoding is a group (such as the group of points on an elliptic curve), we can construct an admissible encoding from it as follows.

Theorem 5.3 (Weak → Admissible Encoding)**.** *Let $\mathbb{G}$ be cyclic group of order N noted additively, and let $\mathbf{G}$ be a generator of $\mathbb{G}$. Let $f\colon S \to \mathbb{G}$ be an α-weak encoding. Then the function $F\colon S \times \mathbb{Z}_N \to \mathbb{G}$ defined by $F(s, x) = f(s) + [x] \cdot \mathbf{G}$ is an ε-admissible encoding into $\mathbb{G}$, with $\varepsilon = (1 - 1/\alpha)^t$ for any t polynomial in the security parameter k, and $\varepsilon = 2^{-k}$ for $t = \alpha \cdot k$.*

We prove this theorem in the next section. As a consequence, we get that if $f\colon S \to \mathbb{G}$ is any weak encoding to a cyclic group with generator $\mathbf{G}$, then the hash function $\mathfrak{H}\colon \{0,1\}^* \to \mathbb{G}$ defined by:

$$\mathfrak{H}(m) = f(\mathfrak{h}_1(m)) + [\mathfrak{h}_2(m)] \cdot \mathbf{G}$$

where $\mathfrak{h}_1\colon \{0,1\}^* \to \mathbb{G}$ and $\mathfrak{h}_2\colon \{0,1\}^* \to \mathbb{Z}_N$ are hash functions, is indifferentiable from a random oracle in the random oracle model for $\mathfrak{h}_1$ and $\mathfrak{h}_2$. In particular, this is the case when f is any of the elliptic curve encodings of Chapter 3.

Note also that the construction extends to non-cyclic finite abelian groups in an obvious way. For example, if $\mathbb{G}$ is a non-cyclic elliptic curve group, it is isomorphic to the product of two cyclic groups generated by points $\mathbf{G}_1$ and $\mathbf{G}_2$ of order N_1, N_2. Then the following hash function is indifferentiable from a random oracle:

$$\mathfrak{H}(m) = f(\mathfrak{h}_1(m)) + [\mathfrak{h}_2(m)] \cdot \mathbf{G}_1 + [\mathfrak{h}_3(m)] \cdot \mathbf{G}_2$$

when $\mathfrak{h}_1$, $\mathfrak{h}_2$, $\mathfrak{h}_3$ are modeled as random oracles to $\mathbb{G}$, $\mathbb{Z}_{N_1}$ and $\mathbb{Z}_{N_2}$ respectively.

5.4.1 Proof of Theorem 5.3

We must show that F is an admissible encoding. Efficient computability is trivially satisfied. Regularity is also satisfied since for uniform $(s, x) \in S \times \mathbb{Z}_N$ the distribution of $F(s, x) = f(s) + [x] \cdot \mathbf{G}$ is uniform in $\mathbb{G}$.

It remains to prove that F is samplable. Let $\mathscr{I}_f$ be the sampling algorithm of the α-weak encoding $f\colon S \to \mathbb{G}$ with sampling algorithm $\mathscr{I}_f$. We denote $\mathbb{G}$ as R as before. Observe that when r is uniformly distributed in R, the distribution of $\mathscr{I}_f(r)$ is not necessarily close to uniform. Therefore we first describe in Algorithm 5.3 a new sampling algorithm $\mathscr{I}'_f$ which artificially aborts with some well chosen probability dependent on the input, so as to ensure that $\mathscr{I}'_f(r)$ is uniformly distributed in S (when $\mathscr{I}'_f$ does not abort).

Algorithm 5.3 Sampling algorithm with uniformly random preimages.

1: **procedure** $\mathscr{I}'_f(r)$
2: $\quad(s, N_r) \leftarrow \mathscr{I}_f(r)$ $\quad\triangleright\ s \in f^{-1}(r) \cup \{\perp\}$ and $N_r = \#f^{-1}(r)$
3: $\quad\delta_r = \dfrac{\#R \cdot N_r}{\alpha \cdot \#S}$ $\quad\triangleright\ 0 \leqslant \delta_r \leqslant 1$
4: $\quad$With probability $(1 - \delta_r)$ **return** $\perp$
5: $\quad$Otherwise **return** s
6: **end procedure**

Lemma 5.6. *Algorithm $\mathscr{I}'_f$ aborts with probability at most $1 - 1/\alpha$, and for r uniformly random in R, the distribution of $\mathscr{I}'_f(r)$ under the condition that $\mathscr{I}'_f(r) \neq \perp$ is uniform in S.*

Proof. Given $r \in R$, we have that $\mathscr{I}'_f(r) \neq \perp$ with probability δ_r given by:

$$\delta_r = \frac{\#R \cdot \#f^{-1}(r)}{\alpha \cdot \#S}.$$

Hence:

$$\Pr_r[\mathscr{I}'_f(r) \neq \perp] = \sum_{r \in R} \frac{1}{\#R} \cdot \delta_r = \frac{1}{\alpha \cdot \#S} \sum_{r \in R} \#f^{-1}(r) = \frac{1}{\alpha}.$$

By definition, $\mathscr{I}(r)$ is uniformly distributed in the set $f^{-1}(r)$. As a result, we have for any $r \in R$ and any $s \in f^{-1}(r)$:

$$\Pr[\mathscr{I}'_f(r) = s] = \delta_r \cdot \frac{1}{\#f^{-1}(r)} = \frac{\#R}{\alpha \cdot \#S}.$$

Since $\mathscr{I}'_f(r) = s$ only if $r = f(s)$, this gives for any $s \in S$:

$$\Pr_r[\mathscr{I}'_f(r) = s] = \frac{1}{\alpha \cdot \#S}$$

and finally for any $s \in S$:

$$\Pr_r[\mathscr{I}'_f(r) = s \mid \mathscr{I}'_f(r) \neq \perp] = \frac{\Pr_r[\mathscr{I}'_f(r) = s]}{\Pr_r[\mathscr{I}'_f(r) \neq \perp]} = \frac{1}{\#S}$$

which shows that the distribution of $\mathscr{I}'_f(r)$ conditioned on $\mathscr{I}'_f(r) \neq \perp$ is uniform in S. This concludes the proof. □

We the construct the sampling algorithm $\mathscr{I}_F$ of encoding F as described in Algorithm 5.4, and show that it satisfies the samplability condition in the definition of an admissible encoding.

Algorithm 5.4 Sampling algorithm for F.

```
1: procedure I_F(P)
2:     repeat t times
3:         Randomly choose x ∈ Z_N
4:         s ← I'_f(P − [x] · G)
5:         if s ≠ ⊥ then
6:             return (s, x) ∈ S × Z_N
7:         end if
8:     end repeat
9:     return ⊥
10: end procedure
```

We must show that for any $\mathsf{P} \in \mathbb{G}$, the distribution of (s, x) is statistically close to uniform in $F^{-1}(\mathsf{P})$. From Lemma 5.6 and the uniform distribution of $\mathsf{P} - [x] \cdot \mathsf{G} \in \mathbb{G}$, we have $s = \perp$ at step i with probability at most $1 - 1/\alpha$. Therefore algorithm $\mathscr{I}_F$ eventually outputs $s = \perp$ with probability at most $(1 - 1/\alpha)^t$. Moreover, Lemma 5.6 ensures that, under the condition $s \neq \perp$, the distribution of s in uniform in S, and hence the distribution of (s, x) is uniform in $S \times \mathbb{Z}_N$. Therefore, for any $\mathsf{P} \in \mathbb{G}$ the distribution of $\mathscr{I}_F(\mathsf{P})$ is ε-statistically close to uniform in $F^{-1}(\mathsf{P})$, with $\varepsilon = (1 - 1/\alpha)^t$. For $t = \alpha \cdot k$, we can take $\varepsilon = 2^{-k}$; this concludes the proof of Theorem 5.3.

5.4.2 Discussion

We see that the construction $\mathfrak{H}(m) = f\big(\mathfrak{h}_1(m)\big) + f\big(\mathfrak{h}_2(m)\big)$ of §5.3 requires two evaluations of Icart's encoding f but no scalar multiplication in the elliptic curve. Since the cost of evaluating f is essentially that of a field exponentiation, and in practice field exponentiations are roughly 10 times faster than full-size scalar multiplications in $E(\mathbb{F}_q)$, the construction of §5.3 is approximately 5 times faster than the general construction given in this section.

As discussed in §3.6.1, we note that for a number of existing schemes that are proved secure in the random oracle model into an elliptic curve, it would actually be sufficient to use $\mathfrak{H}(m) = f\big(\mathfrak{h}(m)\big)$ only. This is because for many existing schemes the underlying complexity assumption (such as CDH or DDH) has the random self-reducibility property. So in the security proof one "programs" the random oracle using a random instance generated from the original problem instance. Hence, instead of letting $\mathfrak{H}(m) = \mathsf{P}$ where P is from the random instance, one can adapt the proof by letting $f\big(\mathfrak{h}(m)\big) = \mathsf{P}$. To make sure that $\mathfrak{h}(m)$ is uniformly distributed, one can "replay" the random instance generation depending on the number of solutions to the equation $f(u) = \mathsf{P}$, as we do in the proof of Theorem 5.3.

However it is easy to construct a cryptosystem that is secure in the random oracle model but insecure with $\mathfrak{H}(m) = f(\mathfrak{h}(m))$. Consider for example the following symmetric-key encryption scheme: to encrypt with symmetric key k, generate a random r and compute $c = m + \mathfrak{H}(k, r)$ where the message m is a point on the curve and $\mathfrak{H}$ hashes into the curve; the ciphertext is (c, r). This scheme is semantically secure in the random oracle model for $\mathfrak{H}$, since it is a one-time pad. But the scheme is insecure with $\mathfrak{H}(k, r) = f(\mathfrak{h}(k, r))$ because in that case, $\mathfrak{H}(k, r)$ is not uniformly distributed. Thus, for two messages m_0 and m_1 the attacker has a good advantage in distinguishing between the encryption of m_0 and m_1.[1]

The advantage of the two constructions of §5.3 and 5.4 is that we have a simple criterion to plug them into existing schemes: it suffices that the scheme has a proof in the random oracle model, and a single-stage security game. Whereas with $\mathfrak{H}(m) = f(\mathfrak{h}(m))$ it seems difficult to derive a formal criterion from the previous observations.

5.5 Extensions

In this section we consider three extensions that apply to both hash functions from §5.3 and §5.4. We show how to hash into any prime order subgroup of an elliptic curve (with cyclic or non-cyclic group), how to use hash functions mapping into bit strings (rather than $\mathbb{F}_q$) and how to take advantage of primes $p = 2^\ell - \omega$ where ω is small. These extensions are based on composition lemmas for admissible encodings, established in Appendix 5.A.

5.5.1 Extension to a prime order subgroup

In many applications only a prime order subgroup of $E(\mathbb{F}_q)$ is used, so we show how to adapt the constructions of §5.3 and §5.4 to hashing into a subgroup. Let E be an elliptic curve over $\mathbb{F}_q$ with N points, and let $\mathbb{G}$ be a subgroup of prime order N' and generator $\mathbf{G}$. Let further ℓ be the co-factor, i.e. $N = \ell \cdot N'$. We assume that N' does not divide ℓ (i.e. that $(N')^2$ does not divide N), which is satisfied in practice for key size and efficiency reasons.

To obtain a well-behaved hash function to $\mathbb{G}$, it suffices to use the constructions of §5.3 or §5.4 and multiply the result by the cofactor ℓ. For example, consider the hash function $\mathfrak{H}\colon \{0,1\}^* \to \mathbb{G}$ defined by:

$$\mathfrak{H}(m) = [\ell] \cdot \Big(f(\mathfrak{h}_1(m)) + f(\mathfrak{h}_2(m)) \Big) \tag{5.7}$$

where $\mathfrak{h}_1, \mathfrak{h}_2 : \{0,1\}^* \to \mathbb{F}_q$ are modeled as random oracles and f is Icart's function.

Proposition 5.2. *The hash function $\mathfrak{H}$ defined by (5.7) is $(t_{\mathscr{D}}, t_{\mathscr{S}}, q_{\mathscr{D}}, \varepsilon)$-indifferentiable from a random oracle in the random oracle model for $\mathfrak{h}_1$ and $\mathfrak{h}_2$, with $\varepsilon = 2^{10} \cdot q_{\mathscr{D}} \cdot 2^{-k}$.*

[1] If we take Icart's function for f, this is even worse: given c the attacker can easily determine whether there exists u and v such that $c - m_0 = f(u)$ or $c - m_1 = f(v)$ and if one of the two equations has no solution then the attacker recovers the plaintext without uncertainty (this happens with good probability over r).

Informally, we show that the composition of two admissible encodings remains an (almost) admissible encoding, and that multiplication by a co-factor is an ε-admissible encoding, with $\varepsilon = 0$. This proves that $\mathfrak{H}$ is an indifferentiable hash function. See Appendix 5.A.2 for the proof.

The same result holds for the construction of §5.4. In this case for both cyclic and non-cyclic elliptic curves we simply use $\mathfrak{H}(m) = [\ell] \cdot f(\mathfrak{h}_1(m)) + [\mathfrak{h}_2(m)] \cdot \mathbf{G}$ where $\mathbf{G}$ is a generator of the subgroup.

5.5.2 Extension to bit string-valued random oracles

The constructions in the previous sections are based on hash functions into $\mathbb{F}_q$ or $\mathbb{Z}_N$. However in practice a hash function outputs a fixed length string in $\{0,1\}^\ell$. We can modify our construction as follows. We consider an elliptic curve E over $\mathbb{F}_p$, with p a $2k$-bit prime. We define the hash function $\mathfrak{H}\colon \{0,1\}^* \to E(\mathbb{F}_p)$ with:

$$\mathfrak{H}(m) = f(\mathfrak{h}_1(m) \bmod p) + f(\mathfrak{h}_2(m) \bmod p)$$

where $\mathfrak{h}_1$ and $\mathfrak{h}_2$ are two hash functions from $\{0,1\}^*$ to $\{0,1\}^{3k}$ and f is Icart's function.

Proposition 5.3. *The previous hash function $\mathfrak{H}$ is $(t_{\mathscr{D}}, t_{\mathscr{S}}, q_{\mathscr{D}}, \varepsilon)$-indifferentiable from a random oracle, in the random oracle model for $\mathfrak{h}_1$ and $\mathfrak{h}_2$, with $\varepsilon = 2^{11} \cdot q_{\mathscr{D}} \cdot 2^{-k}$.*

Informally, we first show that reduction modulo p is an admissible encoding from $\{0,1\}^\ell$ to $\mathbb{F}_p$ if $2^\ell \gg p$. Since the composition of two admissible encodings remains an (almost) admissible encoding, this shows that $F(u,v) = f(u \bmod p) + f(v \bmod p)$ is also an admissible encoding into $E(\mathbb{F}_p)$ and therefore $\mathfrak{H}$ is an indifferentiable hash function. Of course, the same result holds for the general construction of §5.4. See Appendix 5.A.3 for the proof.

5.5.3 Extension to primes $p = 2^\ell - \omega$

We show a slightly more efficient construction for primes p of the form $p = 2^\ell - \omega$ for small ω, as used for example in the NIST curves [FIPS186–3]. Let E be an elliptic curve over $\mathbb{F}_p$ for such a prime p. Our construction $\mathfrak{H}\colon \{0,1\}^* \to E(\mathbb{F}_p)$ is as follows:

$$\mathfrak{H}(m) = f(\mathfrak{h}_1(m) \bmod p) + f(\mathfrak{h}_2(m) \bmod p)$$

where, this time, $\mathfrak{h}_1$ and $\mathfrak{h}_2$ are two hash functions from $\{0,1\}^*$ to $\{0,1\}^\ell$ and f is Icart's function. Note that the output size ℓ of h_1 and h_2 is the same as the bit size of p, as opposed to the previous section in which we took $\ell = 3k$ for a $2k$-bit prime p.

Proposition 5.4. *$\mathfrak{H}$ is $(t_{\mathscr{D}}, t_{\mathscr{S}}, q_{\mathscr{D}}, \varepsilon)$-indifferentiable from a random oracle, in the random oracle model for $\mathfrak{h}_1$ and $\mathfrak{h}_2$, with $\varepsilon = q_{\mathscr{D}} \cdot \left(2^{10} \cdot 2^{-\ell/2} + 4\omega \cdot 2^{-\ell}\right)$.*

The proof is similar to the proof of Proposition 5.3, except that since $p \simeq 2^\ell$, reduction modulo p is now a generalized admissible encoding from $\{0,1\}^\ell$ to $\mathbb{F}_p$. See Appendix 5.A.4 for the full proof. The same result holds for the general construction of §5.4, where we can take $\mathfrak{H}(m) := f(\mathfrak{h}_1(m) \mod p) + [h_2(m)] \cdot \mathbf{G}$ with $\mathfrak{h}_1, \mathfrak{h}_2 : \{0,1\}^* \to \{0,1\}^\ell$.

5.A Composition Lemmas

5.A.1 Generalized admissible encodings

The propositions of §5.5.1, §5.5.2 and §5.5.3 fit in a common framework: they assert that some function is admissible, and that we still get indifferentiable hashing when composing them with one of our constructions.

It is not quite correct that composing two admissible encodings yields another admissible encoding. To circumvent this problem, we introduce the slightly more general notion of *generalized admissible encoding*. We show that admissible encodings are generalized admissible encoding; that generalized admissible encodings are sufficient for indifferentiability; and that the composition of two admissible encodings is again a generalized admissible encoding.

Definition 5.5 (Generalized Admissible Encoding)**.** A function $F : S \to R$ is said to be an ε-generalized admissible encoding if it satisfies the following properties:

Computable: F is computable in deterministic polynomial time;

Invertible: there exists a probabilistic polynomial time algorithm $\mathscr{I}_F$ such that $\mathscr{I}_F(r) \in F^{-1}(r) \cup \{\perp\}$ for all $r \in R$, and the distribution of $\mathscr{I}_F(r)$ is ε-statistically indistinguishable from the uniform distribution in S when r is uniformly distributed in R.

F is an generalized admissible encoding if ε is a negligible function of the security parameter.

From Lemma 5.1 we have that an ε-admissible encoding is also a 2ε-generalized admissible encoding. The next lemma says that Definition 5.5 is sufficient for obtaining the indifferentiability property; it follows immediately from our proof of Theorem 5.1.

Lemma 5.7. *Let $F : S \to R$ be an ε-generalized admissible encoding. The construction $H(m) = F(h(m))$ is $(t_D, t_S, q, \varepsilon')$-indifferentiable from a random oracle, in the random oracle model for $h : \{0,1\}^* \to S$, with $\varepsilon' = 2q\varepsilon$ and $t_S = 2q \cdot t_I$, where t_I is the maximum running time of F's sampling algorithm.*

Finally we prove that the composition of two generalized admissible encoding remains a generalized admissible encoding.

Lemma 5.8. *Let $F\colon S \to R$ be an ε_1-generalized admissible encoding and $G\colon R \to T$ be an ε_2-generalized admissible encoding. Then $G \circ F$ is an $(\varepsilon_1 + \varepsilon_2)$-generalized admissible encoding from S to T.*

Proof. Firstly, $G \circ F$ is computable in polynomial time. Secondly, given t uniformly distributed in T, the random variable $r = \mathscr{I}_G(t)$ is ε_2-statistically indistinguishable from the uniform distribution in R. Thus $s = \mathscr{I}_F(r)$ is $(\varepsilon_1 + \varepsilon_2)$-statistically indistinguishable from the uniform distribution in S. □

5.A.2 Proof of Proposition 5.2

Proposition 5.2 is then an immediate consequence of Lemma 5.8 and of the following result.

Lemma 5.9. *The map $M_\ell : E(\mathbb{F}_q) \to \mathbb{G}$, $\mathbf{P} \mapsto [\ell] \cdot \mathbf{P}$ is an ε-admissible encoding, with $\varepsilon = 0$.*

Proof. The proof is the same as the proof of [BF01, Lemma 5.1]. Since M_ℓ is a group homomorphism, the distribution of $M_\ell(\mathbf{P})$ is uniform in $\mathbb{G}$ for uniform $P \in E$.

The group $E(\mathbb{F}_q)$ is isomorphic to $\mathbb{Z}_{\ell_1 N'} \times \mathbb{Z}_{\ell_2}$ for some ℓ_1, ℓ_2 such that $\ell = \ell_1 \ell_2$. Let $\mathbf{G}_1, \mathbf{G}_2$ be the points corresponding to $(1, 0)$ and $(0, 1)$ under this isomorphism. Given $\mathbf{Q} \in \mathbb{G}$, the sampling algorithm picks $u \in \mathbb{Z}_{\ell_1}$ and $v \in \mathbb{Z}_{\ell_2}$ uniformly at random, and returns $\mathbf{P} = [1/\ell] \cdot \mathbf{Q} + [uN'] \cdot \cdot \mathbf{G}_1 + [v] \cdot \mathbf{G}_2$. Here $1/\ell$ is computed in $(\mathbb{Z}_{N'})^*$. We have that $[\ell] \cdot \mathbf{P} = \mathbf{Q}$ as required, and $\mathbf{P}$ is uniformly distributed among the preimages of $\mathbf{Q}$ under M_ℓ. □

5.A.3 Proof of Proposition 5.3

Let p be an integer. We first show that reduction modulo p is an admissible encoding from $\{0,1\}^\ell$ to $\mathbb{Z}_p$ if $2^\ell \gg p$.

Lemma 5.10 ($\{0,1\}^\ell \to \mathbb{Z}_p$)**.** *Let p be an integer and k be a security parameter. Let $\ell = k + \lceil \log_2 p \rceil + 1$. The function $\textsc{Mod}_p : [0, 2^\ell - 1] \to \mathbb{Z}_p$ with $\textsc{Mod}_p(x) = x \bmod p$ is a 2^{-k}-generalized admissible encoding.*

Proof. Let $\mu \in \mathbb{Z}$ such that $2^\ell - p < \mu \cdot p \leqslant 2^\ell$. We consider the sequence:

$$\{0,1\}^\ell \xrightarrow{F} [0, \mu \cdot p[\xrightarrow{G} \mathbb{Z}_p$$

where $F(x) = x \bmod (\mu \cdot p)$ and $G(y) = y \bmod p$. We show that both F and G are generalized admissible encodings; therefore from Lemma 5.8 the composition of F and G remains a generalized admissible encoding. For F we actually prove a slightly more general result.

Lemma 5.11. *Let $F : S \to (S \cup \Delta_2) \setminus \Delta_3$ be a polynomially computable function such that $F(x) = x$ for all $x \in S \setminus \Delta_1$. Assume that set membership for $S \setminus \Delta_1$ can be decided in polynomial time. Then F is an ε-generalized admissible encoding, with $\varepsilon = (\#\Delta_1 + \#\Delta_2 + \#\Delta_3)/\#S$.*

Proof. Given $x \in S \cup \Delta_2$, the sampling algorithm $\mathscr{I}_F(x)$ returns x for $x \in S \setminus \Delta_1$ and $\perp$ otherwise. Therefore for uniform $x \in (S \cup \Delta_2) \setminus \Delta_3$ the distribution of $\mathscr{I}_F(x)$ is ε-indistinguishable from the uniform distribution in S, with $\varepsilon = (\#\Delta_1 + \#\Delta_2 + \#\Delta_3)/\#S$. □

Applying Lemma 5.11 with $S = \{0,1\}^\ell$, $\Delta_1 = [\mu \cdot p, 2^\ell - 1[$, $\Delta_2 = \varnothing$ and $\Delta_3 = \Delta_1$, we obtain that F is an ε-generalized admissible encoding, with $\varepsilon = 2p/2^\ell \leqslant 2^{-k}$. Moreover,

it is easy to see that G is an ε-admissible encoding for $\varepsilon = 0$. From Lemma 5.8 the composition of two generalized admissible encodings remains a generalized admissible encoding. This concludes the proof of Lemma 5.10. □

We now proceed with the proof of Proposition 5.3. With p a $2k$-bit prime and $\ell = 3k$, from Lemma 5.10 we obtain that reduction mod p is a 2^{-k+1}-admissible encoding from $\{0,1\}^\ell$ to $\mathbb{Z}_p$. Using Lemma 5.8, this shows that $F\colon \{0,1\}^\ell \times \{0,1\}^\ell \to E(\mathbb{F}_p)$ with:

$$F(u,v) = f(u \bmod p) + f(v \bmod p)$$

is an ε-generalized admissible encoding with $\varepsilon = 2^9 \cdot 2^{-k} + 2^{-k+1} \leqslant 2^{10} \cdot 2^{-k}$. Applying Lemma 5.7, this proves Proposition 5.3.

5.A.4 Proof of Proposition 5.4

We consider the function $G\colon : \{0,1\}^\ell \to \mathbb{Z}_p$ with $G(u) = u \mod p$. Since $p = 2^\ell - \omega$, applying Lemma 5.11 with $S = \{0,1\}^\ell$, $\Delta_1 = [p, 2^\ell[$, $\Delta_2 = \varnothing$ and $\Delta_3 = \Delta_1$, we obtain that G is an ε-generalized admissible encoding with $\varepsilon = 2\omega \cdot 2^{-\ell}$. Using Lemma 5.8, this shows that $F\colon \{0,1\}^\ell \times \{0,1\}^\ell \to E(\mathbb{F}_p)$ with:

$$F(u,v) = f(u \bmod p) + f(v \bmod p)$$

is an ε-generalized admissible encoding with $\varepsilon = 2^9 \cdot 2^{-\ell/2} + 2\omega \cdot 2^{-\ell}$. Applying Lemma 5.7, this proves Proposition 5.4.

Chapter 6

Well-Distributed Encodings: A Framework for Indifferentiable Hashing to (Hyper)elliptic Curves

6.1 Introduction

6.1.1 Background

In the previous chapter, we considered the problem of constructing "well-behaved" hash functions to elliptic curves based on the constant-time encodings presented in Chapter 3. We mainly proposed two constructions, both shown to be *indifferentiable from a random oracle*, which implies, as we discussed previously, that they can serve as plug-in replacement for random oracles in most cryptosystems while preserving security proofs in the random oracle model.

The first construction is as follows:

$$\mathfrak{H}(m) = f(\mathfrak{h}_1(m)) + f(\mathfrak{h}_2(m))$$

where f is Icart's encoding [Ica09] and $\mathfrak{h}_1$ and $\mathfrak{h}_2$ are modeled as random oracles to $\mathbb{F}_q$. This construction has the advantage of being quite efficient, requiring only two evaluations of f (and hence, essentially two exponentiations in $\mathbb{F}_q$). However, the geometric proof given in §5.3 only applies to Icart's encoding, and seems difficult to generalize to elliptic curve encodings with a more complicated geometric description, such as the SWU encoding [SvdW06, Ula07, BCI+10a], and even more so to a higher genus setting. In fact, even extending the result to an encoding that is very close to Icart's, such as the Hessian curve encoding of Farashahi [Far11], while most likely possible, would require studying the geometric properties of the morphisms involved all over again from scratch.

The second construction, for a cyclic elliptic curve group of order N and generator $\mathbf{G}$, can be written as:

$$\mathfrak{H}(m) = f(\mathfrak{h}_1(m)) + [\mathfrak{h}_2(m)] \cdot \mathbf{G}$$

where f is essentially any of the known elliptic curve encodings, and $\mathfrak{h}_1$ and $\mathfrak{h}_2$ are modeled as random oracles to $\mathbb{F}_q$ and $\mathbb{Z}_N$ respectively. The indifferentiability proof for this construction is much simpler, and it isn't restricted to just Icart's encoding (indeed, it applies to all so-called *weak encodings*, which is a very mild condition). However it is also much less efficient, and still does not readily generalize to hyperelliptic curves.

6.1.2 Our contributions

In this chapter, we introduce a new approach to deal with hash function constructions of the more general form:

$$\mathfrak{H}(m) = f(\mathfrak{h}_1(m)) + \cdots + f(\mathfrak{h}_s(m)) \tag{6.1}$$

when f is any of the known deterministic encodings. We can show in particular that this construction is well-behaved (indifferentiable from a random oracle, in the random oracle model for the $\mathbb{F}_q$-valued hash functions $\mathfrak{h}_i$) as soon as s is greater than the genus of the target curve (that is, $s \geqslant 2$ for elliptic curves, $s \geqslant 3$ for genus 2 curves, etc.). In particular, with this new approach, we recover the results of the previous chapter about Icart's function as a special case, and with sharper bounds. The methodology also extends to all known deterministic encodings, including encodings to hyperelliptic curves.

For that purpose, we introduce the notion of *well-distributed encoding*, based on a new type of character sums associated with characters of the groups of points of the Jacobians of the target curves. We show that these sums can be estimated using classical results of Weil [Wei95] and Bombieri [Bom66], and combine these estimates with standard number theoretic technique in order to get explicit regularity results for functions of the form $(u_1, \ldots, u_s) \mapsto f(u_1) + \cdots + f(u_s)$.

An expanded version of this work is to appear in *Math. Comp.* [FFS+11].

6.1.3 Outline

In §6.2, we introduce the notion of *well-distributed encoding*, and show how it can be used to derive regularity results formally (Theorem 6.1 and Theorem 6.2). Combining these results with those of Chapter 5, we obtain, in essence, that a well-distributed encoding can be used in a construction of the form (6.1) to obtain an indifferentiable hash function.

We then introduce some arithmetic machinery in §6.3 to establish Theorem 6.3. This theorem reduces the problem of proving that a given encoding is well-distributed to checking some relatively simple geometric properties of the morphisms of curves associated with an encoding (which is a lot simpler than the higher-dimensional geometry that the approach of §5.3 required).

And finally, in §6.4, we pick three representative examples of deterministic encodings to elliptic and hyperelliptic curves, apply this theorem to prove that they are well-distributed, and deduce from our general results that they give rise to hash functions indifferentiable from a random oracle. In particular, we obtain indifferentiable hashing to Jacobians of certain hyperelliptic curves.

6.2 Well-Distributed Encodings

6.2.1 Character sums

Consider an encoding f into a curve X, and let J denote the Jacobian of X. Assume that X has an $\mathbb{F}_q$-rational point $\mathbf{O}$, so that we can fix an embedding $X \to J$ (sending a point $\mathbf{P}$ to the degree 0 divisor $(\mathbf{P}) - (\mathbf{O})$). Regularity properties of functions $f^{\otimes s}$ of the form:

$$\begin{aligned} f^{\otimes s} \colon (\mathbb{F}_q)^s &\to J(\mathbb{F}_q) \\ (u_1, \ldots, u_s) &\mapsto f(u_1) + \cdots + f(u_s) \end{aligned}$$

can be derived formally from the behavior of f with respect to characters of $J(\mathbb{F}_q)$.More precisely, introduce the character sums

$$S_f(\chi) = \sum_{u \in \mathbb{F}_q} \chi(f(u)), \tag{6.2}$$

where χ is any character of the abelian group $J(\mathbb{F}_q)$. We say that f is *well-distributed* if we have good bounds on the magnitude of $S_f(\chi)$ for nontrivial characters χ.

Definition 6.1. Let X be a smooth projective curve over a finite field $\mathbb{F}_q$, J its Jacobian, f a function $\mathbb{F}_q \to X(\mathbb{F}_q)$ and B a positive constant. We say that f is B-well-distributed if for any nontrivial character χ of $J(\mathbb{F}_q)$, the following holds:

$$|S_f(\chi)| \leqslant B\sqrt{q}. \tag{6.3}$$

We say that f is well-distributed if it is B-well-distributed for some B bounded independently of the security parameter.

As we show in §6.4, essentially all known deterministic encoding functions into elliptic and hyperelliptic curves satisfy (6.3) and we can thus establish results on the regularity of $f^{\otimes s}$ for any such encoding similar to those obtained in Chapter 5 for $f^{\otimes 2}$ when f is Icart's function.

6.2.2 Collision probability

Besides character sums our estimates also depend on the number of solutions W_f to the equation $f(u) = f(v)$ where $u, v \in \mathbb{F}_q$. We note that

$$\rho_f = W_f/q^2$$

is the probability of a collision. For all functions considered in this chapter we have a bound of the type

$$\rho_f \leqslant \frac{A_0}{q} + \frac{B_0}{q^2} \tag{6.4}$$

with some explicit constants A_0 and B_0 (different for each function f).

6.2.3 Distribution of image sums

Fix a positive integer s, and consider for any element $\mathbf{D} \in J(\mathbb{F}_q)$ the number of tuples $(u_1, \ldots, u_s)$ such that $\mathbf{D} = f(u_1) + \cdots + f(u_s)$:

$$N_s(\mathbf{D}) = \#\{(u_1, \ldots, u_s) \in (\mathbb{F}_q)^s \mid \mathbf{D} = f(u_1) + \cdots + f(u_s)\}$$

We can establish the following result.

Theorem 6.1. *If $f\colon \mathbb{F}_q \to X(\mathbb{F}_q)$ is a B-well-distributed encoding into a curve X, then for all $\mathbf{D} \in J(\mathbb{F}_q)$, we have:*

$$\left| \frac{N_s(\mathbf{D})}{q^s} - \frac{1}{\#J(\mathbb{F}_q)} \right| \leqslant \frac{B^{s-2}}{q^{s/2-1}} \left(\rho_f - \frac{1}{\#J(\mathbb{F}_q)} \right).$$

Proof. Using the orthogonality relation

$$\sum_{\chi} \chi(\mathbf{D}) = \begin{cases} \#J(\mathbb{F}_q) & \text{if } \mathbf{D} \text{ is the zero divisor } J(\mathbb{F}_q) \\ 0 & \text{otherwise,} \end{cases}$$

where the summation is over all characters of $J(\mathbb{F}_q)$, $N_s(\mathbf{D})$ can be expressed in terms of character sums:

$$\begin{aligned} N_s(\mathbf{D}) &= \sum_{u_1,\ldots,u_s \in \mathbb{F}_q} \frac{1}{\#J(\mathbb{F}_q)} \sum_{\chi} \chi\left(f(u_1) + \cdots + f(u_s) - \mathbf{D}\right) \\ &= \sum_{\chi} \frac{\chi(-\mathbf{D})}{\#J(\mathbb{F}_q)} \sum_{u_1,\ldots,u_s \in \mathbb{F}_q} \chi\left(f(u_1) + \cdots + f(u_s)\right) \\ &= \sum_{\chi} \frac{\chi(-\mathbf{D})}{\#J(\mathbb{F}_q)} \left(S_f(\chi)\right)^s. \end{aligned}$$

Putting aside the contribution of the trivial character χ_0, we get:

$$N_s(\mathbf{D}) - \frac{q^s}{\#J(\mathbb{F}_q)} = \frac{1}{\#J(\mathbb{F}_q)} \sum_{\chi \neq \chi_0} \chi(-\mathbf{D}) \left(S_f(\chi)\right)^s.$$

Therefore, using (6.3), we derive

$$\left| N_s(\mathbf{D}) - \frac{q^s}{\#J(\mathbb{F}_q)} \right| \leqslant \frac{1}{\#J(\mathbb{F}_q)} \sum_{\chi \neq \chi_0} |S_f(\chi)|^s \leqslant \frac{(B\sqrt{q})^{s-2}}{\#J(\mathbb{F}_q)} \sum_{\chi \neq \chi_0} |S_f(\chi)|^2.$$

On the other hand, W_f, defined in §6.2.2, is just:

$$W_f = \#\{(u_1, u_2) \in (\mathbb{F}_q)^2 \mid f(u_1) - f(u_2) = 0\} = \sum_{u_1,u_2 \in \mathbb{F}_q} \frac{1}{\#J(\mathbb{F}_q)} \sum_{\chi} \chi\left(f(u_1) - f(u_s)\right)$$

and hence the same computations as above give:

$$W_f - \frac{q^2}{\#J(\mathbb{F}_q)} = \frac{1}{\#J(\mathbb{F}_q)} \sum_{\chi \neq \chi_0} S_f(\chi) S_f(\overline{\chi}) = \frac{1}{\#J(\mathbb{F}_q)} \sum_{\chi \neq \chi_0} |S_f(\chi)|^2 \tag{6.5}$$

from which the result follows immediately. □

We see from (6.5) that

$$\rho_f - \frac{1}{\#J(\mathbb{F}_q)} \leqslant \frac{B^2}{q}. \tag{6.6}$$

Therefore we have the following corollary.

Corollary 6.1. *If $f\colon \mathbb{F}_q \to X(\mathbb{F}_q)$ is a B-well-distributed encoding into a curve X, then for all $\mathbf{D} \in J(\mathbb{F}_q)$, we have:*

$$\left| \frac{N_s(\mathbf{D})}{q^s} - \frac{1}{\#J(\mathbb{F}_q)} \right| \leqslant \frac{B^s}{q^{s/2}}.$$

Suppose that X is of genus g_X. Then $\#J(\mathbb{F}_q) = q^{g_X} + O(q^{g_X - 1/2})$, so the bound of Corollary 6.1 is negligible compared to $1/\#J(\mathbb{F}_q)$ provided that $s/2 > g_X$. In other words, if f is a well-distributed encoding, then for $s > 2g_X$, *all* elements $D \in J(\mathbb{F}_q)$ have the same number of preimages by $f^{\otimes s}$ up to negligible deviation.

When f is Icart's function, this says that all the points of the target elliptic curve have almost the same number of preimages by $f^{\otimes s}$ for $s \geqslant 3$. This cannot be improved to $s = 2$, as the analysis in the proof of Proposition 5.1 in the previous chapter shows that there is in fact a bounded number of points which have several times more preimages by $f^{\otimes 2}$ than the others.

Nevertheless, we could obtain our indifferentiability result by bounding the statistical distance between the distribution defined by $f^{\otimes 2}$ and the uniform distribution. We can establish a general result of this type for well-distributed encodings.

Theorem 6.2. *If $f\colon \mathbb{F}_q \to X(\mathbb{F}_q)$ is a B-well-distributed encoding into a curve X, then the statistical distance between the distribution defined by $f^{\otimes s}$ on $J(\mathbb{F}_q)$ and the uniform distribution is bounded as:*

$$\sum_{\mathbf{D} \in J(\mathbb{F}_q)} \left| \frac{N_s(\mathbf{D})}{q^s} - \frac{1}{\#J(\mathbb{F}_q)} \right| \leqslant \frac{B^{s-1}}{q^{(s-1)/2}} \sqrt{\rho_f \#J(\mathbb{F}_q) - 1}.$$

Proof. Consider first the sum of squared deviations. Let

$$V_s = \sum_{\mathbf{D} \in J(\mathbb{F}_q)} \left| \frac{N_s(\mathbf{D})}{q^s} - \frac{1}{\#J(\mathbb{F}_q)} \right|^2.$$

Then, as in the proof of Theorem 6.1, we have:

$$\begin{aligned} V_s &= \sum_{\mathbf{D} \in J(\mathbb{F}_q)} \frac{1}{q^{2s} \#J(\mathbb{F}_q)^2} \sum_{\chi, \eta \neq \chi_0} \chi(-\mathbf{D}) \overline{\eta}(-\mathbf{D}) \left(\sum_{u,v \in \mathbb{F}_q} \chi\left(f(u)\right) \overline{\eta}\left(f(v)\right) \right)^s \\ &= \frac{1}{q^{2s} \#J(\mathbb{F}_q)^2} \sum_{\chi, \eta \neq \chi_0} \left(\sum_{\mathbf{D} \in J(\mathbb{F}_q)} \overline{\chi}(\mathbf{D}) \eta(\mathbf{D}) \right) \left(\sum_{u,v \in \mathbb{F}_q} \chi\left(f(u)\right) \overline{\eta}\left(f(v)\right) \right)^s \\ &= \frac{1}{q^{2s} \#J(\mathbb{F}_q)} \sum_{\chi \neq \chi_0} \left| S_f(\chi) \right|^{2s}. \end{aligned}$$

Now, using (6.5), we derive

$$V_s \leqslant \frac{B^{2s-2}}{q^{s-1}} \left(\rho_f - \frac{1}{\#J(\mathbb{F}_q)} \right).$$

On the other hand, by the Cauchy-Schwarz inequality, we have:

$$\sum_{\mathbf{D} \in J(\mathbb{F}_q)} \left| \frac{N_s(\mathbf{D})}{q^s} - \frac{1}{\#J(\mathbb{F}_q)} \right| \leqslant \sqrt{\#J(\mathbb{F}_q) V_s}$$

and combining the two previous estimates concludes the proof. □

In view of (6.6), the following corollary follows.

Corollary 6.2. *If $f\colon \mathbb{F}_q \to X(\mathbb{F}_q)$ is a B-well-distributed encoding into a curve X, then the statistical distance between the distribution defined by $f^{\otimes s}$ on $J(\mathbb{F}_q)$ and the uniform distribution is bounded as:*

$$\sum_{\mathbf{D} \in J(\mathbb{F}_q)} \left| \frac{N_s(\mathbf{D})}{q^s} - \frac{1}{\#J(\mathbb{F}_q)} \right| \leqslant \frac{B^s}{q^{s/2}} \sqrt{\#J(\mathbb{F}_q)}.$$

In particular, we see from Corollary 6.2 that if f is a well-distributed encoding, then for $s > g_X$, the distribution defined by $f^{\otimes s}$ on $J(\mathbb{F}_q)$ is statistically indistinguishable from the uniform distribution. If f is also computable and samplable (which is easily verified to be the case when f is any of the known deterministic encodings), then it is admissible. In particular, the following hash function construction:

$$\mathfrak{H}(m) = f(\mathfrak{h}_1(m)) + \cdots + f(\mathfrak{h}_s(m)) \qquad (s = g_X + 1)$$

is indifferentiable from a random oracle if $\mathfrak{h}_1, \ldots, \mathfrak{h}_s$ are seen as independent random oracles into $\mathbb{F}_q$.

6.3 Character Sums on Curves

Let $Y \to X$ be a morphism of curves[1] over $\mathbb{F}_q$ which is an abelian covering with Galois group G (that is, a non-constant morphism such that the corresponding extension of function fields $\mathbb{F}_q(Y)/\mathbb{F}_q(X)$ is abelian with Galois group G).

Any character of G determines, via the Artin map, a corresponding character on the group of $\mathbb{F}_q$-divisors on X prime to the ramification locus S of $Y \to X$, which extends to a multiplicative map $\chi\colon \mathrm{Div}_{\mathbb{F}_q}(X) \to \mathbb{C}$ vanishing on divisors not prime to S. Let us call such a map χ an *Artin character* of X. One associates to χ a distinguished effective divisor $\mathfrak{f}(\chi)$ of support S called the conductor (in particular, if $Y \to X$ is unramified, $\mathfrak{f}(\chi) = 0$; the character itself is then said to be unramified).

[1] In this section, "curve over k" means smooth, projective, geometrically connected curve over k.

Example 6.1. We consider the following examples of Artin characters.

- Let E be an elliptic curve over $\mathbb{F}_q$. Then any character of the abelian group $E(\mathbb{F}_q)$ extends to an unramified Artin character of E. Indeed, if F denotes the Frobenius endomorphism of E, $(1-F)\colon E \to E$ is an unramified abelian covering with group $G = E(\mathbb{F}_q)$, and characters of G determine Artin characters of E whose restriction to $E(\mathbb{F}_q)$ is as expected.

- More generally, let X be any curve over $\mathbb{F}_q$ with an $\mathbb{F}_q$-point, and J its Jacobian. As usual, we can embed X in J using this rational point. Then any character of the group $J(\mathbb{F}_q)$ extends to an Artin character of X. It is constructed similarly; $(1-F)\colon J \to J$ is again an unramified abelian covering with group $J(\mathbb{F}_q)$ which can be pulled back to an abelian covering $Y \to X$ with group $J(\mathbb{F}_q)$ along the embedding $X \to J$.

- If χ is an Artin character on X, and $h : \widetilde{X} \to X$ is a non constant morphism of curves, there is a natural Artin character $\widetilde{\chi} = h^*\chi$ on $\widetilde{X}$ obtained by pulling back the abelian covering of X along h. On divisors, $\widetilde{\chi}$ can be defined as $\widetilde{\chi}(\mathbf{D}) = \chi(h_*\mathbf{D})$. Clearly, if χ is unramified, then $\widetilde{\chi}$ is too, and more generally, $\mathfrak{f}(\widetilde{\chi}) = h^*\mathfrak{f}(\chi)$.

- Assume that q is odd. The nontrivial quadratic character (i.e. the "generalized Legendre symbol") $\chi_q(\cdot)$ on $\mathbb{P}^1(\mathbb{F}_q)$, which sends the point of abscissa x to 1, -1, or 0 according as whether x is a quadratic residue, a quadratic nonresidue or $0, \infty$, extends to the nontrivial Artin character χ_2 defined by the ramified quadratic covering $\mathbb{P}^1 \to \mathbb{P}^1\colon x \mapsto x^2$. One has $\mathfrak{f}(\chi_2) = (0) + (\infty)$.

- More generally, if X is a curve over $\mathbb{F}_q$ and φ a non constant, rational function on X, the Legendre symbol of φ extends to a Artin character on X, namely $\varphi^*\chi_2$. Its conductor is the sum of the divisor of zeros of φ and its divisor of poles. In particular, $\deg \mathfrak{f}(\varphi^*\chi_2) = 2 \deg \varphi$.

When χ is nontrivial, Weil [Wei95] has proved the following estimate for sums related to χ, as a consequence of the Riemann hypothesis for curves (see, for example, [KS00, §2] or [Ros02, Chapter 9]). For any Artin character χ of X, let:

$$S_X(\chi) = \sum_{\mathbf{P} \in X(\mathbb{F}_q)} \chi(\mathbf{P}).$$

Lemma 6.1. *If χ is nontrivial and X is of genus g, one has*

$$|S_X(\chi)| \leqslant (2g - 2 + \deg \mathfrak{f}(\chi))\, \sqrt{q}.$$

We can now easily deduce the following result, which forms the basis of the proofs of well-distributedness in §6.4.

Theorem 6.3. *Let $h : \widetilde{X} \to X$ be a non constant morphism of curves, and χ be any nontrivial character of $J(\mathbb{F}_q)$, where J is the Jacobian of X. Assume that h does not*

factor through a nontrivial unramified morphism $Z \to X$. Then:

$$\left| \sum_{\mathbf{P} \in \widetilde{X}(\mathbb{F}_q)} \chi(h(\mathbf{P})) \right| \leqslant (2\widetilde{g} - 2)\sqrt{q} \tag{6.7}$$

where $\widetilde{g}$ is the genus of $\widetilde{X}$. Furthermore, if q is odd and φ is a non constant rational function on $\widetilde{X}$:

$$\left| \sum_{\mathbf{P} \in \widetilde{X}(\mathbb{F}_q)} \chi(h(\mathbf{P}))\chi_q(\varphi(\mathbf{P})) \right| \leqslant (2\widetilde{g} - 2 + 2\deg\varphi)\sqrt{q}. \tag{6.8}$$

Proof. Denote also by χ the Artin character of X extending the given character of $J(\mathbb{F}_q)$. The left-hand side of (6.7) is then just $|S_{\widetilde{X}}(h^*\chi)|$, and we know that $h^*\chi$ is an unramified Artin character of $\widetilde{X}$, so the inequality follows from Lemma 6.1 provided that we can prove that $h^*\chi$ is nontrivial. But if it is a trivial character, h_* maps all divisors of $\widetilde{X}$ to the kernel of χ: this means that h factors through the unramified covering $Z \to X$ defined by the kernel of χ, which is impossible by hypothesis. Hence inequality (6.7).

Similarly, the left-hand side of (6.8) is $|S_{\widetilde{X}}(\widetilde{\chi})|$ for $\widetilde{\chi}$ the Artin character of $\widetilde{X}$ defined as the product of $h^*\chi$ and $\varphi^*\chi_2$. This character cannot be trivial: otherwise, $\varphi^*\chi_2$ would be the inverse of $h^*\chi$, and hence unramified, which it is not. Thus, Lemma 6.1 gives $|S_{\widetilde{X}}(\widetilde{\chi})| \leqslant (2\widetilde{g} - 2 + \deg \mathfrak{f}(\widetilde{\chi}))\sqrt{q}$. Since $h^*\chi$ is unramified, we have $\deg \mathfrak{f}(\widetilde{\chi}) = \deg \mathfrak{f}(\varphi^*\chi_2) = 2\deg\varphi$, which concludes the proof. □

6.4 Examples of Well-Distributed Encodings

6.4.1 Icart's encoding

Recall from §3.5.2 that Icart [Ica09] defined a family of deterministic functions to elliptic curves $E\colon y^2 = x^3 + ax + b$ over finite fields $\mathbb{F}_q$ such that $q \equiv 2 \pmod 3$. His map $f\colon \mathbb{F}_q \to E(\mathbb{F}_q)$ is given by $u \mapsto (x, y)$ with

$$x = \left(v^2 - b - \frac{u^2}{27}\right)^{1/3} + \frac{u^2}{3} \quad \text{and} \quad y = ux + v \tag{6.9}$$

where $v = (3a - u^4)/(6u)$. The image of 0 is chosen as the neutral element of the elliptic curve.

Icart showed that a point (x, y) is the image of u if and only if

$$u^4 - 6xu^2 + 6yu - 3a = 0. \tag{6.10}$$

This makes it possible to give a simple geometric interpretation of Icart's function, as has been done in [FSV10, FT10a, BCI+10a].

Indeed, let $\mathbb{K} = \mathbb{F}_q(E)$ be the function field of E, and introduce the smooth projective curve C whose function field is the quartic extension $\mathbb{L} = \mathbb{K}[u]/(P)$ of $\mathbb{K}$, where $P = u^4 - 6xu^2 + 6yu - 3a$. The inclusions $\mathbb{F}_q(u) \subset \mathbb{L}$ and $\mathbb{K} \subset \mathbb{L}$ give rise to rational maps $g\colon C \to \mathbb{P}^1$ and $h\colon C \to E$ which are in fact morphisms, since these curves are smooth and complete.

Then, Icart's function can be described as $f(u) = h(g^{-1}(u))$. This is well-defined when $q \equiv 2 \pmod 3$ because in that case, g induces a bijection from the set of points in $C(\mathbb{F}_q)$ which are not poles of u to $\mathbb{A}^1(\mathbb{F}_q) = \mathbb{F}_q$.

This geometric point of view makes it possible to express the character sum $S_f(\chi)$, for any character χ of $E(\mathbb{F}_q)$, in terms of the Artin character sum $S_C(h^*\chi)$ (they are the same up to a few "bad points"), and then to use Theorem 6.3 to show that Icart's function is well-distributed.

Theorem 6.4. *Let f be Icart's function* (6.9)*. For any nontrivial character χ of $E(\mathbb{F}_q)$, the character sum $S_f(\chi)$ given by* (6.2) *satisfies:*

$$|S_f(\chi)| \leqslant 12\sqrt{q} + 3.$$

Proof. The map $h\colon C \to E$ defined above is a non constant morphism of curves. Moreover, we know from the analysis of Chapters 4–5 (see also [FSV10]) that if $a \neq 0$, C is of genus 7, and that the Galois closure of the quartic extension $\mathbb{F}_q(C)/\mathbb{F}_q(E)$ has Galois group S_4 (for $a = 0$, the discussion in analogous). It follows that $\mathbb{F}_q(C)/\mathbb{F}_q(E)$ has no nontrivial intermediate extension, and it is clearly ramified (because an unramified covering of an elliptic curve must be of genus 1). Thus, h fulfills the hypotheses of Theorem 6.3, and we get

$$\left| \sum_{\mathbf{P} \in C(\mathbb{F}_q)} \chi(h(\mathbf{P})) \right| \leqslant (2 \cdot 7 - 2)\sqrt{q} = 12\sqrt{q}.$$

Now recall that g induces a bijection from $C(\mathbb{F}_q) \setminus \{\text{poles of } u\}$ to $\mathbb{F}_q$. Thus:

$$\sum_{\mathbf{P} \in C(\mathbb{F}_q)} \chi(h(\mathbf{P})) = S_f(\chi) + \sum_{\substack{\mathbf{P} \in C(\mathbb{F}_q) \\ u(\mathbf{P}) = \infty}} \chi(h(\mathbf{P})).$$

It is shown in the proof of Lemma 5.2 in the previous chapter that u has exactly 3 poles on C, hence the result. □

In other words, Theorem 6.4 asserts that f is a $(12+3q^{-1/2})$-well-distributed encoding.

In particular, as a well-distributed encoding to a curve of genus 1, Icart's function f satisfies that $f^{\otimes s}$ is regular for $s \geqslant 2$. Since $f^{\otimes 2}$ is clearly computable and samplable, it is admissible. Thus, we get a new, simpler and conceptually different proof of the main result of the previous chapter, that

$$\mathfrak{H}(m) = f\big(\mathfrak{h}_1(m)\big) + f\big(\mathfrak{h}_2(m)\big)$$

is indifferentiable from a random oracle when $\mathfrak{h}_1, \mathfrak{h}_2$ are seen as random oracles to $\mathbb{F}_q$.

We also get more general results, since we have information about $f^{\otimes s}$ for $s > 2$, and the bounds we obtain are also sharper than those in the previous chapters. If we denote the collision probability for f by ρ_f, we have:

$$\rho_f = q^{-2} \sum_{\mathbf{P} \in E(\mathbb{F}_q)} \#f^{-1}(\mathbf{P})^2 \leqslant 4q^{-2} \sum_{\mathbf{P} \in E(\mathbb{F}_q)} \#f^{-1}(\mathbf{P}) = 4q^{-1}$$

because points on E have at most 4 preimages. Theorem 6.1 then gives:

Corollary 6.3. *For all $\mathbf{P} \in E(\mathbb{F}_q)$ and all s, we have*

$$\left| \frac{N_s(\mathbf{P})}{q^s} - \frac{1}{\#E(\mathbb{F}_q)} \right| \leqslant \frac{\left(12 + 3q^{-1/2}\right)^{s-2} \left(4\#E(\mathbb{F}_q)q^{-1} - 1\right))}{q^{s/2-1}\#E(\mathbb{F}_q)}.$$

Similarly, Theorem 6.2 gives:

Corollary 6.4. *For all s, the statistical distance between the distribution given by $f^{\otimes s}$ and the uniform distribution on $E(\mathbb{F}_q)$ is bounded as*

$$\sum_{\mathbf{P} \in E(\mathbb{F}_q)} \left| \frac{N_s(\mathbf{P})}{q^s} - \frac{1}{\#E(\mathbb{F}_q)} \right| \leqslant \frac{(12 + 3q^{-1/2})^{s-1} \sqrt{4\#E(\mathbb{F}_q)q^{-1} - 1}}{q^{(s-1)/2}}.$$

Since $\#E(\mathbb{F}_q) = q + O(q^{1/2})$, the bounds of Corollaries 6.3 and 6.4 simplify as

$$\left| \frac{N_s(\mathbf{P})}{q^s} - \frac{1}{\#E(\mathbb{F}_q)} \right| \leqslant \frac{3 \cdot 12^{s-2} + O\left(q^{-1/2}\right)}{q^{s/2}} \tag{6.11}$$

and

$$\sum_{\mathbf{P} \in E(\mathbb{F}_q)} \left| \frac{N_s(\mathbf{P})}{q^s} - \frac{1}{\#E(\mathbb{F}_q)} \right| \leqslant \frac{\sqrt{3} \cdot 12^{s-1} + O\left(q^{-1/2}\right)}{q^{(s-1)/2}}. \tag{6.12}$$

For $s = 2$, this gives a bound of the statistical distance of the form $12\sqrt{3}q^{-1/2} + O(q^{-1})$, which is a significant improvement over the estimate from §5.3. We can refine this further by computing the collision probability precisely.

Remark 6.1. As we saw in Chapter 4, the Chebotarev density theorem gives the prevalence of points with any given number of preimages: the number of permutations in $\mathrm{Gal}(\mathbb{F}_q(C)/\mathbb{F}_q(E)) = S_4$ with exactly 1 (respectively 2, 4) fixed point(s) is 8 (respectively 6, 1), so the density of points with 1 (respectively 2, 4) preimages is 8/24 (respectively 6/24, 1/24). Using an effective version of the Chebotarev density theorem, this gives:

$$q^2\rho_f = 1^2 \cdot \left(\frac{8q}{24} + O(\sqrt{q})\right) + 2^2 \cdot \left(\frac{6q}{24} + O(\sqrt{q})\right) + 4^2 \cdot \left(\frac{q}{24} + O(\sqrt{q})\right).$$

Thus $\rho_f = 2q^{-1} + O(q^{-3/2})$, which allows us to drop the factors 3 and $\sqrt{3}$ from (6.11) and (6.12), respectively.

6.4.2 The Kammerer-Lercier-Renault encodings

As discussed in §3.5.3, Kammerer, Lercier and Renault [KLR10] have recently introduced a series of new encodings based on the same principles as Icart's function, namely solving curve equations in radicals. One such example is an encoding f to hyperelliptic curves of genus 2 over fields $\mathbb{F}_q$ such that $q \equiv 2 \pmod 3$ of the form:

$$H\colon x^3 + (y+c)\left(3x + 2a + \frac{2b}{y}\right) = 0.$$

The precise description of the encoding is rather complicated, and we refer the reader to [KLR10, §3.2.2] for details, but the geometric picture is the same as for Icart's function. The parameter u defining the encoding satisfies a relation of the form:

$$4u^2\left(u^2 - 3y - a^2 - c\right)^3 + u^8 + \alpha u^4 + \beta u^2 + \gamma = 0 \tag{6.13}$$

where α, β, γ are constants in $\mathbb{F}_q$ defined in terms of a, b, c, which we assume are nonzero.

In particular, if $\mathbb{K} = \mathbb{F}_q(x, y)$ is the function field of H, one can consider the extension $\mathbb{L} = \mathbb{K}[u]/(P)$ where P is the polynomial of degree 8 given by equation 6.13. This defines a covering $h\colon C \to H$ of degree 8 by a certain smooth projective curve C, and the rational function u on C provides a morphism $g\colon C \to \mathbb{P}^1$ of degree 9 which induces a bijection on $\mathbb{F}_q$-rational points over $\mathbb{A}^1(\mathbb{F}_q) \setminus S$ where S is a finite set of points shown in [KLR10] to be of size at most 75. The encoding $f\colon \mathbb{F}_q \to H(\mathbb{F}_q)$ is then defined as $u \mapsto h(g^{-1}(u))$ for $u \in \mathbb{F}_q \setminus S$, and is extended in some way to all of $\mathbb{F}_q$.

Our machinery applies again to show that f is a well-distributed encoding. Denote by J the Jacobian of H, and fix an embedding $H \to J$.

Theorem 6.5. *Let f be the encoding function described above. For any nontrivial character χ of $J(\mathbb{F}_q)$, the character sum $S_f(\chi)$ given by* (6.2) *satisfies:*

$$|S_f(\chi)| \leqslant 96\sqrt{q} + 759.$$

Proof. The map $h\colon C \to H$ defined above is a non constant morphism of curves. Let us compute the genus of C.

Note first that $P(u)$ can be written as $Q(t)$ where $t = u^2$, which defines a factorization $C \to D \to H$, with $[D : H] = 4$. The discriminant of Q is a polynomial of degree 12 in y, and each of its 12 roots corresponds to 3 ramified points on H, since each value of y corresponds to 3 values of x. Moreover, when regarded as a polynomial over $\mathbb{F}_q((1/y))$, the quartic Q has a Newton polygon with integer slopes (-3 with length 1 and 1 with length 3). Thus, D is unramified over points with $y = \infty$. All in all, the Riemann-Hurwitz formula gives $2g_D - 2 = 4 \cdot (2g_H - 2) + 3 \cdot 12 = 44$: D is of genus 23.

Then, the quadratic covering $C \to D$ is ramified exactly at points such that $t = 0$ or $t = \infty$. At finite distance, this gives $\gamma = 0$, which is excluded, so all the ramification is at infinity. By the previous Newton polygon argument, over each point of H with $y = \infty$ lie 4 points of D, one with $t = 0$ and three with $t = \infty$. And there are 2 such points of H, by another Newton polygon argument. Hence $2g_C - 2 = 2 \cdot (2g_D - 2) + 2 \cdot 4 = 96$, and C is of genus 49.

Let us now show that $h\colon C \to H$ does not factor nontrivially through an unramified covering. To see this, consider $D_0 \to \mathbb{P}^1$, the ramified covering of degree 4 defined by Q (which pulls back to $D \to H$ along $x\colon H \to \mathbb{P}^1$). We see like before that all points of D_0 ramified over $\mathbb{P}^1$ have ramification index 2. Thus, the monodromy group of this covering is a transitive subgroup of S_4 generated by transpositions, hence all of S_4.

By inspection of the ramification of $x\colon H \to \mathbb{P}^1$, it follows that $D \to H$ also has S_4 as its monodromy group. In particular, it has no quadratic subcovering. Now suppose $h\colon C \to H$ factors through some abelian unramified extension $Z \to H$, which we can assume is quadratic. Then the function fields $\mathbb{F}_q(Z)$ and $\mathbb{F}_q(D)$ are everywhere linearly disjoint over $\mathbb{F}_q(H)$ (i.e. all of their embeddings in some algebraic closure of $\mathbb{F}_q(H)$ are linearly disjoint). Thus $\mathbb{F}_q(C) = \mathbb{F}_q(D) \otimes_{\mathbb{F}_q(H)} \mathbb{F}_q(D)$, and in particular, $C \to D$ is the pullback of $Z \to H$ along $D \to H$. This implies that $C \to D$ is unramified, which we know is not the case.

Thus, h does not factor nontrivially through an abelian unramified covering, and hence fulfills the hypotheses of Theorem 6.3. We get

$$\left| \sum_{\mathbf{P} \in C(\mathbb{F}_q)} \chi(h(\mathbf{P})) \right| \leqslant (2g_C - 2)\sqrt{q} = 96\sqrt{q}.$$

Now recall that g induces a bijection from $C(\mathbb{F}_q) \setminus g^{-1}(S \cup \{\infty\})$ to $\mathbb{F}_q \setminus S$. Thus:

$$\begin{aligned} \sum_{\mathbf{P} \in C(\mathbb{F}_q)} \chi(h(\mathbf{P})) &= \sum_{u \in \mathbb{F}_q \setminus S} \chi(f(u)) + \sum_{\mathbf{P} \in g^{-1}(S \cup \{\infty\})} \chi(h(\mathbf{P})) \\ &= S_f(\chi) - \sum_{u \in S} \chi(f(u)) + \sum_{\mathbf{P} \in g^{-1}(S \cup \{\infty\})} \chi(h(\mathbf{P})). \end{aligned}$$

Since $\#S \leqslant 75$ and g is of degree 9, we get

$$|S_f(\chi)| \leqslant 96\sqrt{q} + 9 \cdot 76 + 75 = 96\sqrt{q} + 759$$

as required. □

In other words, f is a $(96 + 759q^{-1/2})$-well-distributed encoding to H. In particular, as a well-distributed encoding to a curve of genus 2, it satisfies that $f^{\otimes s}$ is regular for any $s \geqslant 3$. Thus, $f^{\otimes 3}$ is regular and clearly also computable and samplable, so the following construction:

$$\mathfrak{H}(m) = f\big(\mathfrak{h}_1(m)\big) + f\big(\mathfrak{h}_2(m)\big) + f\big(\mathfrak{h}_3(m)\big) \in J(\mathbb{F}_q)$$

is indifferentiable from a random oracle when $\mathfrak{h}_1, \mathfrak{h}_2, \mathfrak{h}_3$ are seen as random oracles to $\mathbb{F}_q$. This is the first example of an efficient, well-behaved hash function to the Jacobians of a large family of curves of genus 2.

We now estimate (quite coarsely) the collision probability for f: since h is of degree 8, any given point on H has at most 8 preimages by f. Thus,

$$\rho_f = q^{-2} \sum_{\mathbf{P} \in H(\mathbb{F}_q)} \#f^{-1}(\mathbf{P})^2 \leqslant 8q^{-2} \sum_{\mathbf{P} \in H(\mathbb{F}_q)} \#f^{-1}(\mathbf{P}) = 8q^{-1}$$

and so $q\rho_f \leqslant 8$ as required. Thus, Theorem 6.1 gives the following estimate:

Corollary 6.5. *For all* $\mathbf{D} \in J(\mathbb{F}_q)$ *and all* s*, we have*

$$\left|\frac{N_s(\mathbf{D})}{q^s} - \frac{1}{\#J(\mathbb{F}_q)}\right| \leqslant \frac{8\left(96 + 759q^{-1/2}\right)^{s-2}}{q^{s/2}}.$$

Furthermore, from Theorem 6.2 we derive:

Corollary 6.6. *For all* s*, the statistical distance between the distribution given by* $f^{\otimes s}$ *and the uniform distribution on* $J(\mathbb{F}_q)$ *is bounded as*

$$\sum_{\mathbf{D}\in J(\mathbb{F}_q)} \left|\frac{N_s(\mathbf{D})}{q^s} - \frac{1}{\#J(\mathbb{F}_q)}\right| \leqslant \frac{\sqrt{8}\cdot(96 + 759q^{-1/2})^{s-1}\sqrt{\#J(\mathbb{F}_q)}}{q^{s/2}}.$$

Using that $\#J(\mathbb{F}_q) = q^2 + O(q^{3/2})$ one can also obtain simplified versions of Corollaries 6.5 and 6.6 similar to the bounds (6.11) and (6.12).

Of course, we have considered only one of the encodings from [KLR10], but the same technique applies to all of them. In some cases, it is even much easier to apply. For example, in the case of the encoding to hyperelliptic curves $H_a\colon y^2 = x^{2d} + x^d + a$, the covering curve is H_a itself (the parameter u is in fact just $y - x^d \in \mathbb{F}_q(x, y)$), so the hypotheses of Theorem 6.3 are trivially verified.

6.4.3 The simplified SWU encoding

Recall from §3.5.4 the description of the simplified Shallue-van de Woestijne-Ulas encoding over fields $\mathbb{F}_q$ with $q \equiv 3 \pmod 4$, as introduced in [BCI+10a] with the sign tweak from [FT10a]. The function is based on the following observation. Let $g(x) = x^3+ax+b \in \mathbb{F}_q[x]$ with $ab \neq 0$, and define:

$$X_2(u) = -\frac{b}{a}\left(1 + \frac{1}{u^4 - u^2}\right) \qquad X_3(u) = -u^2 X_2(u)$$

and

$$Z(u) = u^3 g(X_2(u)).$$

Then we have $Z(u)^2 = -g(X_2(u)) \cdot g(X_3(u))$.

Let $S = \{0, 1, -1\} \cup \{\text{roots of } g(X_j(u)) = 0,\ j = 2, 3\}$. For any $u \notin S$, $X_2(u)$ and $X_3(u)$ are well-defined and non zero. Since -1 is a quadratic nonresidue in $\mathbb{F}_q$, this implies that for any $u \in \mathbb{F}_q \setminus S$, exactly one of $g\big(X_2(u)\big)$ or $g\big(X_3(u)\big)$ is a square. Therefore, we can define an encoding function f to the elliptic curve $E\colon y^2 = x^3 + ax + b$ by

$$f(u) = \left(X_j(u)\ ;\ (-1)^j\sqrt{g(X_j(u))}\right),$$

where $j = 2$ or 3 is the index such that $g\big(X_3(u)\big)$ is a square, and $\sqrt{\cdot}$ is the standard square root in $\mathbb{F}_q$, given by exponentiation by $(q + 1)/4$ (thus, the y-coordinate is a quadratic residue if $j = 2$ and a quadratic nonresidue if $j = 3$).

As discussed in §4.5, this encoding function admits the following geometric interpretation. It is easy to see that for all $u \in \mathbb{F}_q \setminus \{-1, 0, 1\}$,

$$x = X_2(u) \iff u^4 - u^2 + \frac{1}{\omega} = 0$$
$$x = X_3(u) \iff u^4 - \omega u^2 + \omega = 0$$

where $\omega = \frac{a}{b}x + 1$. Thus, we can introduce coverings $h_j \colon C_j \to E$, $j = 2, 3$, by the smooth projective curves whose function fields are the extensions of $\mathbb{F}_q(x, y)$ defined respectively by $u^4 - u^2 + 1/\omega = 0$ and $u^4 - \omega u^2 + \omega = 0$. The parameter u is a rational function on each of the C_j giving rise to morphisms $g_j \colon C_j \to \mathbb{P}^1$, such that any point in $\mathbb{A}^1(\mathbb{F}_q) \setminus S$ has exactly two preimages in one of the two curves $C_j(\mathbb{F}_q)$, $j = 2, 3$, and none in the other.

If $u \in \mathbb{F}_q \setminus S$ has its preimages in $C_j(\mathbb{F}_q)$, those preimages are conjugate under $y \mapsto -y$, so that exactly one of them satisfies $\chi_q(y) = (-1)^j$. Let $\mathbf{P} \in C_j(\mathbb{F}_q)$ be that preimage. Then, $f(u) = h_j(\mathbf{P})$. This geometric interpretation is enough to apply Theorem 6.3 and establish that f is well-distributed.

Theorem 6.6. *Let f be the encoding described above. For any nontrivial character χ of $E(\mathbb{F}_q)$, the character sum $S_f(\chi)$ given by* (6.2) *satisfies:*

$$|S_f(\chi)| \leqslant 52\sqrt{q} + 151.$$

Proof. In light of the previous discussion, the character sum $S_f(\chi)$, when restricted to parameters u in $\mathbb{F}_q \setminus S$, can be written as:

$$\sum_{u \notin S} \chi(f(u)) = \sum_{\substack{\mathbf{P} \in C_2(\mathbb{F}_q) \setminus S_2 \\ \chi_q(y) = +1}} \chi(h_2(\mathbf{P})) + \sum_{\substack{\mathbf{P} \in C_3(\mathbb{F}_q) \setminus S_3 \\ \chi_q(y) = -1}} \chi(h_3(\mathbf{P})) \tag{6.14}$$

where $S_j = g_j^{-1}(S \cup \{\infty\})$. To estimate such sums, observe that:

$$\begin{aligned} \sum_{\mathbf{P} \in C_j(\mathbb{F}_q)} \chi(h_j(\mathbf{P})) \cdot \left(\frac{1 + (-1)^j \chi_q(y)}{2} \right) \\ = \sum_{\substack{\mathbf{P} \in C_j(\mathbb{F}_q) \\ \chi_q(y) = (-1)^j}} \chi(h_j(\mathbf{P})) + \frac{1}{2} \sum_{\substack{\mathbf{P} \in C_j(\mathbb{F}_q) \\ \chi_q(y) = 0}} \chi(h_j(\mathbf{P})). \end{aligned} \tag{6.15}$$

The first sum on the right-hand side of (6.15) is almost what we want (up to a bounded number of "bad terms") and the second sum on the right-hand side contains at most $3 \cdot 4 = 12$ terms.

Furthermore, the left-hand side of (6.15) is directly estimated using Theorem 6.3. Indeed, by the Eisenstein criterion, h_2 and h_3 are totally ramified over points in H such that $\omega = 0$ (that is, $x = -b/a$), so they cannot factor through any unramified covering of

E. Hence, Theorem 6.3 applies and gives:

$$\left| \sum_{\mathbf{P} \in C_j(\mathbb{F}_q)} \chi(h_j(\mathbf{P})) \cdot \left(\frac{1 + (-1)^j \chi_q(y)}{2} \right) \right| \leqslant (2g_{C_j} - 2 + \deg y)\sqrt{q}$$

where g_{C_j} is the genus of C_j, and $\deg y$ is the degree of y as a rational function on C_j. Clearly, $\deg y = [\mathbb{F}_q(x, y, u) : \mathbb{F}_q(x, y)] \cdot [\mathbb{F}_q(x, y) : \mathbb{F}_q(y)] = 4 \cdot 3 = 12$. Furthermore, to compute g_{C_j}, note that in addition to the totally ramified points mentioned previously, $C_j \to E$ has ramification type $(2, 2)$ at points with $\omega = 4$ and at infinity, and is unramified elsewhere. Thus, the Riemann-Hurwitz formula gives $2g_{C_j} - 2 = 0 + 2 \cdot 3 + 2 \cdot (1 + 1) + 2 \cdot (1 + 1) = 14$: the curves C_j are of genus 8. Thus:

$$\left| \sum_{\mathbf{P} \in C_j(\mathbb{F}_q)} \chi(h_j(\mathbf{P})) \cdot \left(\frac{1 + (-1)^j \chi_q(y)}{2} \right) \right| \leqslant 26\sqrt{q}.$$

Plugging this estimate back into (6.14) using (6.15), we get:

$$\begin{aligned} \left| \sum_{u \notin S} \chi(f(u)) \right| &\leqslant (26\sqrt{q} + \#S_2 + 6) + (26\sqrt{q} + \#S_3 + 6) \\ &= 52\sqrt{q} + 12 + \#S_2 + \#S_3. \end{aligned}$$

Thus $|S_f(\chi)| \leqslant 52\sqrt{q} + 12 + \#S_2 + \#S_3 + \#S$. Now $\#S \leqslant 3 + 2 \cdot 12 = 27$, and since g_j is a map of degree 2, $\#S_j \leqslant 2(\#S + 1) \leqslant 56$, which concludes the proof. □

Thus, we see from Theorem 6.6 that the simplified Shallue-Woestijne-Ulas encoding is $(52 + 151q^{-1/2})$-well-distributed. As in §6.4.1 and §6.4.2, using Theorem 6.1 and Theorem 6.2, we can deduce precise bounds on the maximum deviation of functions of the form $f^{\otimes s}$ and on the statistical distance of the distribution they define on $E(\mathbb{F}_q)$ and the uniform distribution.

In particular, we get that $f^{\otimes s}$ is regular for $s \geqslant 2$. Since $f^{\otimes 2}$ is clearly computable and samplable, it is admissible, and we obtain that

$$\mathfrak{H}(m) = f(\mathfrak{h}_1(m)) + f(\mathfrak{h}_2(m))$$

is indifferentiable from a random oracle when $\mathfrak{h}_1, \mathfrak{h}_2$ are seen as random oracles to $\mathbb{F}_q$.

Once again, the method generalizes to other SWU-like encodings, such as the Ulas encoding to hyperelliptic curves of the form $y^2 = x^5 + ax + b$.

Chapter 7

Hashing and Encoding to Odd Hyperelliptic Curves

7.1 Introduction

In this chapter, which can serve as an illustration for the results of Chapters 3, 5 and 6, we propose a very simple and efficient encoding function from $\mathbb{F}_q$ to points of a hyperelliptic curve over $\mathbb{F}_q$ of the form $H\colon y^2 = g(x)$ where g is an odd polynomial. Hyperelliptic curves of this type have been frequently considered in the literature to obtain Jacobians of good order and pairing-friendly curves.

Our new encoding is nearly a bijection to the set of $\mathbb{F}_q$-rational points on H. This makes it easy to construct well-behaved hash functions to the Jacobian J of H, as well as injective maps to $J(\mathbb{F}_q)$ which can be used to encode scalars for such applications as El Gamal encryption.

The new encoding is already interesting in the genus 1 case, where it provides a well-behaved encoding to Joux's supersingular elliptic curves.

This work was presented at Pairing 2010 [FT10a].

7.1.1 Hyperelliptic curve encodings.

While there are now rather general and efficient constructions for elliptic curves, encodings to hyperelliptic curves are scarce. The first such encoding was proposed by Ulas in [Ula07], for curves of the form $y^2 = x^n + ax + b$ or $y^2 = x^n + ax^2 + bx$. Kammerer, Lercier and Renault, in their recent paper [KLR10], have presented several additional families of hyperelliptic curves for which an Icart-like encoding can be constructed, but the target curves are still of a special form and may not be convenient to use for cryptographic applications. Efficiency is also a problem for both of these constructions.

Moreover, all of these algorithms construct points on the curve itself, whereas the relevant object in cryptography is the group attached to it, namely its Jacobian variety. We have seen in Chapter 6 how to build a well-behaved hash function to the Jacobian based on an encoding to the curve, but the relatively complex geometric description

of a function like the Kammerer-Lercier-Renault genus 2 encoding led to rather large constants in the results of §6.4.2.

7.1.2 Our contribution

This chapter presents a new encoding for hyperelliptic curves of the form $H : y^2 = g(x)$ where g is an odd polynomial over $\mathbb{F}_q$, with $q = 3 \bmod 4$. From this encoding to the curve H, we also deduce efficient injective encodings and well-behaved hash functions to its Jacobian. The new encoding has the following desirable properties:

- it can be very efficiently computed using one exponentiation and no division, in constant time and without branching;
- the encoding is an efficiently invertible bijection: thus, it is possible to encode messages as points on the curve and recover them. This has numerous applications, e.g. to encryption;
- in genus 1, it provides an encoding to supersingular elliptic curves, similar to Boneh and Franklin's construction [BF01], but for different base fields;
- in higher genus, many cryptographically interesting curves are of the form H, including the curves considered in [FKT03, HKT05, Sat09];
- many constructions of pairing-friendly hyperelliptic curves yield curves of the form H [KT08, FS11];
- since the encoding has a simple geometric description, it is easy to obtain well-behaved hash functions from it, and the corresponding regularity bounds are optimally tight.

7.2 Odd Hyperelliptic Curves

Let g be an odd monic polynomial over a finite field $\mathbb{F}_q$ with $q \equiv 3 \pmod 4$, which has simple roots in $\overline{\mathbb{F}}_q$. We denote its degree by $2k+1$, and consider the hyperelliptic curve over $\mathbb{F}_q$ defined by:

$$H\colon y^2 = f(x) = x^{2k+1} + a_1 x^{2k-1} + \cdots + a_k x$$

Let us call such curves *odd hyperelliptic curves*. Many hyperelliptic curves relevant to cryptography, and particularly pairing-based cryptography, are of this form. For example:

- the supersingular elliptic curves of Joux [Jou02]: $y^2 = x^3 + ax$;
- the genus 2 curves studied by Furukawa et al. [FKT03] and their extension to higher genus by Haneda et al. [HKT05]: $y^2 = x^{2k+1} + ax$ (for which one can compute the zeta function);
- in particular, the Type II pairing-friendly curves of genus 2 constructed by Kawazoe and Takahashi [KT08];

- the genus 2 hyperelliptic curves for which Satoh [Sat09] gave an efficient class group counting algorithm: $y^2 = x^5 + ax^3 + bx$;
- in particular, some of the pairing-friendly genus 2 curves constructed by Freeman and Satoh [FS11] (although the case $q \equiv 1 \pmod 4$ is more common).

Additionally, odd hyperelliptic curves and their Jacobians admit an automorphism of order 4 over $\mathbb{F}_{q^2}$ (namely $(x, y) \mapsto (-x, \sqrt{-1} \cdot y)$) which can be used to map points over $\mathbb{F}_q$ to linearly independent points over $\mathbb{F}_{q^2}$, another useful property for pairings.

Remark 7.1. A hyperelliptic curve over $\mathbb{F}_q$ is birational to an odd hyperelliptic curve when the set of points in $\mathbb{P}^1$ over which it is ramified is invariant under an automorphism of $\mathbb{P}^1$ of order 2 fixing two of them, both $\mathbb{F}_q$-rational. For example, hyperelliptic curves of the form:

$$H' \colon y^2 = x^6 + ax^5 + bx^4 - bx^2 - ax - 1$$

are birational to odd hyperelliptic curves, since they are ramified over a set of points invariant under $x \mapsto 1/x$ and containing ± 1. One possible change of variables is $x \mapsto (x-1)/(x+1)$.

This remark shows that the coarse moduli space of odd hyperelliptic curves of genus k over $\overline{\mathbb{F}}_q$ is a subvariety of dimension $k-1$ of the dimension $2k-1$ moduli space of genus k hyperelliptic curves.

7.3 Our New Encoding

7.3.1 Definition

Let $H \colon y^2 = g(x)$ be an odd hyperelliptic curve over $\mathbb{F}_q$, $q \equiv 3 \pmod 4$. Denote by $\sqrt{\cdot}$ the usual square root function on the set of quadratic residues in $\mathbb{F}_q$ (exponentiation by $(q+1)/4$), and by $\chi_q(\cdot)$ the nontrivial quadratic character of $\mathbb{F}_q$.

Over $\mathbb{F}_q$, -1 is a quadratic nonresidue, and for any $t \in \mathbb{F}_q$, we have $g(-t) = -g(t)$, so unless $g(t) = 0$, exactly one of $g(t)$ or $g(-t)$ is a square. In other words, exactly one of t or $-t$ is the abscissa of an $\mathbb{F}_q$-rational point on H.

This observation allows us to define a point encoding function f to $H(\mathbb{F}_q)$ as follows:

$$\begin{aligned} f \colon \mathbb{F}_q &\longrightarrow H(\mathbb{F}_q) \\ t &\longmapsto \left(\varepsilon(t) \cdot t \ ; \ \varepsilon(t) \sqrt{\varepsilon(t) \cdot g(t)} \right) \end{aligned} \tag{7.1}$$

where $\varepsilon(t) = \chi_q(g(t))$. We claim that this function is well-defined and "almost" a bijection.

More precisely, recall that a *Weierstrass point* of H is a point where the rational function y is ramified: these are the points $(x, 0)$ for x a root of g together with the point at infinity ∞. Then, let $W \subset H(\mathbb{F}_q)$ be the set of $\mathbb{F}_q$-rational Weierstrass points on H, and $T \subset \mathbb{F}_q$ the set of roots of f.

Theorem 7.1. *The function f given by (7.1) is well-defined, maps all points in T to $(0,0) \in W$, and induces a bijection $\mathbb{F}_q \setminus T \to H(\mathbb{F}_q) \setminus W$.*

Proof. For $t \in T$, we have $\varepsilon(t) = 0$, hence $f(t) = (0,0) \in W$. Now let $t \in \mathbb{F}_q \setminus T$, and $x = \varepsilon(t) \cdot t$. Since g is odd and $\varepsilon(t) = \pm 1$, $g(x) = \varepsilon(t) \cdot g(t)$. In particular, recalling that $\chi_q(-1) = -1$, we can write:

$$\chi_q(g(x)) = \chi_q(\varepsilon(t) \cdot g(t)) = \varepsilon(t) \cdot \chi_q(g(t)) = \varepsilon(t)^2 = 1$$

Thus, the second component $y = \varepsilon(t)\sqrt{\varepsilon(t) \cdot g(t)}$ of $f(t)$ is well-defined, and we have $y^2 = \varepsilon(t) \cdot g(t) = g(x)$, so $f(t)$ is an affine point on $H(\mathbb{F}_q)$ as required. The condition $t \notin T$ further implies that $g(t) \neq 0$, so $y \neq 0$. Therefore, $f(t) \in \mathbb{F}_q \setminus W$.

Let us show that the restriction of f to $\mathbb{F}_q \setminus T$ is injective. Indeed, suppose $f(t) = f(u)$ with $t, u \notin T$. Equating x-coordinates, we get $\varepsilon(t) \cdot t = \varepsilon(u) \cdot u$, hence $u = \pm t$. If $u = -t$, then comparing the y-coordinates, we obtain

$$\begin{aligned}\varepsilon(t)\sqrt{\varepsilon(t) \cdot g(t)} &= \varepsilon(u)\sqrt{\varepsilon(u) \cdot g(u)} \\ &= \varepsilon(-t)\sqrt{\varepsilon(-t) \cdot g(-t)} = -\varepsilon(t)\sqrt{\varepsilon(t) \cdot g(t)}\end{aligned}$$

which is a contradiction. Therefore, $t = u$ and f is injective on $\mathbb{F}_q \setminus T$.

Finally, we claim that $f(\mathbb{F}_q \setminus T) = H(\mathbb{F}_q) \setminus W$. To see this, take $(x, y) \in H(\mathbb{F}_q) \setminus W$ and let $t = \delta \cdot x$, where $\delta = \pm 1$ is defined by $y = \delta\sqrt{g(x)}$. We have

$$\varepsilon(t) = \chi_q(g(\delta x)) = \chi_q(\delta \cdot g(x)) = \delta \cdot \chi_q(g(x)) = \delta$$

since $g(x) = y^2$ is a square. Thus:

$$f(t) = \left(\delta^2 \cdot x \ ; \ \delta\sqrt{\delta \cdot g(\delta x)}\right) = \left(x \ ; \ \delta\sqrt{g(x)}\right) = (x; y)$$

as required. □

Corollary 7.1. *The cardinality of $H(\mathbb{F}_q)$ is $q + 1$.*

Proof. From the above, we get $\#H(\mathbb{F}_q) = \#(\mathbb{F}_q \setminus T) + \#W = q - \#T + \#W$. But W consists of the point at infinity on H, and all points of the form $(x, 0)$, $x \in T$. Thus, $\#W = \#T + 1$, and hence $\#H(\mathbb{F}_q) = q + 1$. □

Remark 7.2. • Since f is an efficiently computable bijection between all of $\mathbb{F}_q$ and $H(\mathbb{F}_q)$ except at most $2k + 2$ points on both sides, with an efficiently computable inverse (namely $(x, y) \mapsto \chi_q(y) \cdot x$), it is a very well-behaved encoding function.

In particular, it is clear that if t is uniformly distributed in $\mathbb{F}_q$, the distribution of $f(t)$ in $H(\mathbb{F}_q)$ is statistically indistinguishable from the uniform distribution. According to the results of Chapter 5, it follows that if $\mathfrak{h}$ is a hash function to $\mathbb{F}_q$ modeled as a random oracle, then $\mathfrak{H}(m) = f(\mathfrak{h}(m))$ defines a function into $H(\mathbb{F}_q)$ that is indifferentiable from a random oracle. When the genus of H is at least 2, however,

one is usually interested in hashing to the Jacobian of H rather than H itself. This will be discussed in §7.4.

The fact that, unlike most other encodings, f is almost injective, makes it suitable for other purposes than hashing, such as encoding a message to be encrypted, for example with El Gamal.

- Since $\#T = \#(W \setminus \{\infty\})$, it is in fact easy to modify the definition of f to obtain a bijection $f' \colon \mathbb{F}_q \to H(\mathbb{F}_q) \setminus \{\infty\}$ which misses only one rational point on H. This modified encoding f' is slightly less efficient to compute, however, and using one or the other makes no difference in practice (as one is not concerned with a few exceptional points), so we shall stick to f as defined by (7.1).

- When H is in fact an elliptic curve E (i.e. $\deg g = 3$), Corollary 7.1 says that E is supersingular. These are in fact the supersingular elliptic curves $y^2 = x^3 + ax$ discussed by Joux in [Jou02]. Thus, the function f provides a convenient way to encode points into supersingular elliptic curves over $\mathbb{F}_q$ with $q \equiv 3 \pmod 4$. This is an interesting addition to the original encoding of Boneh and Franklin [BF01], which applies to supersingular curves of the form $y^2 = x^3 + b$ over fields $\mathbb{F}_q$ with $q \equiv 2 \pmod 3$. In particular, our encoding can be used in characteristic 3.

- In the general case, we see that $\#H(\mathbb{F}_{q^n}) = q^n + 1$ for any odd extension degree n. This gives some constraints on the zeta function of H, but in genus $k \geqslant 2$, many isogeny classes are possible for the Jacobian J of H nonetheless, so the proposed encoding applies to a wide range of curves. It is not always easy to determine the order of $J(\mathbb{F}_q)$: an approach is given by Satoh in [Sat09] for $g = 2$.

7.3.2 Efficient computation

The definition of f involves a generalized Legendre symbol and one square root, which suggests that its computation might be costly, especially if it is to be done in constant time, an important property in settings where side-channel attacks are a concern. However, it is actually possible to compute f with a single exponentiation, a few multiplications and no division, making it one of the most efficient deterministic encoding function proposed to date. One such implementation is described as Algorithm 7.1. Note that this implementation is also branch-free, contrary to what happens for encodings such as the one by Shallue and van de Woestijne [SvdW06]; this also prevents certain active side-channel attacks.

To see that this implementation is correct, consider α and β as defined in Algorithm 7.1. For $t \in T$, we have $\alpha = 0$, hence the procedure returns $f(t) = (0, 0)$ as required. Now let $t \notin T$. We have

$$\beta^2 = \alpha^{\frac{q-3}{2}} = \frac{1}{\alpha}\chi_q(\alpha) = \frac{\varepsilon(t)}{\alpha}$$

In particular, $\alpha\beta^2 t = \varepsilon(t) \cdot t$ is indeed the abscissa of $f(t)$.

Algorithm 7.1 Constant-time, single-exponentiation implementation of the encoding f. The constant r is $(q-3)/4$ if $q \equiv 3 \pmod 8$, $(q-3)/4 + (q-1)/2$ otherwise.

```
1: function f(t)
2:     α ← g(t)
3:     β ← α^r
4:     return (αβ²t, αβ)
5: end function
```

Moreover, suppose $q \equiv 3 \pmod 8$. Then $(q+1)/4$ is odd and $\varepsilon(t) = \pm 1$, so we have

$$\begin{aligned}\alpha\beta = \alpha^{\frac{q-3}{4}+1} &= \varepsilon(t) \cdot \varepsilon(t) \cdot g(t)^{\frac{q+1}{4}} \\ &= \varepsilon(t) \cdot \big(\varepsilon(t) \cdot g(t)\big)^{\frac{q+1}{4}} = \varepsilon(t)\sqrt{\varepsilon(t) \cdot g(t)}\end{aligned}$$

so the algorithm is correct.

Similarly, when $q \equiv 7 \pmod 8$, $(q+1)/4$ is odd and we obtain

$$\begin{aligned}\alpha\beta = \alpha^{\frac{q-1}{2}+\frac{q-3}{4}+1} &= \varepsilon(t) \cdot g(t)^{\frac{q+1}{4}} \\ &= \varepsilon(t) \cdot \big(\varepsilon(t) \cdot g(t)\big)^{\frac{q+1}{4}} = \varepsilon(t)\sqrt{\varepsilon(t) \cdot g(t)}\end{aligned}$$

which concludes.

7.4 Mapping to the Jacobian

In the previous section, we have constructed a function $f\colon \mathbb{F}_q \to H(\mathbb{F}_q)$ which is efficiently computable and has a number of desirable properties. For cryptographic purposes, however, we are usually interested in obtaining elements of a group attached to the curve, namely the Jacobian, rather than points on the curve itself. In the case of elliptic curves, the curve and its Jacobian are isomorphic so no further work is needed, but for curves of genus $k \geqslant 2$, they are quite different objects.

In the following, we always denote the Jacobian of H by J, and we regard H as embedded in J via the map $H \to J$ sending a point $\mathbf{P}$ to the class of the degree 0 divisor $(\mathbf{P}) - (\infty)$. In particular, if $\mathbf{P}, \mathbf{Q}$ are points in $H(\mathbb{F}_q)$, $\mathbf{P} + \mathbf{Q}$ denotes the class of $(\mathbf{P}) + (\mathbf{Q}) - 2(\infty)$.

We propose two constructions of maps to $J(\mathbb{F}_q)$ to accommodate for different use cases: an injective map with large image, which can be used to encode scalars as group elements (e.g. for encryption), and a map defining an essentially uniform distribution on $J(\mathbb{F}_q)$, to obtain well-behaved hash functions.

7.4.1 Injective encoding to the Jacobian

Let us first recall a few facts about hyperelliptic curves, for which we refer for example to [MWZ98]. Elements of $J(\mathbb{F}_q)$ are classes of $\mathbb{F}_q$-divisors on H and admit a canonical

representation as so-called *reduced divisors* defined over $\mathbb{F}_q$. Let $\widetilde{\cdot}$ denote the hyperelliptic involution on H, $(x, y) \mapsto (x, -y)$. A divisor $\mathbf{D} = \mathbf{P}_1 + \cdots + \mathbf{P}_r$ (where the $\mathbf{P}_i$ are not necessarily distinct points in $H(\overline{\mathbb{F}}_q)$) is said to be reduced when r is less than or equal to the genus k of H, and $\mathbf{P}_i \neq \widetilde{\mathbf{P}}_j$ for all $i \neq j$. The reduced divisors $\mathbf{D}$ and $\mathbf{D}'$ defined by $\mathbf{P}_1, \ldots, \mathbf{P}_r$ and $\mathbf{P}'_1, \ldots, \mathbf{P}'_r$ are distinct and non-equivalent as soon as the multisets $\{\mathbf{P}_1, \ldots, \mathbf{P}_r\}$ and $\{\mathbf{P}'_1, \ldots, \mathbf{P}'_r\}$ are different. Each divisor class in $J(\mathbb{F}_q)$ contains a unique reduced divisor defined over $\mathbb{F}_q$.

Now, with the notations of §7.3, the encoding $f\colon \mathbb{F}_q \to H(\mathbb{F}_q)$ defined by (7.1) satisfies that for all $t \in \mathbb{F}_q \setminus T$, the only u such that $f(u) = \widetilde{f(t)}$ is $u = -t$. Therefore, if $(t_1, \ldots, t_k)$ is any tuple of k elements of $\mathbb{F}_q \setminus T$ (k being the genus of H) such that $t_i + t_j \neq 0$ for all i, j, then $f(t_1) + \cdots + f(t_k)$ is a reduced divisor. In particular, consider the set X of k-element subsets of $\mathbb{F}_q \setminus T$ not containing any two opposite elements. Then it is immediate from the facts above that the map:

$$\begin{aligned} F_{\text{inj}}\colon X &\longrightarrow J(\mathbb{F}_q) \\ \{t_1, \ldots, t_k\} &\longmapsto f(t_1) + \cdots + f(t_k) \end{aligned}$$

is injective. We have

$$\#X = 2^k \binom{(q - \#T)/2}{k} = \frac{1 - o(1)}{k!} \cdot q^k \geqslant c_k \cdot \#J(\mathbb{F}_q)$$

for some constant $c_k > 0$ depending only on k. Thus, F_{inj} is an injective mapping to $J(\mathbb{F}_q)$ covering a constant positive fraction of all points. It is also very easy to compute since points in the image are directly given as reduced divisors, so no actual arithmetic on the Jacobian is needed.

In the case that is most relevant for cryptographic applications, namely $k = 2$, we can define an even simpler injective encoding, from the set Y of 2-element subsets of $\mathbb{F}_q \setminus T$, which may be easier to manipulate than X:

$$\begin{aligned} F'_{\text{inj}}\colon Y &\longrightarrow J(\mathbb{F}_q) \\ \{t_1, t_2\} &\longmapsto f(t_1) + f(-t_2) \end{aligned}$$

This function injective, easy to compute, and reaches roughly one half of all points in $J(\mathbb{F}_q)$.

7.4.2 Indifferentiable hashing to the Jacobian

One can also use f to construct well-behaved hash functions to $J(\mathbb{F}_q)$. For this purpose, we have shown in Chapter 5 how one could use functions to $J(\mathbb{F}_q)$ with good regularity properties, and described in Chapter 6 how character sum estimates could be used to prove such regularity properties for functions of the form:

$$\begin{aligned} f^{\otimes s}\colon (\mathbb{F}_q)^s &\longrightarrow J(\mathbb{F}_q) \\ (t_1, \ldots, t_s) &\longmapsto f(t_1) + \cdots + f(t_s). \end{aligned}$$

Since f is so simple, we do not really need to rely on the entire framework of Chapter 6. Indeed, the following bound, which in the terminology of the previous chapter says that f is a $(2k-2+\varepsilon)$-well-distributed encoding, can be proved using classical results on characters on algebraic curves. Note that this bound is very tight: it gives a better well-distributedness bound for f in genus up to 6 than can be established for Icart's function in genus 1.

Lemma 7.1. *For any character χ of the abelian group $J(\mathbb{F}_q)$, let*

$$S(\chi) = \sum_{t \in \mathbb{F}_q} \chi(f(t)).$$

Then, whenever χ is nontrivial, we have

$$|S(\chi)| \leqslant (2k-2)\sqrt{q} + 4k + 3.$$

Proof. A nontrivial character χ of $J(\mathbb{F}_q)$ is also a nontrivial, unramified Artin character of H (see [KS00, §2] or §6.3). In particular, the Riemann hypothesis for the L-function on H associated with χ gives:

$$\left| \sum_{\mathbf{P} \in H(\mathbb{F}_q)} \chi(\mathbf{P}) \right| \leqslant (2k-2)\sqrt{q}.$$

The result then follows from the observation that

$$\begin{aligned} \sum_{t \in \mathbb{F}_q} \chi(f(t)) &= \#T \cdot \chi((0,0)) + \sum_{\mathbf{P} \in H(\mathbb{F}_q) \setminus W} \chi(\mathbf{P}) \\ &= \#T \cdot \chi((0,0)) - \sum_{\mathbf{P} \in W} \chi(\mathbf{P}) + \sum_{\mathbf{P} \in H(\mathbb{F}_q)} \chi(\mathbf{P}) \end{aligned}$$

since $\#T + \#W \leqslant 4k+3$. □

We can then proceed like in §6.2 and deduce from this lemma a bound on the statistical distance between the distribution defined on $J(\mathbb{F}_q)$ by $f^{\otimes s}$ and the uniform distribution.

For any $\mathbf{D} \in J(\mathbb{F}_q)$, let $N_s(\mathbf{D})$ denote the number of preimages of $\mathbf{D}$ under $f^{\otimes s}$:

$$N_s(\mathbf{D}) = \#\{(t_1, \ldots, t_s) \in (\mathbb{F}_q)^s \mid \mathbf{D} = f(t_1) + \cdots + f(t_s)\}$$

Then we have the following result:

Theorem 7.2. *The statistical distance between the distribution defined by $f^{\otimes s}$ and the uniform distribution on $J(\mathbb{F}_q)$ is bounded as:*

$$\sum_{\mathbf{D} \in J(\mathbb{F}_q)} \left| \frac{N_s(\mathbf{D})}{q^s} - \frac{1}{\#J(\mathbb{F}_q)} \right| \leqslant \frac{\left(2k+2+(4k+3)q^{-1/2}\right)^s \sqrt{\#J(\mathbb{F}_q)}}{q^{s/2}}.$$

Proof. This results from Corollary 6.2 in the previous chapter. □

Note that $\#J(\mathbb{F}_q) \sim q^k$, so that the bound we get on the statistical distance is in $O(q^{(k-s)/2})$. Therefore, as soon as $s > k$, the distribution defined by $f^{\otimes s}$ on $J(\mathbb{F}_q)$ is statistically indistinguishable from the uniform distribution. In particular, in the terminology of Chapter 5, the encoding $f^{\otimes(k+1)}$ to $J(\mathbb{F}_q)$ is *regular* (see Definition 5.3). It is also obviously computable and samplable, so $f^{\otimes(k+1)}$ is an admissible encoding to $J(\mathbb{F}_q)$.

This provides a simple, well-behaved hash function construction to the Jacobian of H. Indeed, it follows that the hash function defined by

$$\mathfrak{H}(m) = f(\mathfrak{h}_1(m)) + \cdots + f(\mathfrak{h}_{k+1}(m))$$

is indifferentiable from a random oracle when $\mathfrak{h}_1, \ldots, \mathfrak{h}_{k+1}$ are seen as independent random oracles into $\mathbb{F}_q$.

7.5 Conclusion

In this chapter, we provide a very efficient construction of a deterministic encoding into odd hyperelliptic curves. Odd hyperelliptic curves are a simple and relatively large class of hyperelliptic curves, compared to the families of curves covered by previous deterministic encodings. They also include many curves of cryptographic interest (because of efficient point-counting on the Jacobian, or pairing-friendliness), even in the elliptic curve case.

This encoding is almost a bijection, which can be useful for a number of applications, such as encryption, and allows us to construct the first efficient injections with large image to the Jacobians of odd hyperelliptic curves, as well as indifferentiable hash functions to these Jacobians with particularly tight regularity bounds.

Chapter 8

Huff's Model for Elliptic Curves

8.1 Introduction

This chapter moves away from the problem of hashing and encoding to elliptic curves to concentrate on a very different practical concern arising in elliptic curve cryptography: constructing elliptic curve models with efficient arithmetic operations.

Specifically, we revisit an elliptic curve model introduced by Huff in 1948, originally to study a diophantine problem, and present explicit formulas for adding or doubling points on these curves with a number of desirable features, including completeness and independence on curve parameters. We also suggest extensions and generalizations of this model that may be of interest for cryptographic applications, and consider the problem of pairing computations over Huff curves. This work was presented at ANTS-IX [JTV10].

8.1.1 Background

Cryptography and efficient elliptic curve arithmetic. The use of elliptic curves in cryptography makes key sizes smaller compared to other groups where the discrete logarithm and related problems are hard, such as subgroups of $\mathbb{Z}_p^*$, but the arithmetic of the underlying group is more complex (for example, with the widely-used Jacobian coordinates, the general addition of two points on an elliptic curve typically requires 16 multiplications in the base field). Therefore, a large amount of research has been devoted to analyzing how efficient the group operations can be made for many "shapes" of elliptic curves proposed in the mathematical literature: Weierstrass cubics, Jacobi intersections, Hessian curves, Jacobi quartics, or the more recent forms of elliptic curves due to Montgomery, Doche-Icart-Kohel or Edwards (see [BL08] for an encyclopedic overview of these models).

For instance, since 2007, there has been a rapid development of the curves introduced by Edwards in [Edw07] and their use in cryptology. Bernstein and Lange proposed a more general version of these curves in [BL07a] and the *inverted Edwards* coordinates in [BL07b]. Bernstein, Birkner, Joye, Lange, and Peters studied *twisted Edwards* curves in [BBJ+08]. Hisil, Wong, Carter and Dawson proposed *extended twisted Edwards* coordinates in [HWCD08]. Bernstein, Lange, and Farashahi covered the *binary case* in [BLF08].

The first formulas for computing *pairings* over Edwards curves were published by Das and Sarkar [DS08]. They were subsequently improved by Ionica and Joux [IJ08]. The best implementation to date is due to Arène, Lange, Naehrig, and Ritzenthaler [ALNR11]. The present chapter provides a similar study for a forgotten model of elliptic curves hinted to by Huff in 1948.

A diophantine problem. Huff [Huf48] considered rational distance sets S (i.e., subsets S of the plane $\mathbb{R}^2$ such that for all $s, t \in S$, the distance between s and t is a rational number) of the following form: given distinct $a, b \in \mathbb{Q}$, S contains the four points $(0, \pm a)$ and $(0, \pm b)$ on the y-axis, plus points $(x, 0)$ on the x-axis, for some $x \in \mathbb{Q}$. Such a point $(x, 0)$ must then satisfy the equations $x^2 + a^2 = u^2$ and $x^2 + b^2 = v^2$ with $u, v \in \mathbb{Q}$. The system of associated homogeneous equations $x^2 + a^2z^2 = u^2$ and $x^2 + b^2z^2 = v^2$ defines a curve of genus 1 in $\mathbb{P}^3$. Huff, and later his student Peeples [Pee54], provided examples where this curve has positive rank over $\mathbb{Q}$, thus exhibiting examples of arbitrarily large rational distance sets of cardinality $k > 4$ such that exactly $k - 4$ points are on one line.

The above mentioned genus 1 curve is birationally equivalent to the curve

$$ax(y^2 - 1) = by(x^2 - 1) \tag{8.1}$$

for some parameters a and b in $\mathbb{Q}$. It is easily seen that, over any field $\mathbb{K}$ of odd characteristic, Equation (8.1) defines an elliptic curve if $a^2 \neq b^2$ and $a, b \neq 0$. Indeed, if $ab \neq 0$, the gradient of the curve $F(X, Y, Z) = aX(Y^2 - Z^2) - bY(X^2 - Z^2)$ in the projective plane $\mathbb{P}^2(\mathbb{K})$ is

$$\left(\frac{\partial F}{\partial X}, \frac{\partial F}{\partial Y}, \frac{\partial F}{\partial Z}\right) = \left(a(Y^2 - Z^2) - 2bXY, 2aXY - b(X^2 - Z^2), 2(-aX + bY)Z\right),$$

which does not vanish at the three points at infinity $(1 : 0 : 0)$, $(0 : 1 : 0)$ and $(a : b : 0)$ and vanishes at a finite point $(x : y : 1)$ if and only if $ax = by$, which together with equation (8.1) implies that $x^2 = y^2$ and therefore $a^2 = b^2$. It is worth noting that in characteristic 2, the point $(1 : 1 : 1)$ is always singular and therefore the family of curves defined by (8.1) does not contain any smooth curve. As will be shown in §8.3, we can extend our study to even characteristic by considering a generalized model.

8.1.2 Our contributions

Our first contribution is a detailed study of Huff's form for elliptic curves over finite fields of odd characteristic and a statement of the addition law in these groups. We show in particular that all elliptic curves over non-binary finite fields with a subgroup isomorphic to $\mathbb{Z}/4\mathbb{Z} \times \mathbb{Z}/2\mathbb{Z}$ can be transformed to Huff's form. We then analyze their arithmetic and investigate several generalizations and extensions. In particular, we present explicit formulas (i.e., as a series of field operations) that

- compute a *complete* addition $(X_1 : Y_1 : Z_1) \oplus (X_2 : Y_2 : Z_2)$ using 12m;
- compute a *unified* addition $(X_1 : Y_1 : Z_1) \oplus (X_2 : Y_2 : Z_2)$ using 11m;

- compute a mixed addition $(X_1 : Y_1 : Z_1) \oplus (X_2 : Y_2 : 1)$ using $10\mathsf{m}$;
- compute a doubling $[2](X_1 : Y_1 : Z_1)$ using $6\mathsf{m} + 5\mathsf{s}$

where m and s denote multiplications and squarings in the base field $\mathbb{K}$.

As a further contribution, since bilinear pairings have found numerous applications in cryptography, we also present formulas for computing Tate pairings using Huff's form. Specifically, we present explicit formulas that

- compute a *full* Miller addition using $1\mathsf{M} + (k + 15)\mathsf{m}$;
- compute a *mixed* Miller addition using $1\mathsf{M} + (k + 13)\mathsf{m}$;
- compute a Miller doubling using $1\mathsf{M} + 1\mathsf{S} + (k + 11)\mathsf{m} + 6\mathsf{s}$

on a Huff curve over $\mathbb{K} = \mathbb{F}_q$ of embedding degree k. M and S denote multiplications and squarings in the larger field $\mathbb{F}_{q^k}$ while m and s are operations in $\mathbb{F}_q$ as before.

Outline. The rest of this chapter is organized as follows. The next section introduces Huff's model. We develop efficient unified addition formulas and discuss the applicability of the model. We explicit the class of elliptic curves covered by Huff's model. In §8.3, we present several generalizations and extensions. We offer dedicated addition formulas. We generalize Huff's model to cover a larger class of elliptic curves. We also extend the model to the case of binary fields. §8.4 deals with pairings over Huff curves. We exploit the relative simplicity of the underlying group law to devise efficient formulas for the evaluation of the Tate pairing. Finally, we conclude in §8.5.

8.2 Huff's Model

Let $\mathbb{K}$ denote a field of characteristic $\neq 2$. Consider the set of projective points $(X : Y : Z) \in \mathbb{P}^2(\mathbb{K})$ satisfying the equation

$$E_{/\mathbb{K}} : aX(Y^2 - Z^2) = bY(X^2 - Z^2) \tag{8.2}$$

where $a, b \in \mathbb{K}^\times$ and $a^2 \neq b^2$. This form is referred to as *Huff's model* of an elliptic curve.

The tangent line at $(0 : 0 : 1)$ is $aX = bY$, which intersects the curve with multiplicity 3, so that $\mathbf{O} = (0 : 0 : 1)$ is an inflection point of E. $(E, \mathbf{O})$ is therefore an elliptic curve with $\mathbf{O}$ as neutral element and whose group law, denoted $\oplus$, has the following property: for any line intersecting the cubic curve E at the three points $\mathbf{P}_1$, $\mathbf{P}_2$ and $\mathbf{P}_3$ (counting multiplicities), we have $\mathbf{P}_1 \oplus \mathbf{P}_2 \oplus \mathbf{P}_3 = \mathbf{O}$. In particular, the inverse of point $\mathbf{P}_1 = (X_1 : Y_1 : Z_1)$ is $\ominus\mathbf{P}_1 = (X_1 : Y_1 : -Z_1)$ and the sum of $\mathbf{P}_1$ and $\mathbf{P}_2$ is $\mathbf{P}_1 \oplus \mathbf{P}_2 = \ominus\mathbf{P}_3$. We note that a point at infinity is its own inverse. Hence, the three points at infinity (i.e., on the line $Z = 0$ in $\mathbb{P}^2$) — namely, $(1 : 0 : 0)$, $(0 : 1 : 0)$ and $(a : b : 0)$, are exactly the three primitive 2-torsion points of E. The sum of any two of them is equal to the third

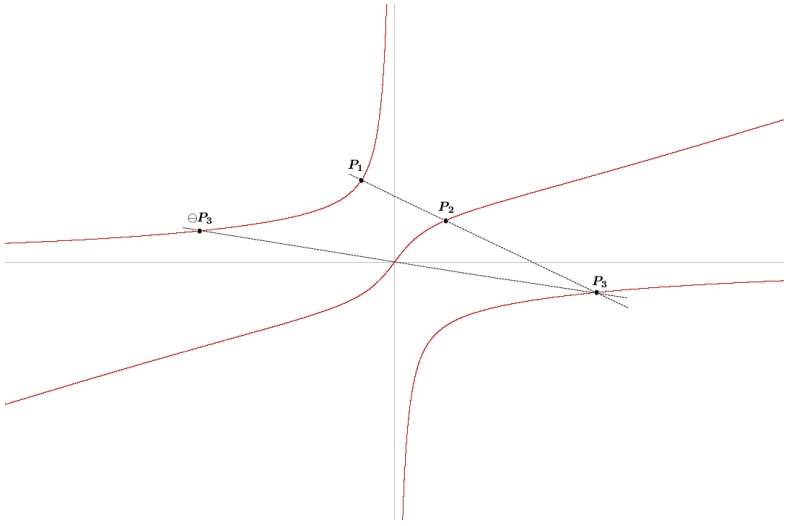


Figure 8.1: Example of a Huff curve (over $\mathbb{R}$).

one. More generally, $(X_1 : Y_1 : Z_1) \oplus (1 : 0 : 0)$ is the inverse of the point of intersection of the "horizontal" line passing through $(X_1 : Y_1 : Z_1)$ with E. When $Z_1 \neq 0$, we have

$$(X_1 : Y_1 : Z_1) \oplus (1 : 0 : 0) = ({Z_1}^2 : -X_1Y_1 : X_1Z_1)\,,$$

and analogously,

$$(X_1 : Y_1 : Z_1) \oplus (0 : 1 : 0) = (-X_1Y_1 : {Z_1}^2 : Y_1Z_1)\ .$$

From $(a : b : 0) = (1 : 0 : 0) \oplus (0 : 1 : 0)$, when $Z_1 \neq 0$, we get $(X_1 : Y_1 : Z_1) + (a : b : 0) = ({Z_1}^2 : -X_1Y_1 : X_1Z_1) \oplus (0 : 1 : 0)$ and therefore

$$(X_1 : Y_1 : Z_1) \oplus (a : b : 0) = \begin{cases} (a : b : 0) & \text{if } (X_1 : Y_1 : Z_1) = (0 : 0 : 1) \\ (Y_1Z_1 : X_1Z_1 : -X_1Y_1) & \text{otherwise} \end{cases}\ .$$

We remark that adding $(a : b : 0)$ to any of the points $(\pm 1 : \pm 1 : 1)$ transforms it into its inverse. It follows that these four points are the four solutions to the equation $[2]\mathbf{P} = (a : b : 0)$ and so are primitive 4-torsion points. The eight remarkable points we identified form a subgroup isomorphic to $\mathbb{Z}/4\mathbb{Z} \times \mathbb{Z}/2\mathbb{Z}$. When $\mathbb{K} = \mathbb{Q}$, this must be the full torsion since, according to a theorem by Mazur, the torsion subgroup is of order at most 12 (and thus exactly 8 here).

Remark 8.1. In [Huf48, p. 445], it is noted that the inverse projective transformations

$$\begin{aligned} \Upsilon : \mathbb{P}^2(\mathbb{K}) &\to \mathbb{P}^2(\mathbb{K}) : \\ (X : Y : Z) &\mapsto (U : V : W) = \big(ab(bX - aY) : ab(b^2 - a^2)Z : -aX + bY\big) \end{aligned}$$

and

$$\Upsilon^{-1} : \mathbb{P}^2(\mathbb{K}) \to \mathbb{P}^2(\mathbb{K}) : \\ (U : V : W) \mapsto (X : Y : Z) = \big(b(U + a^2W) : a(U + b^2W) : V\big)$$

induce a correspondence between equation (8.2) and the Weierstrass equation

$$V^2W = U(U + a^2W)(U + b^2W) \ .$$

Observe that point at infinity $(0 : 1 : 0)$ on the Weierstrass curve is mapped to $(0 : 0 : 1)$ on the Huff curve through Υ^{-1}. Observe also that map Υ^{-1} is a line-preserving transformation. This is another way to see that the group law on a Huff curve E follows the chord-and-tangent rule [Sil86, §2] with $\mathbf{O} = (0 : 0 : 1)$ as neutral element.

8.2.1 Affine formulas

We give explicit formulas for the group law. Excluding the 2-torsion, we use the non-homogeneous form $ax(y^2 - 1) = by(x^2 - 1)$. Let $y = \lambda x + \mu$ denote the secant line passing through two different points $\mathbf{P}_1 = (x_1, y_1)$ and $\mathbf{P}_2 = (x_2, y_2)$. This line intersects the curve at a third point $\ominus\mathbf{P}_3 = (-x_3, -y_3)$. Plugging the line equation into the curve equation, we get

$$ax\big((\lambda x + \mu)^2 - 1\big) = b(\lambda x + \mu)(x^2 - 1) \implies \lambda(a\lambda - b)x^3 + \mu(2a\lambda - b)x^2 + \cdots = 0 \ .$$

Whenever defined, we so obtain

$$\begin{cases} x_3 = x_1 + x_2 + \dfrac{\mu(2a\lambda - b)}{\lambda(a\lambda - b)} \\ y_3 = \lambda x_3 - \mu \end{cases}$$

with $\lambda = \dfrac{y_1 - y_2}{x_1 - x_2}$ and $\mu = y_1 - \lambda x_1$. After simplification, we have

$$\begin{aligned} x_3 &= x_1 + x_2 + \frac{(x_1y_2 - x_2y_1)\big(2a(y_1 - y_2) - b(x_1 - x_2)\big)}{(y_1 - y_2)\big(a(y_1 - y_2) - b(x_1 - x_2)\big)} \\ &= \frac{(x_1 - x_2)\big(a({y_1}^2 - {y_2}^2) - b(x_1y_1 - x_2y_2)\big)}{(y_1 - y_2)\big(a(y_1 - y_2) - b(x_1 - x_2)\big)} \end{aligned}$$

and

$$y_3 = -\frac{(y_1 - y_2)\big(b({x_1}^2 - {x_2}^2) - a(x_1y_1 - x_2y_2)\big)}{(x_1 - x_2)\big(a(y_1 - y_2) - b(x_1 - x_2)\big)} \ .$$

The above formulas can be further simplified by reusing the curve equation. A simple calculation shows that

$$\big(a(y_1 - y_2) - b(x_1 - x_2)\big)(x_1 + x_2)y_1y_2 = a(x_2y_1 - x_1y_2)(y_1y_2 - 1) \ .$$

Hence, we can write

$$\begin{aligned} x_3 &= x_1 + x_2 - \frac{\big(2a(y_1 - y_2) - b(x_1 - x_2)\big)(x_1 + x_2)y_1y_2}{(y_1 - y_2)a(y_1y_2 - 1)} \\ &= x_1 + x_2 - \frac{x_2y_1 - x_1y_2}{y_1 - y_2} - \frac{(x_1 + x_2)y_1y_2}{y_1y_2 - 1} \\ &= \frac{x_1y_1 - x_2y_2}{y_1 - y_2} - \frac{(x_1 + x_2)y_1y_2}{y_1y_2 - 1} \ . \end{aligned}$$

Furthermore, as easily shown

$$b(x_1y_1 - x_2y_2)(x_1x_2 + 1) = (y_1 - y_2)\big(ax_1x_2(y_1 + y_2) + b(x_1 + x_2)\big) \, ,$$

it thus follows that

$$\begin{aligned} x_3 &= \frac{ax_1x_2(y_1 + y_2) + b(x_1 + x_2)}{b(x_1x_2 + 1)} - \frac{(x_1 + x_2)y_1y_2}{y_1y_2 - 1} \\ &= \frac{(x_1 + x_2)(1 + y_1y_2)}{(1 + x_1x_2)(1 - y_1y_2)} \ , \end{aligned} \tag{8.3}$$

since $ax_1x_2(y_1 + y_2)(1 - y_1y_2) = by_1y_2(x_1 + x_2)(1 - x_1x_2)$.

Likewise, by symmetry, we have

$$y_3 = \frac{(y_1 + y_2)(1 + x_1x_2)}{(1 - x_1x_2)(1 + y_1y_2)} \ . \tag{8.4}$$

Equations (8.3) and (8.4) are defined whenever $x_1x_2 \neq \pm 1$ and $y_1y_2 \neq \pm 1$. Advantageously, curve parameters are not involved. Moreover, this addition law is *unified*: it can be used to double a point (i.e., when $\mathbf{P}_2 = \mathbf{P}_1$).

8.2.2 Projective formulas

Previous affine formulas involve inversions in $\mathbb{K}$. To avoid these operations and get faster arithmetic, projective coordinates may be preferred.

We let m and s represent the cost of a multiplication and of a squaring in $\mathbb{K}$, respectively. The projective form of (8.3) and (8.4) is

$$\begin{cases} X_3 = (X_1Z_2 + X_2Z_1)(Y_1Y_2 + Z_1Z_2)^2(Z_1Z_2 - X_1X_2) \\ Y_3 = (Y_1Z_2 + Y_2Z_1)(X_1X_2 + Z_1Z_2)^2(Z_1Z_2 - Y_1Y_2) \\ Z_3 = ({Z_1}^2{Z_2}^2 - {X_1}^2{X_2}^2)({Z_1}^2{Z_2}^2 - {Y_1}^2{Y_2}^2) \end{cases} . \tag{8.5}$$

In more detail, this can be evaluated as

$$\begin{gathered} m_1 = X_1X_2, \ \ m_2 = Y_1Y_2, \ \ m_3 = Z_1Z_2, \\ m_4 = (X_1 + Z_1)(X_2 + Z_2) - m_1 - m_3, \ \ m_5 = (Y_1 + Z_1)(Y_2 + Z_2) - m_2 - m_3, \\ m_6 = (m_2 + m_3)(m_3 - m_1), \ \ m_7 = (m_1 + m_3)(m_3 - m_2), \\ m_8 = m_4(m_2 + m_3), \ \ m_9 = m_5(m_1 + m_3), \\ X_3 = m_8m_6, \ \ Y_3 = m_9m_7, \ \ Z_3 = m_6m_7, \end{gathered}$$

that is, with $\underline{12\mathsf{m}}$.

8.2.3 Applicability

If $(x_1, y_1) \neq (0, 0)$ then $(x_1, y_1) \oplus (a : b : 0) = -(\frac{1}{x_1}, \frac{1}{y_1})$. Observe that Equation (8.5) remains valid for doubling point $(a : b : 0)$ or for adding point $(a : b : 0)$ to another finite point (i.e., which is not at infinity) different from $\mathbf{O}$; we get $(X_1 : Y_1 : Z_1) \oplus (a : b : 0) = (-Y_1 Z_1 : -X_1 Z_1 : X_1 Y_1)$ as expected. The addition formula is however not valid for adding $(0 : 1 : 0)$ or $(1 : 0 : 0)$. More generally, we have:

Theorem 8.1. *Let $\mathbb{K}$ be a field of characteristic $\neq 2$. Let $\mathbf{P}_1 = (X_1 : Y_1 : Z_1)$ and $\mathbf{P}_2 = (X_2 : Y_2 : Z_2)$ be two points on a Huff curve over $\mathbb{K}$. Then the addition formula given by (8.5) is valid provided that $X_1 X_2 \neq \pm Z_1 Z_2$ and $Y_1 Y_2 \neq \pm Z_1 Z_2$.*

Proof. If $\mathbf{P}_1$ and $\mathbf{P}_2$ are finite, we can write $\mathbf{P}_1 = (x_1, y_1)$ and $\mathbf{P}_2 = (x_2, y_2)$. The above affine formula for (x_3, y_3) as given by (8.3) and (8.4) is defined whenever $x_1 x_2 \neq \pm 1$ and $y_1 y_2 \neq \pm 1$. This translates into $X_1 X_2 \neq \pm Z_1 Z_2$ and $Y_1 Y_2 \neq \pm Z_1 Z_2$ for their projective coordinates.

It remains to analyze points at infinity. The points with their Z-coordinate equal to 0 are $(1 : 0 : 0)$, $(0 : 1 : 0)$ and $(a : b : 0)$. If $\mathbf{P}_1$ or $\mathbf{P}_2 \in \{(1 : 0 : 0), (0 : 1 : 0)\}$, the condition $X_1 X_2 \neq \pm Z_1 Z_2$ and $Y_1 Y_2 \neq \pm Z_1 Z_2$ is not satisfied. Suppose now $\mathbf{P}_2 = (a : b : 0)$. The condition becomes $X_1 \neq 0$ and $Y_1 \neq 0$, which corresponds to $\mathbf{P}_1 \notin \{\mathbf{O}, (1 : 0 : 0), (0 : 1 : 0)\}$. As aforementioned, the addition law is then valid for adding $\mathbf{P}_1$ to $(a : b : 0)$. □

The previous theorem says that the addition on a Huff curve is almost complete. However, the exceptional inputs are easily prevented in practice. Cryptographic applications typically involve (large) prime-order subgroups. More specifically, we state:

Corollary 8.1. *Let E be a Huff curve over a field $\mathbb{K}$ of odd characteristic. Let also $\mathbf{P} \in E(\mathbb{K})$ be a point of odd order. Then the addition law in the subgroup generated by $\mathbf{P}$ is complete.*

Proof. All points in $\langle \mathbf{P} \rangle$ are of odd order and thus are finite (remember that points at infinity are of order 2). It remains to show that for any points $\mathbf{P}_1 = (x_1, y_1), \mathbf{P}_2 = (x_2, y_2) \in \langle \mathbf{P} \rangle$, we have $x_1 x_2 \neq \pm 1$ and $y_1 y_2 \neq \pm 1$. Note that $x_1, y_1, x_2, y_2 \neq \pm 1$ since this corresponds to points of order 4 (and thus not in $\langle \mathbf{P} \rangle$). Suppose that $x_1 x_2 = \pm 1$. Then $a x_1({y_1}^2 - 1) = b y_1({x_1}^2 - 1) \implies a \frac{1}{x_1}({y_1}^2 - 1) = b y_1(1 - \frac{1}{{x_1}^2}) \implies \pm a x_2({y_1}^2 - 1) = -b y_1({x_2}^2 - 1)$. Hence, since $a x_2({y_2}^2 - 1) = b y_2({x_2}^2 - 1)$, it follows that $\mp y_2({y_1}^2 - 1) = y_1({y_2}^2 - 1) \implies (y_1 \pm y_2)(1 \mp y_1 y_2) = 0 \implies y_2 = \mp y_1$ or $y_1 y_2 = \pm 1$. As a result, when $x_1 x_2 = \pm 1$, we have $(x_2, y_2) \in \{(\frac{1}{x_1}, -y_1), (\frac{1}{x_1}, \frac{1}{y_1}), (-\frac{1}{x_1}, y_1), (-\frac{1}{x_1}, -\frac{1}{y_1})\}$. In all cases, one of $(x_1, y_1) \oplus (x_2, y_2)$ or $(x_1, y_1) \ominus (x_2, y_2)$ is a 2-torsion point, a contradiction. Likewise, it can be verified that the case $y_1 y_2 = \pm 1$ leads to a contradiction, which concludes the proof. □

The *completeness* of the addition law is very useful as it yields a natural protection against certain side-channel attacks (e.g., see [BSS05]). Another useful feature is that the addition law is *independent* of the curve parameters.

8.2.4 Universality of the model

The next theorem states that every elliptic curve over a field of characteristic $\neq 2$ containing a copy of $\mathbb{Z}/4\mathbb{Z} \times \mathbb{Z}/2\mathbb{Z}$ can be put in Huff's form. Generalizations and extensions are discussed in §8.3.

Theorem 8.2. *Any elliptic curve $(E, \mathbf{O})$ over a perfect field $\mathbb{K}$ of characteristic $\neq 2$ such that $E(\mathbb{K})$ contains a subgroup $\mathbb{G}$ isomorphic to $\mathbb{Z}/4\mathbb{Z} \times \mathbb{Z}/2\mathbb{Z}$ is birationally equivalent over $\mathbb{K}$ to a Huff curve.*

Proof. The Riemann-Roch theorem implies that if $\mathbf{D} = a_1\mathbf{P}_1 + \cdots + a_r\mathbf{P}_r$ is a divisor of degree 0 on E then the dimension of the vector space

$$\mathscr{L}(\mathbf{D}) = \{f \in \mathbb{K}(E)^\times \mid \operatorname{div}(f) \geqslant -\mathbf{D}\} \cup \{0\}$$

is equal to 1 when $a_1\mathbf{P}_1 \oplus \cdots \oplus a_r\mathbf{P}_r = \mathbf{O}$, and to 0 otherwise.

Let $\mathbf{H}_{++}, \mathbf{H}_{+-}, \mathbf{H}_{-+}$ and $\mathbf{H}_{--}$ denote the four points of $\mathbb{G}$ of order exactly 4 (with the convention $\mathbf{H}_{++} \oplus \mathbf{H}_{--} = \mathbf{O}$). Doubling these points produces a unique primitive 2-torsion point that we denote $\mathbf{R}$. We further let $\mathbf{P}$ and $\mathbf{Q}$ denote the other two 2-torsion points; say, $\mathbf{P} = \ominus\mathbf{H}_{++} \oplus \mathbf{H}_{+-}$ and $\mathbf{Q} = \mathbf{H}_{++} \oplus \mathbf{H}_{+-}$. We have $\mathbf{P} \oplus \mathbf{R} \ominus \mathbf{Q} \ominus \mathbf{O} = \mathbf{O}$; so there exists a nonzero rational function x with divisor exactly $\mathbf{Q} + \mathbf{O} - \mathbf{P} - \mathbf{R}$. In particular, x is well-defined and nonzero at $\mathbf{H}_{++}$ and thus without loss of generality we may assume that $x(\mathbf{H}_{++}) = 1$. Similarly, there exists a rational function y with divisor $\mathbf{P} + \mathbf{O} - \mathbf{Q} - \mathbf{R}$ such that $y(\mathbf{H}_{++}) = 1$.

The rational function $x - 1$ has the same poles as x and vanishes at $\mathbf{H}_{++}$. Its divisor $\operatorname{div}(x-1)$ is thus given by $\mathbf{H}_{++} + \mathbf{X} - \mathbf{P} - \mathbf{R}$ for some point $\mathbf{X}$. Since this divisor is principal, we have $\mathbf{H}_{++} \oplus \mathbf{X} \ominus \mathbf{P} \ominus \mathbf{R} = \mathbf{O}$. Hence, it follows that $\mathbf{X} = \mathbf{P} \oplus \mathbf{R} \ominus \mathbf{H}_{++} = \ominus\mathbf{H}_{++} \oplus \mathbf{H}_{+-} \oplus \mathbf{R} \ominus \mathbf{H}_{++} = \mathbf{H}_{+-}$. Consequently, we have $x(\mathbf{H}_{+-}) = 1$. Likewise, it is verified that $y(\mathbf{H}_{-+}) = 1$.

Now, consider the map ι taking a rational function f to $\iota f : \mathbf{M} \mapsto f(\ominus\mathbf{M})$. This is an endomorphism of the vector space $\mathscr{L}(\mathbf{P}+\mathbf{R}-\mathbf{Q}-\mathbf{O})$. Indeed, the poles of ιf are $\ominus\mathbf{P} = \mathbf{P}$ and $\ominus\mathbf{R} = \mathbf{R}$ and its zeros are $\ominus\mathbf{Q} = \mathbf{Q}$ and $\ominus\mathbf{O} = \mathbf{O}$. Moreover, since $\iota^2 = \mathrm{id}$ and since $\mathscr{L}(\mathbf{P} + \mathbf{R} - \mathbf{Q} - \mathbf{O})$ is a one-dimensional vector space, ι is the multiplication map by 1 or -1. The equality $\iota x = x$ would imply $x(\mathbf{H}_{--}) = x(\mathbf{H}_{++}) = 1$, which contradicts the previous calculation of $\operatorname{div}(x-1)$. As a result, we must have $\iota x = -x$. In particular, noting that $\mathbf{H}_{-+} = \ominus\mathbf{H}_{+-}$, we obtain

$$x(\mathbf{H}_{-+}) = \iota x(\mathbf{H}_{+-}) = -x(\mathbf{H}_{+-}) = -1\,,$$

and similarly for $\mathbf{H}_{--}$. Since $x + 1$ has the same poles as x, its divisor is then given by $\operatorname{div}(x+1) = \mathbf{H}_{-+}+\mathbf{H}_{--}-\mathbf{P}-\mathbf{R}$. Analogously, we obtain $\operatorname{div}(y+1) = \mathbf{H}_{+-}+\mathbf{H}_{--}-\mathbf{Q}-\mathbf{R}$.

Finally, consider the rational functions $u = x(y^2-1)$ and $v = y(x^2-1)$. We have:

$$\begin{aligned}\operatorname{div}(u) &= \operatorname{div}(x) + \operatorname{div}(y-1) + \operatorname{div}(y+1)\\ &= (\mathbf{Q} + \mathbf{O} - \mathbf{P} - \mathbf{R}) + (\mathbf{H}_{++} + \mathbf{H}_{-+} - \mathbf{Q} - \mathbf{R}) + (\mathbf{H}_{+-} + \mathbf{H}_{--} - \mathbf{Q} - \mathbf{R})\\ &= \mathbf{H}_{++} + \mathbf{H}_{+-} + \mathbf{H}_{-+} + \mathbf{H}_{--} + \mathbf{O} - \mathbf{P} - \mathbf{Q} - 3\mathbf{R}\end{aligned}$$

and

$$\begin{aligned}\operatorname{div}(v) &= \operatorname{div}(y) + \operatorname{div}(x-1) + \operatorname{div}(x+1)\\ &= (\mathbf{P}+\mathbf{O}-\mathbf{Q}-\mathbf{R}) + (\mathbf{H}_{++}+\mathbf{H}_{+-}-\mathbf{P}-\mathbf{R}) + (\mathbf{H}_{-+}+\mathbf{H}_{--}-\mathbf{P}-\mathbf{R})\\ &= \mathbf{H}_{++}+\mathbf{H}_{+-}+\mathbf{H}_{-+}+\mathbf{H}_{--}+\mathbf{O}-\mathbf{P}-\mathbf{Q}-3\mathbf{R}\ .\end{aligned}$$

But the vector space $\mathscr{L}(\mathbf{P}+\mathbf{Q}+3\mathbf{R}-\mathbf{O}-\mathbf{H}_{++}-\mathbf{H}_{+-}-\mathbf{H}_{-+}-\mathbf{H}_{--})$ is of dimension 1, so there exists a linear relation between u and v. In other words, there exist $a, b \in \mathbb{K}^\times$ such that $au = bv$; i.e., such that $ax(y^2-1) = by(x^2-1)$.

The rational map $E \to \mathbb{P}^2(\mathbb{K})$ given by $\mathbf{M} \mapsto (x(\mathbf{M}) : y(\mathbf{M}) : 1)$ extends to a morphism defined on all of E, and its image is contained in $E_{a,b}$ in view of the previous relation (and $E_{a,b}$ itself is a smooth irreducible curve as seen in §8.1.1). We therefore have a non-constant —and hence surjective— morphism of curves $E \to E_{a,b}$. Moreover, its degree is at most 1: indeed, if a point $(x_0 : y_0 : 1) \in E_{a,b}(\overline{\mathbb{K}})$ has two distinct preimages $\mathbf{M} \neq \mathbf{M'} \in E(\overline{\mathbb{K}})$, the functions $x - x_0$ and $y - y_0$ vanish at $\mathbf{M}$ and $\mathbf{M'}$. Since they have the same poles as x and y, their divisors are respectively $\mathbf{M}+\mathbf{M'}-\mathbf{P}-\mathbf{R}$ and $\mathbf{M}+\mathbf{M'}-\mathbf{Q}-\mathbf{R}$, which yields $\mathbf{P}\oplus\mathbf{R} = \mathbf{M}\oplus\mathbf{M'} = \mathbf{Q}\oplus\mathbf{R}$, a contradiction. As a surjective morphism of degree 1, the map $E \to E_{a,b}$ is thus an isomorphism. □

8.3 Generalizations and Extensions

This section presents dedicated addition formulas. It also presents a generalization of the model as originally introduced by Huff so that it covers more curves and extends to binary fields.

8.3.1 Faster computations

Dedicated doubling. The doubling formula can be sped up by evaluating squarings in $\mathbb{K}$ with a specialized implementation. The cost of a point doubling then becomes $\underline{7\mathsf{m}+5\mathsf{s}}$. When $\mathsf{s} > \frac{3}{4}\mathsf{m}$, an even faster way for doubling a point is given by

$$\begin{gathered}m_1 = X_1Y_1,\ \ m_2 = X_1Z_1,\ \ m_3 = Y_1Z_1,\ \ s_1 = {Z_1}^2,\\ m_4 = (m_2-m_3)(m_2+m_3),\ \ m_5 = (m_1-s_1)(m_1+s_1),\\ m_6 = (m_1-s_1)(m_2-m_3),\ \ m_7 = (m_1+s_1)(m_2+m_3),\\ X([2]\mathbf{P}_1) = (m_6-m_7)(m_4+m_5),\ \ Y([2]\mathbf{P}_1) = (m_6+m_7)(m_4-m_5),\\ Z([2]\mathbf{P}_1) = (m_4+m_5)(m_4-m_5),\end{gathered}$$

that is, with $\underline{10\mathsf{m}+1\mathsf{s}}$.

Moving the origin. Choosing $\mathbf{O'} = (0 : 1 : 0)$ as the neutral element results in translating the group law. If we let $\oplus'$ denote the corresponding point addition, we have $\mathbf{P}_1 \oplus' \mathbf{P}_2 = (\mathbf{P}_1 \ominus \mathbf{O'}) \oplus (\mathbf{P}_2 \ominus \mathbf{O'}) \oplus \mathbf{O'} = \mathbf{P}_1 \oplus \mathbf{P}_2 \oplus \mathbf{O'}$. Hence, we get

$$\begin{cases}X_3 = (X_1Z_2+X_2Z_1)(Y_1Y_2+Z_1Z_2)(Y_1Z_2+Y_2Z_1)\\ Y_3 = (X_1X_2-Z_1Z_2)({Z_1}^2{Z_2}^2-{Y_1}^2{Y_2}^2)\\ Z_3 = (Y_1Z_2+Y_2Z_1)(X_1X_2+Z_1Z_2)(Y_1Y_2-Z_1Z_2)\end{cases}.$$

This can be evaluated with $\underline{11\mathsf{m}}$ as

$$\begin{gathered} m_1 = X_1X_2,\ \ m_2 = Y_1Y_2,\ \ m_3 = Z_1Z_2, \\ m_4 = (X_1+Z_1)(X_2+Z_2) - m_1 - m_3,\ \ m_5 = (Y_1+Z_1)(Y_2+Z_2) - m_2 - m_3, \\ X_3 = m_4(m_2+m_3)m_5,\ \ Y_3 = (m_1-m_3)(m_3-m_2)(m_3+m_2), \\ Z_3 = m_5(m_1+m_3)(m_2-m_3)\ . \end{gathered} \tag{8.6}$$

This addition formula is unified: it can be used for doubling as well.

For a mixed point addition (i.e., when $Z_2 = 1$), we have $m_3 = Z_1$ and the number of required multiplications drops to $\underline{10\mathsf{m}}$. When used for dedicated doubling, the above addition formula requires $\underline{6\mathsf{m}+5\mathsf{s}}$, which can equivalently be obtained as

$$\begin{gathered} s_1 = {X_1}^2,\ \ s_2 = {Y_1}^2,\ \ s_3 = {Z_1}^2, \\ s_4 = (X_1+Y_1)^2 - s_1 - s_2,\ \ s_5 = (Y_1+Z_1)^2 - s_2 - s_3, \\ X([2]\mathbf{P}_1) = 2s_3s_4(s_2+s_3),\ \ Y([2]\mathbf{P}_1) = (s_1-s_3)(s_3-s_2)(s_3+s_2), \\ Z([2]\mathbf{P}_1) = s_5(s_1+s_3)(s_2-s_3)\ . \end{gathered} \tag{8.7}$$

Note that the expression for the inverse of point $\mathbf{P}_1$ is unchanged: $\ominus'\mathbf{P}_1 = \ominus(\mathbf{P}_1 \ominus \mathbf{O}') \oplus \mathbf{O}' = \ominus\mathbf{P}_1 = (X_1 : Y_1 : -Z_1)$.

8.3.2 More formulas

Alternative addition formulas can be derived using the curve equation. For example, whenever defined, we can write $(x_3, y_3) = (x_1, y_1) \oplus (x_2, y_2)$ with

$$x_3 = \frac{(x_1-x_2)(y_1+y_2)}{(y_1-y_2)(1-x_1x_2)} \quad \text{and} \quad y_3 = \frac{(y_1-y_2)(x_1+x_2)}{(x_1-x_2)(1-y_1y_2)}\ .$$

In projective coordinates, this gives

$$\begin{cases} X_3 = (X_1Z_2 - X_2Z_1)^2(Y_1Z_2 + Y_2Z_1)(Z_1Z_2 - Y_1Y_2) \\ Y_3 = (Y_1Z_2 - Y_2Z_1)^2(X_1Z_2 + X_2Z_1)(Z_1Z_2 - X_1X_2) \\ Z_3 = (X_1Z_2 - X_2Z_1)(Y_1Z_2 - Y_2Z_1)(Z_1Z_2 - X_1X_2)(Z_1Z_2 - Y_1Y_2) \end{cases},$$

which can be evaluated with $13\mathsf{m}$ as

$$\begin{gathered} m_1 = X_1Z_2,\ \ m_2 = X_2Z_1,\ \ m_3 = Y_1Z_2,\ \ m_4 = Y_2Z_1, \\ m_5 = (Z_1-X_1)(Z_2+X_2) + m_1 - m_2,\ \ m_6 = (Z_1-Y_1)(Z_2+Y_2) + m_3 - m_4, \\ m_7 = (m_1-m_2)m_6,\ \ m_8 = (m_3-m_4)m_5, \\ X_3 = (m_1-m_2)(m_3+m_4)m_7,\ \ Y_3 = (m_1+m_2)(m_3-m_4)m_8,\ \ Z_3 = m_7m_8\ . \end{gathered}$$

Although not as efficient as the usual addition, this alternative formula is useful in some pairing computations (see §8.4.2).

8.3.3 Twisted curves

As shown in Theorem 8.1, the group of points of a Huff elliptic curve contains a copy of $\mathbb{Z}/4\mathbb{Z} \times \mathbb{Z}/2\mathbb{Z}$. This implies that the curve order is a multiple of 8. Several cryptographic standards, however, require elliptic curves with group order of the form $h\,n$ where $h \in \{1, 2, 3, 4\}$ and n is a prime.

We can generalize Huff's model to accommodate the case $h = 4$. Let $P \in \mathbb{K}[t]$ denote a monic polynomial of degree 2, with non-zero discriminant, and such that $P(0) \neq 0$. We can then introduce the cubic curve

$$axP(y) = byP(x)$$

where $a, b \in \mathbb{K}^\times$. The set of points $\{(0:0:1), (0:1:0), (1:0:0), (a:b:0)\} \cong \mathbb{Z}/2\mathbb{Z} \times \mathbb{Z}/2\mathbb{Z}$ belongs to the curve. Moreover, when P factors in $\mathbb{K}$ —i.e., when $P(t) = (t - \omega_1)(t - \omega_2)$ with $\omega_1, \omega_2 \in \mathbb{K}^\times$, the four points $(\pm\omega_1 : \pm\omega_2 : 1)$ are also on the curve.

When $\operatorname{Char}\mathbb{K} \neq 2$, we consider $P(t) = t^2 - d$ for some $d \in \mathbb{K}^\times$. So we deal with the set of projective points $(X : Y : Z) \in \mathbb{P}^2(\mathbb{K})$ satisfying the non-singular cubic equation

$$\hat{E}_d : aX(Y^2 - dZ^2) = bY(X^2 - dZ^2) \tag{8.8}$$

where $a, b, d \in \mathbb{K}^\times$ and $a^2 \neq b^2$. This equation corresponds to Weierstrass equation $V^2W = U(U + \frac{a^2}{d}W)(U + \frac{b^2}{d}W)$ under the inverse transformations $(X : Y : Z) = \big(b(dU + a^2W) : a(dU + b^2W) : dV\big)$ and $(U : V : W) = \big(ab(bX - aY) : ab(b^2 - a^2)Z : d(-aX + bY)\big)$. The transformation $(X : Y : Z) \leftarrow (X : Y : Z\sqrt{d})$ induces an isomorphism from $E = \hat{E}_1$ to $\hat{E}_d$ over $\mathbb{K}(\sqrt{d})$. Curves $\hat{E}_d$ are therefore *quadratic twists* of Huff curves.

In affine coordinates, we consider the curve equation $ax(y^2 - d) = by(x^2 - d)$. The sum of two finite points $\mathbf{P}_1 = (x_1, y_1)$ and $\mathbf{P}_2 = (x_2, y_2)$ such that $x_1x_2 \neq \pm d$ and $y_1y_2 \neq \pm d$ is given by (x_3, y_3) where

$$x_3 = \frac{d(x_1 + x_2)(d + y_1y_2)}{(d + x_1x_2)(d - y_1y_2)} \quad \text{and} \quad y_3 = \frac{d(y_1 + y_2)(d + x_1x_2)}{(d - x_1x_2)(d + y_1y_2)} \ . \tag{8.9}$$

Extending the computations of § 8.2.2, it is readily verified that the sum of two points can be evaluated with 12m (plus a couple of multiplications by constant d) using projective coordinates. The faster computations of the previous section also generalize to twisted curves.

8.3.4 Binary fields

Huff's form can be extended to a binary field as

$$ax(y^2 + y + 1) = by(x^2 + x + 1) \ .$$

This curve is birationally equivalent to Weierstrass curve

$$v(v + (a + b)u) = u(u + a^2)(u + b^2)$$

under the inverse maps

$$(x, y) = \left(\frac{b(u + a^2)}{v}, \frac{a(u + b^2)}{v + (a + b)u} \right) \quad \text{and} \quad (u, v) = \left(\frac{ab}{xy}, \frac{ab(axy + b)}{x^2 y} \right) \ .$$

The neutral element is $\mathbf{O} = (0, 0)$.

8.4 Pairings

8.4.1 Preliminaries

Let $(E, \mathbf{O})$ be an elliptic curve over $\mathbb{K} = \mathbb{F}_q$, with q odd. Suppose that $\#E(\mathbb{F}_q) = hn$ where n is a prime such that $\gcd(n, q) = 1$. Let further k denote the embedding degree with respect to n, namely the smallest extension $\mathbb{F}_{q^k}$ of $\mathbb{F}_q$ containing all n-th roots of unity. In other words, k is the smallest positive integer k such that $n \mid q^k - 1$. For better efficiency, we further assume that $k > 1$ is even.

For any point $\mathbf{P} \in E(\mathbb{F}_q)[n]$, we let $f_{\mathbf{P}}$ denote a rational function on E defined over $\mathbb{F}_q$ such that $\mathrm{div}(f_{\mathbf{P}}) = n\mathbf{P} - n\mathbf{O}$; it exists and is unique up to a multiplicative constant, according to the Riemann-Roch theorem. The group of n-th roots of unity in $\mathbb{F}_{q^k}$ is denoted by μ_n. The *(reduced) Tate pairing* is then defined as

$$T_n : E(\mathbb{F}_q)[n] \times E(\mathbb{F}_{q^k})/[n]E(\mathbb{F}_{q^k}) \to \mu_n : (\mathbf{P}, \mathbf{Q}) \mapsto f_{\mathbf{P}}(\mathbf{Q})^{(q^k-1)/n} \ .$$

This definition does not depend on the choice of $f_{\mathbf{P}}$ with the appropriate divisor, nor on the class of $\mathbf{Q}$ mod $[n]E(\mathbb{F}_{q^k})$.

In practice, T_n can be computed using a technique due to Miller [Mil04], in terms of rational functions $g_{\mathbf{R},\mathbf{P}}$ depending on $\mathbf{P}$ and on a variable point $\mathbf{R}$. Function $g_{\mathbf{R},\mathbf{P}}$ is the so-called *line function* with divisor $\mathbf{R} + \mathbf{P} - \mathbf{O} - (\mathbf{R} \oplus \mathbf{P})$, which arises in addition formulas when E is represented as a plane cubic. The core idea is to derive function $f_{\mathbf{P}}$ iteratively. Letting $f_{i,\mathbf{P}}$ be the function with divisor $\mathrm{div}(f_{i,\mathbf{P}}) = i\mathbf{P} - ([i]\mathbf{P}) - (i - 1)\mathbf{O}$, it is easily verified that

$$f_{i+j,\mathbf{P}} = f_{i,\mathbf{P}} \cdot f_{j,\mathbf{P}} \cdot g_{[i]\mathbf{P},[j]\mathbf{P}} \ .$$

Observe that $f_{1,\mathbf{P}} = 1$ and $f_{n,\mathbf{P}} = f_{\mathbf{P}}$. Hence, if $n = \overline{n_{\ell-1} n_{\ell-1} \cdots n_0}_2$ is the binary representation of n, the Tate pairing can be computed as follows.

Algorithm 8.1 Miller's algorithm.

1: $f \leftarrow 1$; $\mathbf{R} \leftarrow \mathbf{P}$
2: **for** $i = \ell - 2$ down to 0 **do**
3: $\quad f \leftarrow f^2 \cdot g_{\mathbf{R},\mathbf{R}}(\mathbf{Q})$; $\mathbf{R} \leftarrow [2]\mathbf{R}$
4: $\quad$ **if** $(n_i = 1)$ **then**
5: $\quad\quad f \leftarrow f \cdot g_{\mathbf{R},\mathbf{P}}(\mathbf{Q})$; $\mathbf{R} \leftarrow \mathbf{R} \oplus \mathbf{P}$
6: $\quad$ **end if**
7: **end for**
8: **return** $f^{(q^k-1)/n}$

Contrary to Edwards curves or Jacobi quartics, Huff curves are represented as plane cubics. This makes Miller's algorithm, along with a number of improvements proposed for Weierstrass curves (e.g., as presented in [BLS04a]), directly applicable to the computation of pairings over Huff curves.

8.4.2 Pairing formulas for Huff curves

Throughout the for-loop of Algorithm 8.1, the line function is always evaluated at the same point $\mathbf{Q} \in E(\mathbb{F}_{q^k}) \setminus E(\mathbb{F}_q)$. It is therefore customary to represent this point in affine coordinates. In our case, it is most convenient to choose the coordinates of $\mathbf{Q}$ as $\mathbf{Q} = (y, z) = (1 : y : z)$. Indeed, since the embedding degree k is even, the field $\mathbb{F}_{q^k}$ can be represented as $\mathbb{F}_{q^{k/2}}(\alpha)$, where α is any quadratic non-residue in $\mathbb{F}_{q^{k/2}}$. As a result, $\mathbf{Q}$ can be chosen of the form $\mathbf{Q} = (y_{\mathbf{Q}}, z_{\mathbf{Q}}\alpha)$ with $y_{\mathbf{Q}}, z_{\mathbf{Q}} \in \mathbb{F}_{q^{k/2}}$ [BLS03]. To do so, it suffices to pick a point on a quadratic twist of E over $\mathbb{F}_{q^{k/2}}$ and take its image under the isomorphism over $\mathbb{F}_{q^k}$.

Now, for any two points $\mathbf{R}$, $\mathbf{P}$ in $E(\mathbb{F}_q)$, let $\ell_{\mathbf{R},\mathbf{P}}$ denote the rational function vanishing on the line through $\mathbf{R}$ and $\mathbf{P}$. In general, we have

$$\ell_{\mathbf{R},\mathbf{P}}(\mathbf{Q}) = \frac{(zX_{\mathbf{P}} - Z_{\mathbf{P}}) - \lambda(yX_{\mathbf{P}} - Y_{\mathbf{P}})}{Y_{\mathbf{P}}}$$

where λ is the "(y, z)-slope" of the line through $\mathbf{R}$ and $\mathbf{P}$. Then, the divisor of $\ell_{\mathbf{R},\mathbf{P}}$ is

$$\mathrm{div}(\ell_{\mathbf{R},\mathbf{P}}) = \mathbf{R} + \mathbf{P} + \mathbf{T} - (1:0:0) - (0:1:0) - (a:b:0)$$

where $\mathbf{T}$ is the third point of intersection (counting multiplicities) of the line through $\mathbf{R}$ and $\mathbf{P}$ with the elliptic curve. In particular, if the neutral element of the group law $\oplus$ is denoted by $\mathbf{U}$, the line function $g_{\mathbf{R},\mathbf{P}}$ can be written as

$$g_{\mathbf{R},\mathbf{P}} = \frac{\ell_{\mathbf{R},\mathbf{P}}}{\ell_{\mathbf{R}\oplus\mathbf{P},\mathbf{U}}} \ .$$

We concentrate on the case when $\mathbf{U} = \mathbf{O} = (0:0:1)$. Then for any $\mathbf{Q} = (y_{\mathbf{Q}}, z_{\mathbf{Q}}\alpha)$, we have

$$\ell_{\mathbf{R}\oplus\mathbf{P},\mathbf{O}}(\mathbf{Q}) = y_{\mathbf{Q}} - \frac{Y_{\mathbf{R}\oplus\mathbf{P}}}{X_{\mathbf{R}\oplus\mathbf{P}}} \in \mathbb{F}_{q^{k/2}} \ .$$

Since this quantity lies in a proper subfield of $\mathbb{F}_{q^k}$, it goes to 1 after the final exponentiation in Miller's algorithm, which means that it can be discarded altogether. Similarly, divisions by $X_{\mathbf{P}}$ can be omitted, and denominators in the expression of λ can be canceled. In other words, if $\lambda = A/B$, we can compute the line function as

$$g_{\mathbf{R},\mathbf{P}}(\mathbf{Q}) = (zX_{\mathbf{P}} - Z_{\mathbf{P}}) \cdot B - (yX_{\mathbf{P}} - Y_{\mathbf{P}}) \cdot A$$

and get the required result.

We can now detail precise formulas for the addition and doubling steps in the so-called Miller loop (i.e., the main for-loop in Algorithm 8.1). We let M and S represent the cost of a multiplication and of a squaring in $\mathbb{F}_{q^k}$ while m and s are operations in $\mathbb{F}_q$ as before.

Addition step. In the case of addition, the (y, z)-slope of the line through $\mathbf{R} = (X_R : Y_R : Z_R)$ and $\mathbf{P} = (X_P : Y_P : Z_P)$ is

$$\lambda = \frac{Z_\mathbf{R} X_\mathbf{P} - Z_\mathbf{P} X_\mathbf{R}}{Y_\mathbf{R} X_\mathbf{P} - Y_\mathbf{P} X_\mathbf{R}} \ .$$

Therefore, the line function to be evaluated is of the form

$$g_{\mathbf{R},\mathbf{P}}(\mathbf{Q}) = (z_\mathbf{Q}\alpha \cdot X_\mathbf{P} - Z_\mathbf{P})(Y_\mathbf{R} X_\mathbf{P} - Y_\mathbf{P} X_\mathbf{R}) - (y_\mathbf{Q} \cdot X_\mathbf{P} - Y_\mathbf{P})(Z_\mathbf{R} X_\mathbf{P} - Z_\mathbf{P} X_\mathbf{R}) \ .$$

Since $\mathbf{P}$ and $\mathbf{Q}$ are constant throughout the loop, the values depending only on $\mathbf{P}$ and $\mathbf{Q}$ — in this case $y'_\mathbf{Q} = y_\mathbf{Q} \cdot X_\mathbf{P} - Y_\mathbf{P}$ and $z'_\mathbf{Q} = z_\mathbf{Q}\alpha \cdot X_\mathbf{P}$, can be precomputed.

Then, each Miller addition step requires computing $\mathbf{R} \oplus \mathbf{P}$ (one addition on the curve over $\mathbb{F}_q$), evaluating $g_{\mathbf{R},\mathbf{P}}(\mathbf{Q})$, and computing $f \cdot g_{\mathbf{R},\mathbf{P}}(\mathbf{Q})$ (one multiplication in the field $\mathbb{F}_{q^k}$).

We consider two types of Miller addition steps: full addition, for which no assumption is made on the representation of $\mathbf{P}$, and mixed addition, for which we further assume that $\mathbf{P}$ is given in affine coordinates (i.e., $X_\mathbf{P} = 1$). Both steps start with computing $\mathbf{R} \oplus \mathbf{P}$, including all intermediate results.

Full addition. Computing $\mathbf{R} \oplus \mathbf{P}$ requires $13\mathsf{m}$ using the dedicated addition formula from §8.3.1, including all intermediate results $m_1, \ldots, m_8$. Compute further $m_9 = (X_\mathbf{R} + Y_\mathbf{R})(X_\mathbf{P} - Y_\mathbf{P})$. We then have

$$g_{\mathbf{R},\mathbf{P}}(\mathbf{Q}) = (z'_\mathbf{Q} - Z_\mathbf{P})(m_9 + m_5 - m_6) - y'_\mathbf{Q}(m_1 - m_2)$$

where the first term requires $(\frac{k}{2} + 1)\mathsf{m}$ and the second term $\frac{k}{2}\mathsf{m}$. With the final multiplication over $\mathbb{F}_{q^k}$, the total cost of full addition is thus of $\underline{1\mathsf{M} + (k + 15)\mathsf{m}}$.

Mixed addition. Now that $X_\mathbf{P} = 1$, computing $\mathbf{R} \oplus \mathbf{P}$ using the formula from §8.2.2, including all the intermediate results $m_1, \ldots, m_9$, only requires $11\mathsf{m}$, since the computation of m_1 is free. We then have

$$g_{\mathbf{R},\mathbf{P}}(\mathbf{Q}) = (z'_\mathbf{Q} - Z_\mathbf{P})(Y_\mathbf{R} - Y_\mathbf{P} X_\mathbf{R}) - y'_\mathbf{Q}(2Z_\mathbf{R} - m_4)$$

where both terms require the same number of multiplications as before, plus one for $Y_\mathbf{P} X_\mathbf{R}$. The total cost of mixed addition is thus of $\underline{1\mathsf{M} + (k + 13)\mathsf{m}}$.

Doubling step. In the case of doubling, the (y, z)-slope of the tangent line at $\mathbf{R} = (X_R : Y_R : Z_R)$ is

$$\lambda = \frac{a(Z_\mathbf{R})^2 - 2bY_\mathbf{R} Z_\mathbf{R} - a(X_\mathbf{R})^2}{b(Y_\mathbf{R})^2 - 2aY_\mathbf{R} Z_\mathbf{R} - b(X_\mathbf{R})^2} = \frac{A}{B} \ .$$

Thus, the line function is of the form

$$g_{\mathbf{R},\mathbf{R}}(\mathbf{Q}) = z_\mathbf{Q}\alpha \cdot X_\mathbf{R} B - Z_\mathbf{R} B - y_\mathbf{Q} \cdot X_\mathbf{R} A + Y_\mathbf{R} A \ .$$

Miller's doubling involves computing the point $[2]\mathbf{R}$, which we do using the formulas from §8.2.2 in $7\mathsf{m} + 5\mathsf{s}$. Then the quantities A and B are obtained by computing the additional product $m_{10} = 2Y_{\mathbf{R}}Z_{\mathbf{R}} = (Y_{\mathbf{R}} + Z_{\mathbf{R}})^2 - m_2 - m_3$ using a single squaring. Computing $g_{\mathbf{R},\mathbf{R}}(\mathbf{Q})$ requires multiplying those two values by $X_{\mathbf{R}}$ and $Y_{\mathbf{R}}$ (resp. $X_{\mathbf{R}}$ and $Z_{\mathbf{R}}$), hence an additional $4\mathsf{m}$. And finally, multiplications by $y_{\mathbf{Q}}$ and $z_{\mathbf{Q}}\alpha$ both require $\frac{k}{2}\mathsf{m}$. Taking into account the multiplication and the squaring in $\mathbb{F}_{q^k}$ needed to complete the doubling step, the total cost of Miller doubling is thus of $\underline{1\mathsf{M} + 1\mathsf{S} + (k + 11)\mathsf{m} + 6\mathsf{s}}$.

8.5 Conclusion and Perspectives

This work introduced and studied Huff's model, a representation of elliptic curves to be considered alongside previous models such as Montgomery, Doche-Icart-Kohel and Edwards. This model provides efficient arithmetic, competitive with some of the fastest known implementations (although not quite as fast as "inverted Edwards" for now). Moreover, it has a number of additional desirable properties, including unified/complete addition laws and formulas that do not depend on curve parameters (both properties are useful in cryptographic applications to thwart certain implementation attacks). It is also suitable to other computations on elliptic curves, such as the evaluation of pairings.

Binary Huff curves, as introduced in §8.3.4, have been studied further by Devigne and Joye [DJ11]. They have proposed very efficient addition and doubling formulas, making Huff's model currently the most efficient representation for elliptic curve arithmetic in characteristic 2.

Additionally, further generalizations of Huff's model have been suggested in odd characteristic by Wu and Feng [WF10].

Part Two

Cryptanalysis of RSA-based Schemes

Overview

A year after the discovery of public-key cryptography by Diffie and Hellman in 1976 [Hel76], Rivest, Shamir and Adleman proposed the first example of a public-key cryptosystem [RSA78], which became known as RSA. Nowadays, the most widely used public-key cryptographic schemes, both for encryption and signature, are based on RSA and deployed in many everyday applications, from payment cards to secure web servers and virtual private networks.

Since its inception, the RSA function itself and the cryptographic protocols based on it have received a lot of attention from many researchers, and a number of attacks and vulnerabilities have been discovered, though, in the words of Boneh [Bon99], "none of them [were] devastating" for RSA-based cryptography at large. Rather, this large body of cryptanalytic work has shaped our understanding of the "proper way" to use the RSA function in cryptographic protocols, and underscored the dangers of incautious uses. It has also uncovered exploitable flaws in widespread systems and caused widely deployed standards to be withdrawn, but RSA as a whole remains robust.

Many different types of attacks have been considered. Here are a few examples.

Factoring algorithms: the most direct way to break RSA cryptosystems is certainly to factor the public modulus. As a result, there has arguably been more progress on the millennia-old problem of factoring integers in the last three decades than ever before. The recent factorization of the RSA-768 challenge modulus [KAF$^+$10] using the General Number Field Sieve (GNFS) [LJMP90, LL93], as developed by Adleman, Buhler, Lenstra, Pollard, Pomerance and others, puts in perspective the prediction by Martin Gardner in 1977 that his proposed 129-digit (426-bit) RSA modulus would be safe for 40 quadrillion years!

The RSA problem: while it not known whether a random instance of the RSA problem

is equivalent to factoring, some instances are known to be weaker. For example, Wiener [Wie90] observed that given an RSA public key (N, e), it is possible to factor N efficiently, and hence solve RSA, when the private exponent $d \equiv 1/e \pmod{N}$ is small ($d < (1/3) \cdot N^{1/4}$, a bound which was later improved to $d < N^{0.292}$ by Boneh and Durfee [BD00] using techniques introduced by Coppersmith [Cop97]). On a different note, the best known attack on the "one more RSA" problem, by Joux, Naccache and Thomé [JNT07], has a better heuristic asymptotic complexity than GNFS.

Special attack models: for example, even if the RSA problem is hard, RSA encryption may not remain one-way when the same message is encrypted for several different recipients. Clearly, if the same message is encrypted with the same small public exponent e to more than e recipients with textbook RSA encryption, an eavesdropping adversary can recover the message using a simple application of the Chinese Remainder Theorem. A similar but more sophisticated attack in this so-call *broadcast* model was proposed by Håstad [Hås88], who showed that if what is encrypted is not the message itself but a polynomial function of it different for all recipients, it is still possible to recover it.

Stronger security notions: the one-wayness of textbook RSA encryption under chosen-plaintext attack and the universal unforgeability of textbook RSA signatures under key-only attack are both equivalent to the RSA problem, but these are relatively weak security notions that do not capture such simple flaws as the *blinding attacks* deriving from the homomorphic properties of the RSA function. Many *padding* methods were proposed early on to thwart those simple attacks, but they usually did not come with clearly stated security guarantees, and many publications exhibited vulnerabilities in those *ad-hoc* schemes, such as [CCG+08, JNT07, CJNP00, Gri00]. Additionally, attacks like that of Bleichenbacher against SSL [Ble98] made it clear that strong security notions like CCA-security were not just purely theoretical constructs, as some may have believed at the time, but actually mattered in applications too.

Physical attacks: as embedded cryptographic devices such as smart cards became more common, it was observed that a real-world adversary does not interact with a mathematical algorithm in a black-box way, but can gain information from observing the physical process of the computation and from tampering with it. Key-recovery attacks on physical implementations of RSA, in particular, have been an active research area since the seminal timing and DPA attacks by Kocher et al. [Koc96, KJJ99] and the fault attacks of Boneh, DeMillo and Lipton [BDL01].

This part of the thesis is devoted to various contributions to the vast research area that is RSA cryptanalysis. Those contributions include several different "types" of attacks, and use different mathematical techniques. Chapter 9 describes the implementation of an existential forgery attack on the ad-hoc RSA signature paddings ISO/IEC 9796-2 [ISO9796–2:2002] and EMV [EMV], using sieving techniques reminiscent of algorithms

such as the quadratic sieve and index calculus. Chapters 10–11 present fault attacks on implementations of RSA signatures that use the Chinese Remainder Theorem; our main technical tool in the orthogonal lattice technique of Nguyen and Stern [NS97]. Chapter 12 describes new attacks on the ad-hoc RSA encryption padding PKCS#1 v1.5 [PKCS#1 v1.5], some of which are similar to Bleichenbacher's attack [Ble98], while others consider a broadcast model à la Håstad [Hås88] and are based on generalizations of the techniques of Coppersmith [Cop97]. Finally, Chapter 13 shows that certain RSA moduli of a special form introduced by Groth in [Gro05] can be factored faster than he suggested, making his proposed choice of parameters potentially insecure.

Contents

Chapter 9

Practical Cryptanalysis of ISO/IEC 9796-2 and EMV Signatures

9.1 Introduction

In 1999, Coron, Naccache and Stern [CNS99] discovered an existential signature forgery attack against two popular RSA signature standards, ISO/IEC 9796-1 [ISO9796–1] and ISO/IEC 9796-2 [ISO9796–2]. Following that attack, ISO/IEC 9796-1 was withdrawn and ISO/IEC 9796-2 was amended by imposing higher parameter size requirements that make Coron et al.'s attack infeasible in practice.

In this chapter, we describe a number of algorithmic improvements for the attack by Coron, Naccache and Stern, which provide a speed-up of several orders of magnitude and make it possible to attack the amended version of ISO/IEC 9796-2 [ISO9796–2:2002]. With these improvements, we were able to actually implement the attack on the Amazon EC2 cloud, and obtain a forgery in two days for a total cost of about US$800.

In response to this new attack, the ISO/IEC 9796-2 standard was amended again in late 2010 [ISO9796–2:2010].

Our algorithmic improvements also apply to attacking RSA signatures following the EMV banking industry specifications, which are based on ISO/IEC 9796-2 but in a more constrained format. We estimate that an EMV forgery would cost around US$45,000 on Amazon EC2.

This work was presented at CRYPTO 2009 [CNTW09].

9.1.1 Ad-hoc vs. provable RSA paddings

RSA [RSA78] is the first, and certainly the most popular public-key cryptosystem, both for encryption and signature. However, it has long been known that, in its textbook form, it has a number of security flaws, due in particular to its homomorphic properties.

For example, if two messages m_1, m_2 are signed with textbook RSA:

$$\sigma_1 = m_1^d \bmod N$$
$$\sigma_2 = m_2^d \bmod N$$

then $(\sigma_1 \cdot \sigma_2) \bmod N$ is a valid signature on $m_1 \cdot m_2$, which shows that textbook RSA signature is not unforgeable under chosen message attack.

As a result, to actually sign or encrypt with RSA, it is necessary to first "pad" the message using a certain encoding function μ before actually applying the RSA exponentiation. Thus, an RSA signature is really computed as:

$$\sigma = \mu(m)^d \bmod N.$$

We can roughly divide the RSA encoding functions μ in use nowadays in two categories:

Ad-hoc encodings are "handcrafted" to thwart certain classes of attacks. While still in use, ad-hoc encodings are now being phased out. PKCS#1 v1.5 [PKCS#1 v1.5], ISO/IEC 9796-1 [ISO9796–1] and ISO/IEC 9796-2 [ISO9796–2, ISO9796–2:2002] are typical examples of such encodings.

Provably secure encodings are designed to make cryptanalysis equivalent to inverting RSA (possibly in somewhat idealized attack models such as the Random Oracle Model [BR93]). OAEP [BR94] (for encryption) and PSS [BR96] (for signature) are typical examples of provably secure encodings.

For *ad-hoc* encodings, there is no guarantee that forging signatures is as hard as inverting RSA. And as a matter of fact, many such encodings were found to be weaker than the RSA problem. We refer the reader to [Ble98, CCG+08, JNT07, CJNP00, Gri00] for a few typical examples. It is thus a practitioner's rule of thumb to use provably secure encodings whenever possible. Nevertheless, ad-hoc encodings remain in widespread use in many commercial products (such as EMV payment cards). A periodic re-evaluation of such encodings is thus necessary.

9.1.2 The ISO/IEC 9796-2 encoding

ISO/IEC 9796-2 is an international standard that defines a specific, add-hoc encoding function for RSA signatures [ISO9796–2]. In [CNS99], Coron, Naccache and Stern discovered an attack against it that led to a revision of the standard.

More precisely, ISO/IEC 9796-2 can be used with hash functions of various digest sizes k_h. Originally, ISO/IEC 9796-2 recommended $128 \leqslant k_h \leqslant 160$, but $k_h = 128$ was within reach of the attack by Coron et al., so the standard was amended and the official requirement became $k_h \geqslant 160$ [ISO9796–2:2002].

The attack by Coron et al. is based on an earlier cryptanalytic result by Desmedt and Odlyzko [DO85], which was first presented as a chosen-ciphertext attack on RSA encryption, but applies to RSA signatures as well, as noted in [Mis98], provided that

the encoding function μ maps to integers of small size (much smaller than the modulus N). The ISO/IEC 9796-2 encoding itself does not satisfy this requirement (it maps to integers to full size), but Coron et al. showed that the Desmedt-Odlyzko attack can be adapted to this setting nonetheless.

9.1.3 Our contribution

In this chapter, we describe an improved attack against the amended version of ISO/IEC 9796-2, with $k_h = 160$. The new attack applies to EMV signatures as well, which are ISO/IEC 9796-2-compliant signatures with extra redundancy.

Our attack is based on the same idea as Coron et al.'s forgery, with a number of algorithmic refinements: a more careful choice of queried messages, a more efficient algorithm due to Bernstein for detecting smooth integers, the use of "large primes" as is common in modern implementations of the quadratic and number field sieves, and an optimized exhaustive search.

Using these refinements, we were able to compute an ISO/IEC 9796-2 forgery in two days on a few dozen nodes of the Amazon EC2 cloud, for a total cost of US$800. The forgery was implemented for public exponent $e = 2$ but attacking odd exponents would not take significantly longer[1]. We estimate that under similar conditions an EMV signature forgery would cost US$45,000. Note that all costs are per modulus. After computing a first forgery for a given N, additional forgeries come at a negligible cost.

9.2 The ISO/IEC 9796-2 Standard

ISO/IEC 9796-2 is an standard for RSA signatures that defines RSA encoding functions allowing partial or total message recovery [ISO9796–2, ISO9796–2:2002]. Here we consider only partial message recovery. As we have already mentioned, ISO/IEC 9796-2 can be used with hash functions $H(m)$ of various output sizes k_h. For the sake of simplicity we assume that k_h, the size of m and the size of N (denoted k) are all multiples of 8;[2] this is also the case in the EMV specifications. Then, the ISO/IEC 9796-2 encoding function has the following form:

$$\mu(m) = \mathtt{6A}_{16} \| m[1] \| H(m) \| \mathtt{BC}_{16}$$

where the message $m = m[1]\|m[2]$ is split in two: $m[1]$ consists of the $k - k_h - 16$ leftmost bits of m and $m[2]$ represents all the remaining bits of m. The size of $\mu(m)$ is therefore always $k - 1$ bits.

[1] It might slow down the final linear algebra stage of the attack somewhat, as sparse linear algebra in characteristic 2 can benefit from some bit fiddling optimizations on typical CPU architectures, but the linear algebra stage only takes a very small fraction of the overall CPU time anyway.

[2] As will be clarified later, if we drop the assumption that $k \equiv 0 \mod 8$, the attack actually becomes faster. Indeed, if m consists of an odd number of 4-bit nibbles, the encoding function becomes $\mu(m) = \mathtt{7}_{16} \| m[1] \| H(m) \| \mathtt{BC}_{16}$, with only 12 fixed bits instead of 16, which in turn makes the involved integers about 4 bits shorter. For example, the RSA-2048 $\{a, b\}$-pair (see §9.4.3) becomes $\{45, 28\}$ instead of $\{625, 332\}$.

The original version of the standard recommended $128 \leqslant k_h \leqslant 160$ for partial message recovery (see [ISO9796–2], §5, note 4). The amended versions of ISO/IEC 9796-2 [ISO9796–2:2002, ISO9796–2:2010] require $k_h \geqslant 160$. The EMV specifications also use $k_h = 160$.

Note that newer versions of ISO/IEC 9796-2 [ISO9796–2:2002, ISO9796–2:2010] also define and recommend a different encoding function known as ISO/IEC 9796-2 mode 3, which is essentially a deterministic variant of PSS-R [PKCS#1 v2.1]. This encoding function does not suffer from the vulnerabilities described in this chapter. However, its adoption and deployment remain unclear.

9.3 Previous Attacks

9.3.1 The Desmedt-Odlyzko forgery

In Desmedt and Odlyzko's attack [Mis98] (existential forgery under a chosen-message attack), the forger asks for the signature of messages of his choice before computing, by his own means, the signature of a (possibly meaningless) message that was never signed by the legitimate owner the signing key.

The attack only applies if $\mu(m)$ is much smaller than N and works as follows:

1. Select a bound B and let $\mathfrak{P} = \{p_1, \ldots, p_\ell\}$ be the list of all primes smaller than B.
2. Find $\tau \geqslant \ell + 1$ messages m_i such that each $\mu(m_i)$ is a product of primes in $\mathfrak{P}$.
3. Express one $\mu(m_j)$ as a multiplicative combination of the other $\mu(m_i)$'s, by solving a linear system given by the exponent vectors of the $\mu(m_i)$ with respect to the primes in $\mathfrak{P}$.
4. Ask for the signatures of the m_i for $i \neq j$ and forge the signature of m_j.

In the following we assume that e is prime; this includes $e = 2$. We say that an integer is B-smooth if all its prime factors are smaller than B. The integers $\mu(m_i)$ obtained at step 2 are therefore B-smooth and we can write for all messages m_i, $1 \leqslant i \leqslant \tau$:

$$\mu(m_i) = \prod_{j=1}^{\ell} p_j^{v_{i,j}}. \tag{9.1}$$

To each $\mu(m_i)$ we associate the ℓ-dimensional vector of the exponents modulo e:

$$\boldsymbol{V}_i = (v_{i,1} \bmod e, \ldots, v_{i,\ell} \bmod e).$$

Since e is prime, the set of all ℓ-dimensional vectors modulo e forms a linear space of dimension ℓ. Therefore, since $\tau \geqslant \ell + 1$, one can express one vector, say $\boldsymbol{V}_\tau$, as a linear combination of the others modulo e, which gives for $1 \leqslant j \leqslant \ell$:

$$\boldsymbol{V}_\tau = \boldsymbol{\Gamma} \cdot e + \sum_{i=1}^{\tau-1} \beta_i \boldsymbol{V}_i$$

for some $\mathbf{\Gamma} = (\gamma_1, \ldots, \gamma_\ell) \in \mathbb{Z}^\ell$. That is,

$$v_{\tau,j} = \gamma_j \cdot e + \sum_{i=1}^{\tau-1} \beta_i \cdot v_{i,j}.$$

Then using (9.1), one obtains:

$$\mu(m_\tau) = \prod_{j=1}^{\ell} p_j^{v_{\tau,j}} = \prod_{j=1}^{\ell} p_j^{\gamma_j \cdot e + \sum_{i=1}^{\tau-1} \beta_i \cdot v_{i,j}} = \left(\prod_{j=1}^{\ell} p_j^{\gamma_j} \right)^e \cdot \prod_{j=1}^{\ell} \prod_{i=1}^{\tau-1} p_j^{v_{i,j} \cdot \beta_i}$$

$$\mu(m_\tau) = \left(\prod_{j=1}^{\ell} p_j^{\gamma_j} \right)^e \cdot \prod_{i=1}^{\tau-1} \left(\prod_{j=1}^{\ell} p_j^{v_{i,j}} \right)^{\beta_i} = \left(\prod_{j=1}^{\ell} p_j^{\gamma_j} \right)^e \cdot \prod_{i=1}^{\tau-1} \mu(m_i)^{\beta_i}.$$

That is:

$$\mu(m_\tau) = \delta^e \cdot \prod_{i=1}^{\tau-1} \mu(m_i)^{\beta_i}, \quad \text{where we let} \quad \delta = \prod_{j=1}^{\ell} p_j^{\gamma_j}. \tag{9.2}$$

Therefore, we see that $\mu(m_\tau)$ can be written as a multiplicative combination of the other $\mu(m_i)$. To produce a forgery, the attacker will ask for the signatures of $m_1, \ldots, m_{\tau-1}$ and output the following valid signature on m_τ:

$$\sigma_\tau = \mu(m_\tau)^d = \delta \cdot \prod_{i=1}^{\tau-1} \left(\mu(m_i)^d \right)^{\beta_i} = \delta \cdot \prod_{i=1}^{\tau-1} \sigma_i^{\beta_i} \mod N.$$

Note that when $e = 2$, one uses with Rabin-Williams signatures rather than RSA signatures, and an additional component in the vectors $\boldsymbol{V}_i$ should be added to keep track of the Jacobi symbols of the messages involved, but apart from this minor modification, the attack works in exactly the same way.

The complexity of the attack depends on ℓ and on the probability that the integers $\mu(m_i)$ are B-smooth. But a "large" integer is very rarely B-smooth for a B corresponding to a manageable factor base $\mathfrak{P}$.

For example, the probability that a random 1024-bit integer is, say, 2^{25}-smooth, is less than 2^{-247}, so there is no hope to obtain a forgery in this way if the encoding function outputs random full size integers (as Full Domain Hash does, for example). On the other hand, a random 100-bit integer is 2^{25}-smooth with probability about 0.5%, so finding $\ell \approx 2^{25}/\log(2^{25}) \approx 2^{21}$ of them should be easy. As a result, the Desmedt-Odlyzko attack is feasible when the encoding function μ outputs integers of small size.

See [CND+06] for an asymptotic complexity analysis. In practice, the attack is feasible only if the $\mu(m_i)$ are smaller than around 200 bits.

9.3.2 The Coron-Naccache-Stern forgery

In ISO/IEC 9796-2, the encoding function's output $\mu(m)$ is as long as N. This thwarts Desmedt and Odlyzko's attack. However, Coron, Naccache and Stern observed [CNS99] that it is actually sufficient to construct messages m_i such that a fixed *multiple* t_i of

$\mu(m_i)$ modulo N is very small. Then, the attack can be applied to the integers t_i instead of $\mu(m_i)$.

More precisely, suppose that one can find a constant a and messages m_i such that for all i:

$$t_i = a \cdot \mu(m_i) \bmod N$$

is small. Then we can hope to find many smooth t_i's, and the attack can proceed as before. There is no problem in handling of the extra factor a in the multiplicative relations: one only needs to add a corresponding column in the matrix of exponents considered in §9.3.1 to keep track of it.

In their attack Coron et al. used $a = 2^8$. Then, to construct messages m such that $a \cdot \mu(m) \bmod N$ is small can be done as follows. From the definition of ISO/IEC 9796-2, we have:

$$\begin{array}{rcccccccc} \mu(m) & = & \mathtt{6A_{16}} & \| & m[1] & \| & H(m) & \| & \mathtt{BC_{16}} \\ & = & \mathtt{6A_{16}} \cdot 2^{k-8} & + & m[1] \cdot 2^{k_h+8} & + & H(m) \cdot 2^8 & + & \mathtt{BC_{16}} \end{array}$$

Euclidean division by N provides b and $0 \leqslant r < N < 2^k$ such that:

$$(\mathtt{6A_{16}} + 1) \cdot 2^k = b \cdot N + r$$

Denoting $N' = b \cdot N$ one can write:

$$\begin{array}{rcccc} N' & = & \mathtt{6A_{16}} \cdot 2^k & + & (2^k - r) \\ & = & \mathtt{6A_{16}} & \| & N'[1] \| N'[0] \end{array}$$

where N' is $k+7$ bits long and $N'[1]$ is $k - k_h - 16$ bits long.

Consider the linear combination:

$$\begin{array}{rcccc} t & = & b \cdot N & - & a \cdot \mu(m) \\ & = & N' & - & 2^8 \cdot \mu(m) \end{array}$$

By setting $m[1] = N'[1]$ we get:

$$\begin{array}{rcccccccl} t & = & \mathtt{6A_{16}} & \| & N'[1] & \| & N'[0] & & \\ & - & \mathtt{6A_{16}} & \| & m[1] & \| & H(m) \| \mathtt{BC00_{16}} & & \\ & = & \cancel{\mathtt{6A_{16}}} & \| & \cancel{N'[1]} & \| & N'[0] & & \\ & - & \cancel{\mathtt{6A_{16}}} & \| & \cancel{N'[1]} & \| & H(m) \| \mathtt{BC00_{16}} & & \\ & = & & & & & N'[0] - (H(m) \| \mathtt{BC00_{16}}) & < 2^{k_h+16} & \end{array}$$

For $k_h = 160$, the integer t is therefore at most 176-bits long.

The forger can thus modify $m[2]$ (and therefore $H(m)$), until he gets a set of messages whose t-values are B-smooth and express one such $\mu(m_\tau)$ as a multiplicative combination of the others. As per the analysis in [CNS99], attacking the instances $k_h = 128$ and $k_h = 160$ requires 2^{54} and 2^{61} operations respectively.

Note that the sign of the t_i's must be accounted for. This is simple because $(-1)^d \bmod N$ is public. Hence, when an odd number of t_i's is used a minus sign is inserted into δ.

9.4 Building Blocks of the New Attack

We improve the above complexities by using four new ideas: we speed up the process of finding smooth integers by using Bernstein's batch smoothness detection algorithm [Ber04a] instead of trial division; we also use the large prime variant [BP96]; moreover, we modify Coron et al.'s attack by selecting better messages and by optimizing exhaustive search to balance complexities. In this section we present these new building blocks.

9.4.1 Bernstein's smoothness detection algorithm

Bernstein [Ber04a] describes the following algorithm for finding smooth integers.

Algorithm: Given prime numbers $p_1, \ldots, p_\ell$ in increasing order and positive integers $t_1, \ldots, t_n$, output the p_ℓ-smooth part of each t_k:

1. Compute $z \leftarrow p_1 \times \cdots \times p_\ell$ using a product tree.
2. Compute $z_1 \leftarrow z \bmod t_1, \ldots, z_n \leftarrow z \bmod t_n$ using a remainder tree.
3. For each $k \in \{1, \ldots, n\}$: Compute $y_k \leftarrow (z_k)^{2^e} \bmod t_k$ by repeated squaring, where e is the smallest non-negative integer such that $2^{2^e} \geqslant t_k$.
4. For each $k \in \{1, \ldots, n\}$: output $\gcd(t_k, y_k)$.

We refer the reader to [Ber08] for a description of the product and remainder trees.

Theorem 9.1 (Bernstein)**.** *The algorithm computes the p_ℓ-smooth part of each integer t_k in $O(b \log^2 b \log\log b)$ time, where b is the number of input bits.*

In other words, given a list of n_t integers $t_i < 2^a$ and the list of the first ℓ primes, the algorithm will detect the B-smooth t_i's, where $B = p_\ell$, in time:

$$O(b \cdot \log^2 b \cdot \log\log b)$$

where $b = n_t \cdot a + \ell \cdot \log_2 \ell$ is the total number of input bits.

When n_t is very large, it becomes more efficient to run the algorithm k times, on batches of $n'_t = n_t/k$ integers. We explain in Appendix 9.A how to select the optimal n'_t, and derive the corresponding running time.

Bernstein recommends a number of speed-up ideas of which we used a few. In our experiments we used the *scaled remainder tree* [Ber04b], which replaces most division steps in the remainder tree by multiplications. This algorithm is fastest when FFT multiplications are done modulo numbers of the form $2^\alpha - 1$: we used this *Mersenne FFT multiplication* as well, as implemented in Gaudry, Kruppa and Zimmermann's GMP patch [GKZ07]. Other optimizations included computing the product z only once, and treating the prime 2 separately.

Bernstein's algorithm was actually the main speed up in our attack. It proved $\simeq 1000$ faster than the trial division used in [CNS99].

9.4.2 The large prime variant

An integer is *semi-smooth* with respect to y and z if its greatest prime factor is $\leqslant y$ and all other factors are $\leqslant z$. Bach and Peralta [BP96] define the function $\sigma(u, v)$, which plays for semi-smoothness the role played by Dickman's ρ function for smoothness [Dic30]: $\sigma(u, v)$ is the asymptotic probability that an integer n is semi-smooth with respect to $n^{1/v}$ and $n^{1/u}$.

After an integer t_i has had all its factors smaller than B stripped-off, if the remaining factor ω is lesser than B^2 then ω must be prime. This is very easy to detect using Bernstein's algorithm. As Bernstein computes the B-smooth part z_i of each t_i, it only remains to check whether t_i/z_i is small enough. In most cases it isn't even necessary to perform the actual division since comparing the sizes of t_i and z_i suffices to rule out most non-semi-smooth numbers.

Hence, one can use a second bound B_2 such that $B < B_2 < B^2$ and keep the t_i's whose remaining factor ω is $\leqslant B_2$, hoping to find a second t_i with the same remaining factor ω to divide ω out. We refer the reader to Appendix 9.B for a detailed analysis of the large prime variant in our context.

9.4.3 Constructing smaller $a \cdot \mu(m) - b \cdot N$ candidates

In this paragraph we show how to construct smaller $t_i = a \cdot \mu(m_i) - b \cdot N$ values for ISO/IEC 9796-2. Smaller t_i-values increase smoothness probability and hence speed up the forgery process. We write:

$$\mu(x, h) = \texttt{6A}_{16} \cdot 2^{k-8} + x \cdot 2^{k_h+8} + h \cdot 2^8 + \texttt{BC}_{16}$$

where $x = m[1]$ and $h = H(m)$, with $0 < x < 2^{k-k_h-16}$.

We first determine $a, b > 0$ such that the following two conditions hold:

$$0 < \quad b \cdot N - a \cdot \mu(0,0) \quad < a \cdot 2^{k-8} \tag{9.3}$$

$$b \cdot N - a \cdot \mu(0,0) \quad = 0 \mod 2^8 \tag{9.4}$$

and a is of minimal size. Then by Euclidean division we compute x and r such that:

$$b \cdot N - a \cdot \mu(0,0) = (a \cdot 2^{k_h+8}) \cdot x + r$$

where $0 \leqslant r < a \cdot 2^{k_h+8}$ and using (9.3) we have $0 \leqslant x < 2^{k-k_h-16}$ as required. This gives:

$$b \cdot N - a \cdot \mu(x,0) = b \cdot N - a \cdot \mu(0,0) - a \cdot x \cdot 2^{k_h+8} = r$$

Moreover as per (9.4) we must have $r = 0 \mod 2^8$; denoting $r' = r/2^8$ we obtain:

$$b \cdot N - a \cdot \mu(x,h) = r - a \cdot h \cdot 2^8 = 2^8 \cdot (r' - a \cdot h)$$

where $0 \leqslant r' < a \cdot 2^{k_h}$. We then look for smooth values of $r' - a \cdot h$, whose size is at most k_h plus the size of a.

Challenge	RSA-704	RSA-768	RSA-896	RSA-1024	RSA-1536	RSA-2048
a	481	251	775	311	581	625
b	228	132	412	172	316	332

Table 9.1: $\{a, b\}$ values for several RSA challenge moduli.

If a and b are both 8-bit integers, this gives 16 bits of freedom to satisfy both conditions (9.3) and (9.4); heuristically each of the two conditions is satisfied with probability $\simeq 2^{-8}$; therefore, we can expect to find an $\{a, b\}$ pair with a and b not much larger than 8 bits. For example, for the RSA-2048 challenge, we found $\{a, b\}$ to be $\{625, 332\}$; therefore, for RSA-2048 and $k_h = 160$, the integer to be smooth is 170-bits long (instead of 176-bits in Coron et al.'s original attack). This decreases the complexity of the attack further. The optimal $\{a, b\}$ values for other RSA challenge moduli are given in Table 9.1.

9.5 Attacking ISO/IEC 9796-2

We combined all the building-blocks listed in the previous section to compute an actual forgery for ISO/IEC 9796-2, with the RSA-2048 challenge modulus. The implementation replaced Coron et al.'s trial division by Bernstein's algorithm, replaced Coron et al.'s $a \cdot \mu(m) - b \cdot N$ values by the shorter t_i's introduced in §9.4.3 and took advantage of the large prime variant. Additional speed-up was obtained by exhaustive searching for particular digest values. Code was written in C++ and run on 19 Linux-based machines on the Amazon EC2 cloud. The final linear algebra step was performed on a single desktop PC.

9.5.1 The Amazon cloud

Amazon.com, Inc. offers virtualized computer instances for rent on a pay by the hour basis, which we found convenient to run our computations. Various models are available, of which the best-suited for CPU-intensive tasks, at the time when we carried out the attack, featured 8 Intel Xeon 64-bit cores clocked at 2.4 GHz supporting the Core2 instruction set and offering 7 GB RAM and 1.5 TB disk space. Renting such a capacity cost US$0.80 per hour (plus tax). One could run up to 20 such instances in parallel, and possibly more subject to approval by Amazon (20 were enough for our purpose so we didn't apply for more).

When an instance on the grid is launched, it starts up from a disk image containing a customizable UNIX operating system. In the experiment, we ran a first instance using the basic Fedora installation provided by default, installed necessary tools and libraries, compiled our own programs and made a disk image containing our code, to launch subsequent instances with. When an instance terminates, its disk space is freed, making it necessary to save results to some permanent storage means. We simply `rsync`'ed

results to a machine of ours. Note that Amazon also charges for network bandwidth but data transmission costs were negligible in our case.

All in all, we used about 1,100 instance running hours (including setup and tweaks) during a little more than two days. While we found the service to be rather reliable, one instance failed halfway through the computation, and its intermediate results were lost.

9.5.2 The experiment: Outline, details and results

The attack can be broken down into the following elementary steps, which we shall review in turn:

1. Determining the constants $a, b, x, \mu(x, 0)$ for the RSA-2048 challenge modulus N.
2. Computing the product of the first ℓ primes, for a suitable choice of ℓ.
3. Computing the integers $t_i = bN - a\mu(m_i)$, and hence the SHA-1 digests, for sufficiently many messages m_i.
4. Finding the smooth and semi-smooth integers amongst the t_i's.
5. Factoring the smooth integers, as well as the colliding pairs of semi-smooth integers, obtaining the sparse, singular matrix of exponents, with ℓ rows and more than ℓ columns.
6. Reducing this matrix modulo $e = 2$, with possible changes in the first row (corresponding to the prime 2) depending on the Jacobi symbols $(2|t_i)$ and $(2|a)$.
7. Finding nontrivial vectors in the kernel of this reduced matrix and inferring a forgery.

Steps 2–4 were executed on the Amazon EC2 grid, whereas all other steps were run on one offline PC. Steps 3–4, and to a much lesser extent Step 7, were the only steps that claimed a significant amount of CPU time.

Determining the constants. The cost of the attack doesn't depend on the choice of N. Since N has to be congruent to 5 mod 8 for Rabin-Williams signatures, we used the RSA-2048 challenge. The resulting constants were computed in SAGE [S+10b]. We found the smallest $\{a, b\}$ pair to be $\{625, 332\}$. The integers $t_i = bN - a\mu(x, h_i)$ are thus at most 170-bits long.

Product of the first primes. The optimal choice of ℓ for 170 bits is about 2^{21}. Since the Amazon instances are memory-constrained (less than 1 GB of RAM per core), we preferred to use $\ell = 2^{20}$. This choice had the additional advantage of making the final linear algebra step faster, which is convenient since this step was run on a single offline PC. Computing the product of primes itself was done once and for all in a matter of seconds using MPIR [H+09].

Hashing. Since the smoothness detection part works on batches of t_i's (in our cases, we chose batches of 2^{19} integers), we had to compute digests of messages m_i in batches as well. The messages themselves are 2048-bit long, i.e. as long as N, and we chose them in a fixed format facilitating the distributed computation: a constant 246-byte prefix followed by a 10-byte seed. The first two bytes identify a family of messages examined on a single core of one Amazon instance, and the remaining eight bytes are explored by increments of 1 starting from 0.

Messages were hashed using the SHA-1 implementation from OpenSSL. For each message, we only need to compute one SHA-1 block, since the first three 64-byte blocks are fixed. This computation is relatively fast compared to Bernstein's algorithm, so we have a bit of leeway for exhaustive search. We can compute a large number of digests, keeping the ones likely to give rise to a smooth t_i. We did this by selecting digests for which the resulting t_i would have many zeros as leading and trailing bits.

More precisely, we looked for a particular bit pattern at the beginning and at the end of each digest h_i, such that finding n matching bits results in n null bits at the beginning and at the end of t_i. The probability of finding n matching bits when we add the number of matches at the beginning and at the end is $(1 + n/2) \cdot 2^{-n}$, so we expect to compute $2^n/(1 + n/2)$ digests per selected message. We found $n = 8$ to be optimal: on average, we need about 50 digests to find a match, and the resulting t_i is at most $170 - 8 = 162$ bit long once powers of 2 are factored out.

Note that faster (e.g. hardware-enhanced) ways to obtain digests might significantly reduce the running time of the attack. We considered for example an FPGA-based solution called COPACOBANA [P+09], which could in principle perform a larger amount of exhaustive search, and speed up the attack dramatically. It turned out that our attack was fast enough, hence pursuing the hardware-assisted search idea further proved unnecessary, but a practical attack on EMV (see §9.7) could certainly benefit from hardware acceleration.

At present, Amazon is also renting time on instances equipped with fast GPUs. As GPUs are notoriously well-suited for the evaluation of hash functions likes SHA-1, it might be interesting to try and run the attack on such machines.

Finding smooth and semi-smooth integers. Once a batch of 2^{19} appropriate t_i's is generated, we factor out powers of 2, and feed the resulting odd numbers into our C++ implementation of Bernstein's algorithm. This implementation uses the MPIR multi-precision arithmetic library [H+09], which we chose over vanilla GMP because of a number of speed improvements, including J.W. Martin's patch for the Core2 architecture. We further applied Gaudry, Kruppa and Zimmermann's FFT patch, mainly for their implementation of *Mersenne FFT multiplication*, which is useful in the scaled remainder tree [Ber04b].

We looked for B-smooth and for (B, B_2)-semi-smooth t_i's, where $B = 16{,}290{,}047$ is the 2^{20}-th prime, and $B_2 = 2^{27}$. Each batch took $\simeq 40$ seconds to generate and to process, and consumed about 500 MB of memory. We ran 8 such processes in parallel on each instance to take advantage of the 8 cores, and 19 instances simultaneously.

Finding the 1,050,667 columns (slightly in excess of the $\ell = 2^{20} = 1{,}048{,}576$ required) took a little over 2 days.

Factoring and finding collisions. The output of the previous stage is a large set of text files containing the smooth and semi-smooth t_i's together with the corresponding message numbers. Turning this data into a matrix suitable for the linear algebra stage mostly involved text manipulation in Perl to convert it to commands that could be piped into PARI/GP [Gro08]. The resulting $1{,}048{,}576 \times 1{,}050{,}667$ matrix had 14,215,602 non-zero entries (13.5 per column on average, or 10^{-5} sparsity; the columns derived from the large prime variant tend to have twice as many non-zero entries, of course).[3]

Linear algebra. We found non-zero kernel elements of the final sparse matrix over GF(2) using Coppersmith's block Wiedemann algorithm [Cop94] implemented in WLSS2 [KL99, Lob95], with parameters $m = n = 4$ and $\kappa = 2$. The whole computation took 16 hours on one 2.7 GHz personal computer, with the first (and longest) part of the computation using 2 cores, and the final part using 4 cores.

The program discovered 124 kernel vectors with Hamming weights ranging from 337,458 to 339,641. Since columns obtained from pairs of semi-smooth numbers account for two signatures each, the number of signature queries required to produce the 124 corresponding forgeries is slightly larger, and ranges between 432,903 and 435,859.

Being written with the quadratic sieve in mind, the block Wiedemann algorithm in WLSS2 works over GF(2). There exist, however, other implementations for different finite fields.

The whole experiment is summarized in Figure 9.1.

Fewer signature queries. In Appendix 9.E we address the question of reducing the number of signature queries in the attack.

9.6 Cost Estimates

The experiment described in the previous section can be used as a benchmark to estimate the cost of the attack as a function of the size of the t_i's, denoted a; this will be useful for analyzing the security of the EMV specifications, where a is bigger (204 bits instead of 170 bits).

We assume that the t_i's are uniformly distributed a-bit integers and express costs as a function of a. It is then easy to estimate the number of hash computations and smoothness tests necessary for the attack, since we know, by the work of Bach and

[3] This matrix actually contained a number of rows with only one nonzero entry or less. Those rows and the corresponding columns can be safely removed, and the process can be repeated on the resulting matrix until a fix-point is reached (see Appendix 9.F for details). This reduction process is frequently the first operation carried out by linear algebra packages when searching for kernel vectors. When applied to our matrix, it produced a reduced matrix of dimension $750{,}031 \times 839{,}908$. We found it most convenient to leave any such reduction step to the linear algebra package itself.

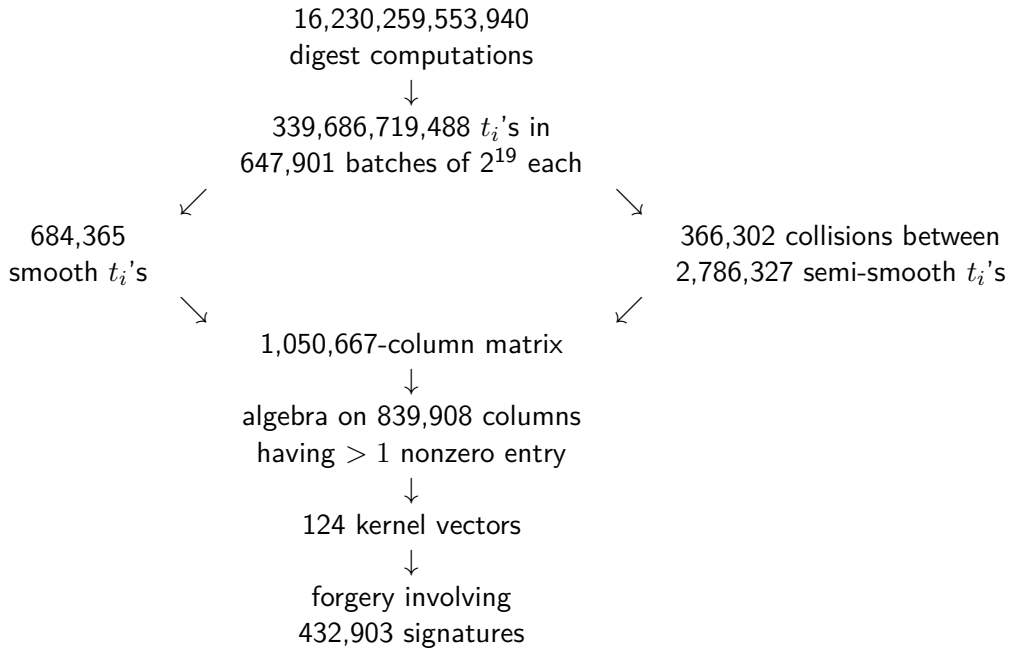


Figure 9.1: Summary of the practical attack on ISO/IEC 9796-2.

$a = \log_2 t_i$	$\log_2 \ell$	Estimated TotalTime	$\log_2 \tau$	EC2 cost (US$)
64	11	15 seconds	20	negligible
128	19	4 days	33	10
160	21	6 months	38	470
170	22	1.8 years	40	1,620
176	23	3.8 years	41	3,300
204	25	95 years	45	84,000
232	27	19 centuries	49	1,700,000
256	30	320 centuries	52	20,000,000

Table 9.2: Bernstein & large prime variant. Estimated parameter trade-offs, running times and costs, for various t_i sizes.

Peralta [BP96], the proportion of semismooth integers with respect to given bounds among all a-bit integers. The semismoothness bounds themselves are selected according to the results of Appendix 9.B.

Results are summarized in Table 9.2. Cost figures do not include the linear algebra step whose computational requirements are very low compared to the smoothness detection step. Another difference with our experiment is that here we do not assume any exhaustive search on the t_i's; this is why the cost estimate for $a = 170$ in Table 9.2 is about twice

the actual cost of our experimental ISO/IEC 9796-2 forgery.

Running times are given for a single 2.4 GHz PC. Costs correspond to the Amazon EC2 cloud (as of February 2009) like in the previous section. Estimates show that the attack is feasible up to $\simeq 200$ bits, but becomes infeasible for larger values of a. We also estimate $\log_2 \tau$, where τ is the number of messages in the forgery.

9.7 Application to EMV Signatures

EMV is a collection of industry specifications for the inter-operation of payment cards, POS terminals and ATMs. The EMV specifications [EMV] rely on ISO/IEC 9796-2 signatures to certify public keys and to authenticate data. For instance, when an Issuer provides application data to a Card, this data must be signed using the Issuer's private key S_{I}. The corresponding public-key P_{I} must be signed by a Certification Authority (CA) whose public key is denoted by P_{CA}. The signature algorithm is RSA with $e = 3$ or $e = 2^{16} + 1$. The bit length of all moduli is always a multiple of 8.

EMV uses special message formats; 7 different formats are used, depending on the message type. We first describe one of these formats: the *Static Data Authentication, Issuer Public Key Data* (SDA-IPKD), and adapt our attack to it. The other six formats are examined in Appendix 9.D.

9.7.1 EMV Static Data Authentication, Issuer Public Key Data (SDA-IPKD)

We refer the reader to §5.1, Table 2, page 41 in EMV [EMV]. SDA-IPKD is used by the CA to sign the issuer's public-key P_{I}. The message to be signed is as follows:

$$m = \texttt{02}_{16} \| X \| Y \| N_{\text{I}} \| \texttt{03}_{16}$$

where X represents 6 bytes that can be controlled by the adversary and Y represents 7 bytes that cannot be controlled. N_{I} is the Issuer's modulus to be certified. More precisely, $X = \text{ID} \| \text{DATE}$ where ID is the issuer identifier (4 bytes) and DATE is the *Certificate Expiration Date* (2 bytes); we assume that both can be controlled by the adversary. $Y = \text{CSN} \| C$ where CSN is the 3-bytes *Certificate Serial Number* assigned by the CA and C is a constant. Finally, the modulus to be certified N_{I} can also be controlled by the adversary.

Together with ISO/IEC 9796-2 encoding, this gives:

$$\mu(m) = \texttt{6A02}_{16} \| X \| Y \| N_{\text{I},1} \| H(m) \| \texttt{BC}_{16}$$

where $N_{\text{I}} = N_{\text{I},1} \| N_{\text{I},2}$ and the size of $N_{\text{I},1}$ is $k - k_h - 128$ bits. k denotes the modulus size and $k_h = 160$ as in ISO/IEC 9796-2.

9.7.2 Attacking SDA-IPKD

To attack SDA-IPKD write:

$$\mu(X, N_{\text{I},1}, h) = \texttt{6A02}_{16} \cdot 2^{k_1} + X \cdot 2^{k_2} + Y \cdot 2^{k_3} + N_{\text{I},1} \cdot 2^{k_4} + h$$

where Y is constant and $h = H(m)\|\texttt{BC}_{16}$. We have:

$$\begin{cases} k_1 = k - 16 \\ k_2 = k_1 - 48 = k - 64 \\ k_3 = k_2 - 56 = k - 120 \\ k_4 = k_h + 8 = 168. \end{cases}$$

Generate a random k_a-bit integer a, where $36 \leqslant k_a \leqslant 72$, and consider the equation:

$$b \cdot N - a \cdot \mu(X, 0, 0) = b \cdot N - a \cdot X \cdot 2^{k_2} - a \cdot (\texttt{6A02}_{16} \cdot 2^{k_1} + Y \cdot 2^{k_3}).$$

If we can find integers X and b such that $0 \leqslant X < 2^{48}$ and:

$$0 \leqslant b \cdot N - a \cdot \mu(X, 0, 0) < a \cdot 2^{k_3} \tag{9.5}$$

then as previously we can compute $N_{\text{I},1}$ by Euclidean division:

$$b \cdot N - a \cdot \mu(X, 0, 0) = (a \cdot 2^{k_4}) \cdot N_{\text{I},1} + r \tag{9.6}$$

where $0 \leqslant N_{\text{I},1} < 2^{k_3 - k_4}$ as required, and the resulting $b \cdot N - a \cdot \mu(X, N_{\text{I},1}, h)$ value will be small for all values of h.

In the above we assumed Y to be a constant. Actually the first 3 bytes of Y encode the CSN assigned by the CA, and may be different for each new certificate (see Appendix 9.C). However if the attacker can predict the CSN, then he can compute a different a for every Y and adapt the attack by factoring a into a product of small primes.

Finding small X and b so as to minimize the value of

$$|b \cdot N - a \cdot X \cdot 2^{k_2} - a \cdot (\texttt{6A02}_{16} \cdot 2^{k_1} + Y \cdot 2^{k_3})|$$

is a Closest Vector Problem in a bidimensional lattice—a problem that can be easily solved using the LLL algorithm [LLL82]. We first determine heuristically the minimal size that can be expected; we describe the LLL attack in Appendix 9.C.

Since $a \cdot \texttt{6A02}_{16} \cdot 2^{k_1}$ is an $(k + k_a)$-bit integer, with $X \simeq 2^{48}$ and $b \simeq 2^{k_a}$, we can heuristically hope to find X and b such that:

$$0 \leqslant b \cdot N - a \cdot \mu(X, 0, 0) < 2^{(k+k_a)-48-k_a} = 2^{k-48} \simeq a \cdot 2^{k-48-k_a} = a \cdot 2^{k_3+72-k_a}$$

which is $(72 - k_a)$-bit too long compared to condition (9.5). Therefore, by exhaustive search we will need to examine roughly 2^{72-k_a} different integers a to find a pair (b, X) that satisfies (9.5); since a is k_a-bits long, this can be done only if $72 - k_a \leqslant k_a$, which gives $k_a \geqslant 36$. For $k_a = 36$ we have to exhaust the 2^{36} possible values of a.

Once this is done we obtain from (9.6):

$$t = b \cdot N - a \cdot \mu(X, N_{\text{I},1}, h) = r - a \cdot h$$

with $0 \leqslant r < a \cdot 2^{k_4}$. This implies that the final size of t values is $168 + k_a$ bits. For $k_a = 36$ this gives 204 bits (instead of 170 bits for plain ISO/IEC 9796-2). The cost of the attack will thus be higher than for plain ISO/IEC 9796-2.

EMV mode	Format	$\|X\|$	$\|Y\|$	$\|t\|$	EC2 cost
SDA-IPKD	$\mathtt{02}_{16}\|X\|Y\|N_{\mathrm{I}}\|\mathtt{03}_{16}$	48	56	204	45,000
SDA-SAD	Y	-	$k-176$	-	-
ODDA-IPKD	$\mathtt{02}_{16}\|X\|Y\|N_{\mathrm{I}}\|\mathtt{03}_{16}$	48	56	204	45,000
ODDA-ICC-PKD	$\mathtt{04}_{16}\|X\|Y\|N_{\mathrm{ICC}}\|\mathtt{03}_{16}\|\mathrm{DATA}$	96	56	204	45,000
ODDA-DAD1	Y	-	$k-176$	-	-
ODDA-DAD2	Y	-	$k-176$	-	-
ICC-PIN	$\mathtt{04}_{16}\|X\|Y\|N_{\mathrm{ICC}}\|\mathtt{03}_{16}$	96	56	204	45,000

Table 9.3: EMV message formats. X denotes a data field controllable by the adversary. Y is not controllable. Data sizes for X, Y and t are expressed in bits.

In Appendix 9.C we exhibit concrete (a, b, X) values for $k_a = 52$ and for the RSA-2048 challenge; this required $\simeq 2^{23}$ trials (109 minutes on a single PC). We estimate that for $k_a = 36$ this computation will take roughly 13 years on a single PC, or equivalently US$11,000 using the EC2 cloud.

Table 9.2 shows that attacking 204-bit t_i's would cost US$84,000. As for the ISO/IEC 9796-2 attack, we can decrease this cost by first doing exhaustive search on the bits of $H(m)$ to obtain a smaller t-value. We found that with 8 bits of exhaustive search cost drops to about US$45,000 (without the matrix step, but in our attack linear algebra takes a relatively small amount of time).

9.7.3 Summary

In Appendix 9.D we provide an analysis of the other formats in the EMV specifications, with corresponding attacks when such attacks exist. We summarize results in Table 9.3 where an X represents a string that can be controlled by the adversary, while Y cannot be controlled. The size of the final t-value to be smooth is given in bits. Note that cost estimates are cheaper than Table 9.2 because we first perform exhaustive search on 8 bits of $H(m) = \text{SHA-1}(m)$; however here we do take into account the cost of computing these SHA-1(m) values.

Table 9.3 shows that only four of the EMV formats can be attacked, with the same cost as the SDA-IPKD format. The other formats seem out of reach because the non-controllable part Y is too large.

9.8 Conclusion

This chapter exhibited a practically exploitable flaw in the ISO/IEC 9796-2 standard and a conceptual flaw in EMV signatures. In response to this attack, the ISO/IEC 9796-2 standard was amended [ISO9796–2:2010] to discourage the use of the ad-hoc signature padding in contexts where chosen-message attacks are an issue.

We can see this cryptanalysis as another indication that it might wise for the industry to move away from ad-hoc constructs from the 1980s and early 1990s, and embrace provable security as the cryptologic community and standardization bodies have done in the past decade.

9.A Optimizing Bernstein's Batch Size

We assume that for a single batch the algorithm runs in time:

$$\text{BatchTime}(n'_t, a, \ell) = c \cdot b' \cdot \log^2 b' \cdot \log\log b'$$

where c is a constant and:

$$b' = n'_t \cdot a + u \tag{9.7}$$

is the bit-length of the batch, and $u = \ell \cdot \log_2 \ell$ is the p_i-list's size in bits. The total running time is then:

$$\text{TotalTime}(n_t, a, \ell, n'_t) = \frac{n_t}{n'_t} \cdot c \cdot b' \cdot \log^2 b' \cdot \log\log b'.$$

The running time of a single batch only depends on b'. Hence, as a first approximation one could select an n'_t equating the sizes of the t_i-list and the p_i-list; this yields $n'_t \cdot a = u$. A more accurate analysis (see below) reveals that TotalTime is minimized for a slightly larger n'_t value; more precisely for an n'_t such that:

$$n'_t \cdot a = \frac{u \log u}{2}.$$

Using (9.7) this gives $b' = (u \log u)/2$ and a total running time of:

$$\text{TotalTime}(n_t, a, \ell) \simeq c \cdot n_t \cdot a \cdot \log^2 b' \cdot \log\log b'.$$

We now proceed with the analysis of the optimal n'_t. For the sake of clarity we temporarily denote b' by b. Let $u = \ell \cdot \log_2 \ell$ and

$$n'_t = \frac{u}{a} \cdot \alpha$$

for some parameter α. We look for the optimal α. We have $b = u \cdot (\alpha + 1)$ and:

$$\text{TotalTime}(n_t, a, \ell, \alpha) = \frac{n_t \cdot a}{u \cdot \alpha} \cdot c \cdot b \cdot \log^2 b \cdot \log\log b.$$

We neglect the $\log\log b$ term and consider the function:

$$f(u, \alpha) = \frac{b \cdot \log^2 b}{\alpha} \quad \text{where} \quad b = u \cdot (\alpha + 1).$$

Setting:

$$\frac{\partial f(u, \alpha)}{\partial \alpha} = 0$$

we get:

$$u \cdot (\log^2 b + 2\log b) \cdot \alpha - b\log^2 b = 0 \quad \text{i.e.} \quad (\log b + 2) \cdot \alpha = (\alpha + 1)\log b$$

and then $2\alpha = \log b$, which gives:

$$2\alpha = \log u + \log(\alpha + 1).$$

Neglecting the $\log(\alpha+1)$ term, we finally get $\alpha \simeq \log u/2$ as the optimal value of α. This translates into running time as:

$$\text{TotalTime}(n_t, a, \ell) \simeq c \cdot n_t \cdot a \cdot \log^2 b \log\log b$$

where $b = (u \log u)/2$ and $u = \ell \cdot \log_2 \ell$.

9.B Large Prime Variant: Complexity Analysis

In this appendix we provide an accurate analysis of the large prime variant in the context of our attack.

Assume that we check our t_i-list for (B, B_2)-semi-smoothness (instead of B-smoothness) and detect η semi-smooth numbers. Amongst those, we expect to find $\eta\lambda$ numbers that are actually B-smooth, for some $\lambda \in [0, 1]$ that can be expressed in terms of ρ and σ functions. If we further assume that the $\eta(1-\lambda)$ remaining numbers, which are semi-smooth but non-smooth, have their largest prime factors uniformly distributed amongst the h primes between B and B_2, we expect to find about $\eta^2(1-\lambda)^2/(2h)$ "collisions" between them, that is, about $\eta^2(1-\lambda)^2/(2h)$ pairs of numbers with the same largest prime factor.

Note that:

$$h \simeq \frac{B_2}{\log B_2} - B.$$

Let ℓ be the number of primes less than B. The smooth numbers in the list yield a total of $\eta\lambda$ exponent vectors over the first ℓ primes, and each of the collisions between the remaining semi-smooth numbers yields such an additional exponent vector. Since we need (slightly more than) ℓ vectors to forge a signature, we should examine enough t_i's to find η semi-smooth numbers, where η satisfies:

$$\ell = \eta\lambda + \frac{\eta^2(1-\lambda)^2}{2h}.$$

Solving for η, we get:

$$\eta = \frac{2\ell}{\lambda + \sqrt{2\ell \cdot (1-\lambda)^2/h + \lambda^2}}.$$

The probability β that a random a-bit integer is semi-smooth with respect to B_2 and $B \simeq \ell \cdot \log \ell$ is:

$$\beta = \sigma\left(\frac{a\log 2}{\log(\ell \log \ell)}, \frac{a \log 2}{\log B_2}\right)$$

a	128	144	160	176	192
Optimal $\log_2(\ell)$	19	20	21	23	24
Best ϑ	1.43	1.46	1.49	1.43	1.45

Table 9.4: Improvement factors from the large prime variant.

and if α denotes the probability that a random a-bit integer is B-smooth, we have:

$$\lambda = \frac{\alpha}{\beta} = \frac{\rho\left(\frac{a\log 2}{\log(\ell\log\ell)}\right)}{\sigma\left(\frac{a\log 2}{\log(\ell\log\ell)}, \frac{a\log 2}{\log B_2}\right)}.$$

In this large prime variant, we only need to generate $n'_t = \eta/\beta$ numbers to find enough exponent vectors, as opposed to $n_t = \ell/\alpha$ previously. Therefore, the large prime variant improves upon simple smoothness by a factor of roughly:

$$\vartheta = \frac{n_t}{n'_t} = \frac{\ell/\alpha}{\eta/\beta} = \frac{1}{\lambda}\cdot\frac{\ell}{\eta} = \frac{1}{2}\left[1 + \sqrt{1 + \frac{2\ell}{h}\left(\frac{1}{\lambda} - 1\right)^2}\right] \geqslant 1. \tag{9.8}$$

ϑ is always greater than 1, and for the sizes we are interested in, say $100 \leqslant a \leqslant 200$, we find $\vartheta \simeq 1.5$ for the best choice of B, and $B_2 \gtrsim 7B$.[4] The reader is referred to Table 9.4 for precise figures.

9.C LLL Attack on EMV SDA-IPKD Encoding

9.C.1 The LLL attack

Given a, N we must minimize the value of:

$$\left| b\cdot N - a\cdot X\cdot 2^{k_2} - a\cdot(\texttt{6A02}_{16}\cdot 2^{k_1} + Y\cdot 2^{k_3})\right|.$$

We show how this can be done using LLL. We write:

$$u = a\cdot 2^{k_2}$$
$$v = a\cdot(\texttt{6A02}_{16}\cdot 2^{k_1} + Y\cdot 2^{k_3})$$

[4] According to formula (9.8), ϑ increases until B_2 reaches $\simeq 7B$, and decreases slowly thereafter. This is actually *not* the case: finding a larger t_i population to be semi-smooth can only produce *more* collisions. The decrease suggested by formula (9.8) stems from the assumption that the largest prime factors of the t_i's are uniformly distributed amongst the h primes between B and B_2, which is only approximately true. The imprecision grows with h (a larger B_2 doesn't spread the largest prime factors more thinly). Choosing a very large B_2 is not advisable, however, because it produces considerable extra output (searching for collisions becomes cumbersome) with negligible returns in terms of actual collisions. In the practical attack, we selected $\ell = 2^{20}$ and $B_2 = 2^{27} \simeq 9B$.

where $N \simeq 2^k$, $X \simeq 2^{48}$, $a \simeq 2^{k_a}$, $u \simeq 2^{k-64+k_a}$ and $v \simeq 2^{k+k_a}$. Hence we must minimize the absolute value of:

$$t = b \cdot N - x \cdot u - v.$$

Consider the lattice of column vectors:

$$L = \begin{pmatrix} & & 2^{k-48} \\ & 2^{k-96} & \\ N & -u & -v \end{pmatrix}$$

As seen previously, heuristically, we can obtain $t \simeq 2^{k-48}$; therefore the coefficients in L are chosen so as to obtain a short vector of norm $\simeq 2^{k-48}$. More precisely, we look for a short column vector $\boldsymbol{c} \in L$ of the form:

$$\boldsymbol{c} = \begin{pmatrix} 2^{k-48} \\ x \cdot 2^{k-96} \\ b \cdot N - u \cdot x - v \end{pmatrix}$$

Theorem 9.2 (LLL)**.** *Let L be a lattice spanned by $(u_1, \ldots, u_\omega)$. The LLL algorithm, given the vectors $(u_1, \ldots, u_\omega)$, finds in polynomial time a vector b_1 such that:*

$$\|b_1\| \leqslant 2^{(\omega-1)/4} \det(L)^{1/\omega}.$$

Therefore, using LLL we can find a short vector of norm:

$$\|b_1\| \leqslant 2 \cdot (\det L)^{1/3} \leqslant 2 \cdot (2^{3k-144})^{1/3} \leqslant 2^{k-47}.$$

Heuristically we hope that $b_1 = \boldsymbol{c}$, which allows solving for the values of b and X. The attack is heuristic but it works very well in practice, as shown in the next section.

9.C.2 Practical value for EMV SDA-IPKD

Consider again the SDA-IPKD EMV format; we write:

$$\mu(X, N_{\text{I},1}, h) = \texttt{6A02}_{16} \cdot 2^{k_1} + X \cdot 2^{k_2} + Y \cdot 2^{k_3} + N_{\text{I},1} \cdot 2^{k_4} + h$$

where the constant Y is taken to be:

$$Y = \texttt{010203 0101 F8 01}_{16}.$$

The first 3 bytes correspond to the CSN assigned by the CA (we took $\texttt{010203}_{16}$), $\texttt{0101}_{16}$ corresponds to the *hash algorithm indicator* and to the *public-key algorithm indicator*. $\texttt{F8}_{16} = 248$ is the issuer public-key length (in bytes) and $\texttt{01}_{16}$ is the length of the public exponent ($e = 3$).

Taking the RSA-2048 challenge for N, we have run the attack (Appendix 9.C.1) for $k_a = 52$ and found the following values after 8,303,995 $\simeq 2^{23}$ iterations:

$$a = 4127135343129068 \quad b = 2192055331476458 \quad X = 66766242156276$$

which are such that $0 < X < 2^{48}$ and:

$$0 \leqslant b \cdot N - a \cdot \mu(X, 0, 0) < a \cdot 2^{k_3} \tag{9.9}$$

as required.

The computation took $\simeq 109$ minutes on a single 2 GHz PC. Therefore, for $k_a = 36$ we expect that 2^{36} trials to yield a triple $\{a, b, X\}$ satisfying condition (9.9) such that $|a| \leqslant 2^{36}$, within a running time of $\simeq 109 \cdot 2^{36-20} = 4.3 \cdot 10^8$ minutes $= 13$ years on a single PC, or equivalently for US$11,000 using the EC2 cloud.

9.D EMV Signature Encoding Formats

EMV specifies the following encoding formats, based on the ISO/IEC 9796-2 standard. The new attack applies to modes preceded by the sign ⧫ and does not apply to modes preceded by a ◊.

1. ⧫ Static Data Authentication, Issuer Public Key Data. EMV SDA-IPKD §5.1, Table 2, page 41.
The signing entity is the CA. The signed message is the Issuer's public-key P_{I}.

$$m = 02_{16} \| X \| Y \| N_{\text{I}} \| 03_{16}.$$

While being operationally debatable we assume that X (the concatenation of the *Issuer's Identifier* (4 bytes) and the *Certificate Expiration Date* (2 bytes)) and N_{I} (the Issuer's modulus to be certified) can be both controlled by the attacker. Y (7 bytes) cannot be controlled by the adversary.

2. ◊ Static Data Authentication, Static Application Data. EMV SDA-SAD §5.1, Table 3, page 42.
The signing entity is the Issuer. The signed message is the Issuer's public-key P_{I}. As the first part of the message m is fixed, the attack does not apply.

3. ⧫ Offline Dynamic Data Authentication, Issuer Public-Key Data. EMV ODDA-IPKD §6.1, Table 10, page 57.
The signing entity is the CA. The signed message is the Issuer's public-key P_{I}. The message format is identical to SDA-IPKD.

4. ⧫ Offline Dynamic Data Authentication, ICC Public-Key Data. EMV SDA-ICC-PKD §6.1, Table 11, page 58.
The signing entity is the Issuer. The signed message is the Card's public-key P_{I}.

$$m = 04_{16} \| X \| Y \| N_{\text{ICC}} \| 03_{16} \| \text{DATA}.$$

While being operationally debatable we assume that X (12 bytes) and N_{ICC} (the Card's modulus to be certified) can be both controlled by the attacker. Y (7 bytes)

cannot be controlled by the adversary. Here DATA is static data to be authenticated, as specified in Section 10.3, Book 3, EMV specifications; DATA can only appear in the non-recoverable part of the message in the ISO/IEC 9796-2 standard.

5. ♢ Offline Dynamic Data Authentication, Dynamic Application Data. EMV ODDA-DAD1 §6.5, Table 15, page 67.
The signing entity is the Card. As the first part of the message m is fixed ($\mathtt{BB}_{16}$ padding), the attack does not apply.

6. ♢ Offline Dynamic Data Authentication, Dynamic Application Data. EMV ODDA-DAD2 §6.6, Table 18, page 73.
The signing entity is the Card. As the first part of the message m is fixed[5], the attack does not apply.

7. ♦ Personal Identification Number Encipherment, ICC PIN Encipherment Public Key Data. EMV ICC PIN §7.1, Table 23, page 83.
The signing entity is the Issuer.

$$m = \mathtt{04}_{16} \| X \| Y \| N_{\mathrm{ICC}} \| \mathtt{03}_{16}.$$

While being operationally debatable we assume that X (12 bytes) and N_{ICC} (the Card's modulus to be certified) can be both controlled by the attacker. Y (7 bytes) cannot be controlled by the adversary.

Commercial impact. It is very fortunate that ODDA-DAD1 and ODDA-DAD2 do not lend themselves to the new attack. Indeed, in the ODDA-DAD modes the signing device is the payment card, which is supposedly in the opponent's hands. In addition, the signing capacity of EMV cards is limited by a ratification counter restricting the number of signatures performed by the card during its lifetime.

9.E Fewer Queries

The number of signatures *actually used* by the forger is not τ but the number of nonzero β_i values in the formula:

$$\mu(m_\tau) = \left(\prod_{j=1}^{\ell} p_j^{\gamma_j} \right)^e \cdot \prod_{i=1}^{\tau-1} \mu(m_i)^{\beta_i}.$$

Assuming that $(\beta_1, \ldots, \beta_{\tau-1})$ is a random vector of $\mathbb{Z}_e^{\tau-1}$ only $\tau(e-1)/e$ of the signatures will be actually used to compute the forgery. The gain is significant when

[5] Here $m = \mathtt{0501}_{16} \| Y \| X$ where Y is a $32 + 1 = 33$ bytes string that cannot be controlled by the adversary (32 leftmost bytes of ICC Dynamic Data). X can be controlled by the adversary.

e is a very small exponent (e.g. 2 or 3). However, one can try to generate more than τ candidates but select the subset of signatures minimizing the number of nonzero β_i values. Such a sparse β-vector may allow to reduce the number of queries and defeat *ratification counters* meant to restrict the number of authorized signature queries.

In essence, we are looking at a random $[\ell, k]$ code: a kernel vector has ℓ components which, for $e = 2$, can be regarded as a set of independent unbiased Bernoulli variables. The probability that such a vector has weight less than $w = \sum_{i=1}^{\tau-1} \beta_i$ is thus:

$$\sum_{j=1}^{w} \binom{\ell}{j} 2^{-\ell} \simeq \frac{1}{2}\left(1 + \operatorname{erf}\left(\frac{w - \ell/2}{\sqrt{\ell/2}}\right)\right).$$

We have 2^k such vectors in the kernel, hence the probability that at least one of them has a Hamming weight smaller than w is surely bounded from above by:

$$2^k \times \frac{1}{2}\left(1 + \operatorname{erf}\left(\frac{w - \ell/2}{\sqrt{\ell/2}}\right)\right) = 2^{k-1}\left(1 + \operatorname{erf}\left(\frac{w - \ell/2}{\sqrt{\ell/2}}\right)\right).$$

Let c denote the density bias of w, i.e. $w = (1/2 - c)\ell$. The previous bound becomes:

$$\begin{aligned} p(c) &= 2^{k-1}\left(1 + \operatorname{erf}\left(-c\sqrt{2\ell}\right)\right) = 2^{k-1}\left(1 - \operatorname{erf}\left(c\sqrt{2\ell}\right)\right) \\ &= 2^{k-1}\operatorname{erfc}(c\sqrt{2\ell}) \underset{\ell \to +\infty}{\sim} \frac{2^{k-1}\exp(-2\ell c^2)}{c\sqrt{2\pi\ell}}. \end{aligned}$$

For $\ell = 2^{20}$, even if we take k as large as 2^{10} (the largest subspace dimension considered tractable, even in much smaller ambient spaces), we get $p(1/50) \simeq 10^{-58}$, so the probability that there exists a kernel vector of weight $w < 500{,}000$ is negligible. In addition, even if such a vector existed, techniques for *actually computing it*, e.g. [BLP08], seem to lag far behind the dimensions we deal with.

It follows that a better strategy to diminish w is to simply decrease ℓ. The expected payoff might not be that bad: If the attacker is limited to, say, 2^{16} signatures, then he can pick $\ell = 2^{17}$, and for 196-bit numbers (204 bits minus 8 bits given by exhaustive search), the attack becomes about 15 times slower than the optimal choice, $\ell = 2^{24}$ (note as well that more exhaustive search becomes possible in that case). That's slow, but perhaps not excruciatingly so.

9.F Expected Number of Queries

A further question that arises when studying the problem of minimizing the number of signature queries in the attack is that of evaluating the expected Hamming weight of the nullspace vectors returned by the linear algebra step, with or without the large prime variant. The problem is nontrivial and still open. For simplicity we only consider the GF(2) case ($e = 2$) the following, but the problem is not fundamentally different for other exponents.

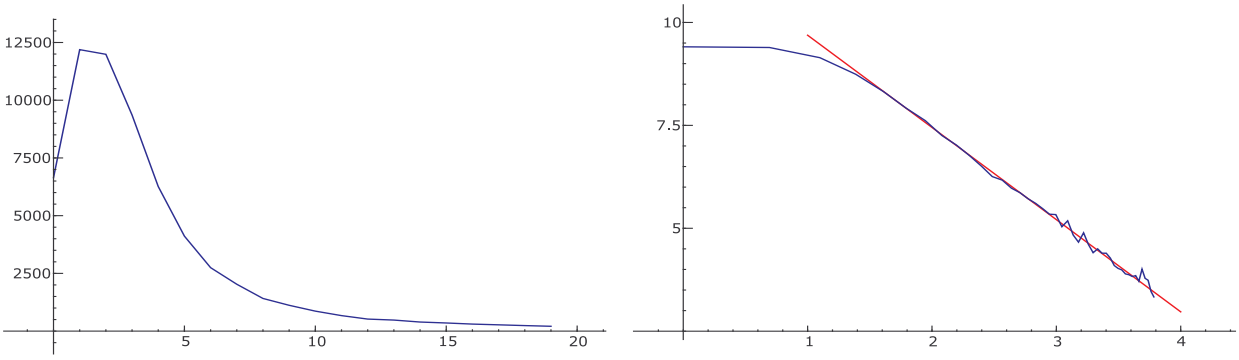


Figure 9.2: Distribution of row weights for a smooth matrix of size 2^{16}. Linear scale (left) and log-log scale (right).

Some primes can never appear as components of nullspace vectors. That is the case, in particular, when the corresponding row of the matrix contains only one nonzero element (i.e. if the prime divides only one of our smooth or semismooth numbers to an odd power): indeed, the column containing that nonzero element is clearly linearly independent from all other columns. Hence, we may as well remove that row and column from the matrix, and do so for every similar pair of rows and columns.

Additional rows of weight 1 may turn up after those removals, so we need to repeat the process recursively. A fix-point is reached (i.e. the algorithm stops) when there is no singleton row left in the reduced matrix.

Let ℓ_0 denote the total number of rows in this reduced matrix. It seems reasonable to conjecture that (in the case of a square matrix at least) the expected weight of nullspace vectors is of the order of $\ell_0/2$. In other words, the primes that *can* appear as components of nullspace vectors are likely to do so randomly. The question then reduces to evaluating ℓ_0.

All this prompts us to estimate how many rows in the original matrix contain exactly one nonzero entry, or more generally exactly n nonzero entries for small n. We expected this variable to follow a Poisson distribution, or perhaps a sum of Poisson distributions, but experiments appear to challenge this expectation: the empirical distribution does not have exponential decay but a fat tail, as evidenced by the plots in Figures 9.2 and 9.3. Finding a simple description of this distribution seems like a rather difficult open problem in itself.

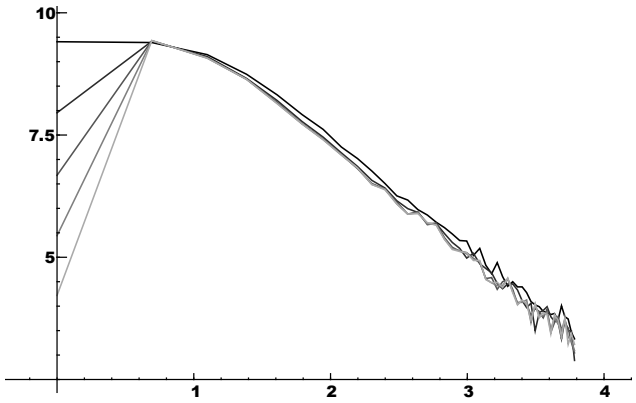


Figure 9.3: Distribution of row weights (log-log scale) for a smooth matrix of size 2^{16} and the first four iterations of the reduction step applied to it. Lighter curves correspond to further iterations.

Chapter 10

Fault Attacks on EMV Signatures

10.1 Introduction

In this chapter, we continue our investigation of the vulnerabilities of the ISO/IEC 9796-2 and EMV signature standards from a different angle. Whereas in Chapter 9 we were interested in constructing signature forgeries in the "black-box" chosen-ciphertext attack model, we now turn to fault attacks: in this threat model, the adversary can tamper with a physical signing device and try to introduce errors during signature generation, by hardware attacks such as voltage spikes in the power source or laser beams targeted at memory chips.

Fault attacks were introduced by Boneh, DeMillo and Lipton [BDL01] in 1997. They showed that RSA signatures, in particular, and especially when used with the Chinese Remainder Theorem as is almost always the case in practice, were quite vulnerable to this type of attack. In fact, a single faulty signature produced by an unprotected device is enough to reveal the private signing key. Since then, fault attacks on RSA signatures and countermeasures against them have been an active research topic.

Boneh et al.'s attack, however, is not directly applicable to ISO/IEC 9796-2 and EMV signatures, because some of the information used to compute the signature is unknown to the adversary (either because it is randomly chosen at the time of signature generation, or because it is not transmitted along with the signature but recovered when verifying the—correct!—signature).

At CHES 2009, Coron, Joux, Kizhvatov, Naccache and Paillier [CJK+09] showed how the attack could still be adapted if the unknown message parts were small enough, by retrieving the small unknown bit strings as small roots of a multivariate polynomial, using variants of the lattice-based technique introduced by Coppersmith [Cop97]. The size restriction on unknown message parts with their technique was quite severe, however, making their attack difficult to apply in practice.

In this chapter, we present a different strategy for recovering the unknown message parts when several faulty signatures are available, also based on lattice reduction but in a simpler way, and that can handle much longer unknown bit strings very efficiently.

We show for example how ten faulty signatures in a specific EMV signature format are enough to retrieve the private signing key with our new attack, while the same format was well beyond the reach of the previous attack. This work was presented at CT-RSA 2010 [CNT10].

10.1.1 Fault attacks on RSA-CRT

As mentioned above, one of the first and best-known examples of a fault attack in cryptography is the one described by Boneh, DeMillo and Lipton [BDL97, BDL01, JLQ99] on RSA-CRT signatures. Let us first recall it succinctly.

To sign a message m with RSA, a signer computes $\sigma = \mu(m)^d \bmod N$, where N is the public modulus, d is the private exponent and μ is a certain encoding function. Since this computation is time-consuming, it is very common to use the Chinese Remainder Theorem (CRT) to obtain a roughly 4-fold speed-up, by first evaluating:

$$\sigma_p = \mu(m)^{d \bmod p-1} \pmod p \quad \text{and} \quad \sigma_q = \mu(m)^{d \bmod q-1} \pmod q$$

and then deducing σ from σ_p and σ_q with the CRT.

This method is vulnerable to fault attacks: if an attacker can inject a fault during the computation of σ_q, the whole computation will produce a faulty signature σ' satisfying

$$\sigma' \equiv \mu(m)^d \pmod p \quad \text{and} \quad \sigma' \not\equiv \mu(m)^d \pmod q$$

which allows the attacker to factor N by computing

$$\gcd\left((\sigma')^e - \mu(m) \bmod N, N\right) = p.$$

Clearly, Boneh et al.'s fault attack applies to any deterministic RSA encoding μ, such as the Full Domain Hash [BR96], where μ is a full-length hash function, or certain add-hoc signature paddings such as PKCS#1 v1.5 [PKCS#1 v1.5]. The attack is also applicable to probabilistic signature schemes where the random nonce used to generate the signature is sent along with the signature, like PFDH [Cor02].

However, if the nonce is only recovered when verifying the signature, or if some part of the message is unknown, the attack does not apply directly. For example, with a probabilistic encoding of the form $\mu(m) = m\|r$ where a random nonce r is simply concatenated with the message m, the nonce r is only recovered when verifying a correct signature. Given a faulty signature σ', the attacker cannot retrieve r nor infer $(m\|r)$ which would be necessary to compute $\gcd\left((\sigma')^e - \mu(m) \bmod N, N\right) = p$. It is not advisable to use this particular toy encoding (for example, one can clearly forge signatures if r is allowed to be as large as $N^{1-1/e}$), but more serious probabilistic encodings in which the randomness is recovered as part of signature verification, such as PSS [BR96], are similarly immune to Boneh et al.'s attack.

10.1.2 The attack of Coron et al.

At CHES 2009, Coron, Joux, Kizhvatov, Naccache and Paillier [CJK+09] showed how, for certain padding schemes, the case of unknown nonces or partially unknown messages

can be tackled nonetheless if the unknown part is not too large. The attack uses a technique by Herrmann and May [HM08] for finding small roots of linear equations modulo an unknown factor p of a public modulus N. This technique is in turned based on Coppersmith's lattice-based method for finding small roots of polynomials [Cop97].

The main application considered by Coron et al. was RSA signatures using the ISO/IEC 9796-2 standard padding scheme [ISO9796–2:2002], where the message is partially unknown, a common situation in smart card applications, especially EMV smart cards [EMV]. The ISO/IEC 9796-2 encoding function is of the form:

$$\mu(m) = \texttt{6A}_{16} \,\|\, m[1] \,\|\, H(m) \,\|\, \texttt{BC}_{16}$$

where $m = m[1]\|m[2]$ is split in two parts, and H is a hash function, typically SHA-1. Only $m[2]$ is transmitted together with the signature $\sigma = \mu(m)^d \bmod N$. To recover the whole message and verify the signature, the recipient first computes $\mu(m) = \sigma^e \bmod N$, from which he deduces $m[1]$, and then checks that the hash value $H(m)$ is correct.

In the cases considered in [CJK+09], the message prefix $m[1]$ is partially known to a fault attacker (because messages are formatted in a particular way, as is common in applications) but some variable part is unknown, and the hash value $H(m)$ is unknown as well as a result. Yet, if the unknown part of $m[1]$ is not too large (up to about 160 bits for a 2048-bit modulus), then, like in Boneh et al.'s attack, the private key can be recovered with a single fault.

In addition, the same paper presents an extension of this attack to the case when multiple faults are available, which makes it theoretically possible to deal with larger unknown message parts. However, this variant involves lattice reduction in dimensions exponential in the number of faults (using heuristic, multivariate versions of Coppersmith's method), and becomes quickly impractical.

10.1.3 Our contribution

In this chapter, we present another multiple fault attack in the same setting eschewing Coppersmith-like techniques entirely. It is based on orthogonal lattices, as used in earlier cryptanalytic results such as [NS97, NS98]. The lattice sizes involved are moderate and make the attack quite practical for relatively large unknown message parts.

We implement a simulation of the attack and show it to scale gracefully with the number of available faulty signatures. Unknown message parts very close to the theoretical maximum of half of the modulus length can be recovered in seconds.

We also show that the attack applies to a number of EMV signature formats. For a particular EMV use case well beyond the reach of the attack in [CJK+09], 10 faulty signatures are sufficient with our technique to recover the private key in a fraction of a second.

10.1.4 Outline

In preparation for the orthogonal lattice techniques used in this and the next chapter, we start by collecting a few results about integer lattices and lattice reduction (§10.2).

We then recall the definition of the ISO/IEC 9796-2 padding function and describe the structure of CRT faults on ISO/IEC 9796-2 signatures (§10.3). Based on this structure, we explain the idea of Coron et al.'s Coppersmith-based attack as well as its limitations (§10.4). Then, we describe our new multiple-fault attack (§10.5) and provide some simulation results to assess its practicality and compare it to the previous attack (§10.6). We show that our new attack does apply to EMV signatures (§10.7) and finally suggest some possible countermeasures (§10.8).

10.2 Preliminaries on Lattices

We start with some preliminary definitions and results about lattices. A similar presentation with a larger scope and more details can be found in [Ngu09, §6].

10.2.1 Notation and background

We will consider $\mathbb{R}^n$ endowed with its usual structure as an Euclidean vector space. Bold letters will denote vectors, usually in row notation. If $\boldsymbol{x} = (x_1, \ldots, x_n)$ and $\boldsymbol{y} = (y_1, \ldots, y_n)$ are two vectors, their Euclidean inner product is denoted by

$$\langle \boldsymbol{x}, \boldsymbol{y} \rangle = \sum_{i=1}^{n} x_i y_i.$$

The corresponding Euclidean norm is denoted by

$$\|\boldsymbol{x}\| = \sqrt{\langle \boldsymbol{x}, \boldsymbol{x} \rangle} = \sqrt{x_1^2 + \cdots + x_n^2}.$$

For any subset S of $\mathbb{R}^n$, we define the linear span of S, denoted by $\mathrm{span}(S)$, as the minimal vector subspace of $\mathbb{R}^n$ containing S.

The Gram determinant of $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_m \in \mathbb{R}^n$, denoted by $\Delta(\boldsymbol{b}_1, \ldots, \boldsymbol{b}_m)$, is by definition the determinant of the Gram matrix $(\langle \boldsymbol{b}_i, \boldsymbol{b}_j \rangle)_{1 \leqslant i,j \leqslant m}$. This real number $\Delta(\boldsymbol{b}_1, \ldots, \boldsymbol{b}_m)$ is always $\geqslant 0$, and it turns out to be zero if and only if the $\boldsymbol{b}_i$'s are linearly dependent. The Gram determinant is invariant by any permutation of the m vectors, and by any integral linear transformation of determinant ± 1 such as adding to one of the vectors a linear combination of the others. The Gram determinant has a very useful geometric interpretation: when the $\boldsymbol{b}_i$'s are linearly independent, $\sqrt{\Delta(\boldsymbol{b}_1, \ldots, \boldsymbol{b}_m)}$ is the m-dimensional volume of the parallelepiped spanned by the $\boldsymbol{b}_i$'s.

10.2.2 Lattices and lattice bases

For our purposes, a *lattice* will be any subgroup L of $(\mathbb{Z}^n, +)$ for some n (such a subgroup is sometimes called an *integral lattice*, to distinguish it from more general definitions of a lattice). Examples include $\mathbb{Z}^n$ itself, the zero lattice $\{\boldsymbol{0}\}$, and the set $a\mathbb{Z} + b\mathbb{Z}$ of linear combinations of any two integers $a, b \in \mathbb{Z}$ (which is simply $\gcd(a, b)\mathbb{Z}$). For another example, consider n integers $a_1, \ldots, a_n$, together with a modulus M. Then the set of

all $(x_1, \ldots, x_n) \in \mathbb{Z}^n$ such that $\sum_{i=1}^n a_i x_i \equiv 0 \pmod{M}$ is a lattice, as it is clearly a subgroup of $\mathbb{Z}^n$.

Let $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_m$ be arbitrary vectors in $\mathbb{Z}^n$. Denote by $L(\boldsymbol{b}_1, \ldots, \boldsymbol{b}_m)$ the set of all integral linear combinations of the $\boldsymbol{b}_i$'s:

$$L(\boldsymbol{b}_1, \ldots, \boldsymbol{b}_m) = \left\{ \sum_{i=1}^m n_i \boldsymbol{b}_i : n_1, \ldots, n_m \in \mathbb{Z} \right\}.$$

This set is a subgroup of $\mathbb{Z}^n$, and hence a lattice L, called the lattice *generated* or *spanned* by the $\boldsymbol{b}_i$'s. When the $\boldsymbol{b}_i$'s are linearly independent, we say that $(\boldsymbol{b}_1, \ldots, \boldsymbol{b}_m)$ is a *basis* of the lattice L, in which case each lattice vector decomposes itself uniquely as an integral linear combination of the $\boldsymbol{b}_i$'s.

Bases and sets of generators are useful to represent lattices, and to perform computations. One will typically represent a lattice by some lattice basis, which can itself be represented by a matrix with integer coefficients. If $(\boldsymbol{b}_1, \ldots, \boldsymbol{b}_m)$ is a basis of L, with $\boldsymbol{b}_i = (b_{i,1}, \ldots, b_{i,n})$, we will represent the lattice $L = L(\boldsymbol{b}_1, \ldots, \boldsymbol{b}_m)$ by the following matrix:

$$\begin{pmatrix} b_{1,1} & \cdots & b_{1,n} \\ \vdots & & \vdots \\ b_{m,1} & \cdots & b_{m,n} \end{pmatrix}$$

whose rows are the coordinates of the $\boldsymbol{b}_i$'s.

We define the *dimension* or *rank* of a lattice L, denoted by $\dim(L)$, as the dimension d of the vector space $\mathrm{span}(L)$. The dimension is the maximal number of linearly independent lattice vectors. Any lattice basis of L must have exactly d elements. It is clear that there exist d linearly independent lattice vectors, but such vectors do not necessarily form a basis, contrary to what happens in the case of vectors spaces. However, it is in fact always possible to find a basis of L: in other words, lattices of dimension d in $\mathbb{Z}^n$ are exactly all subsets of $\mathbb{Z}^n$ of the form

$$L = L(\boldsymbol{b}_1, \ldots, \boldsymbol{b}_d)$$

for some linearly independent vectors $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_d \in \mathbb{Z}^n$.

10.2.3 Lattice volume

Let $(\boldsymbol{b}_1, \ldots, \boldsymbol{b}_d)$ and $(\boldsymbol{c}_1, \ldots, \boldsymbol{c}_d)$ be two bases of a lattice L in $\mathbb{R}^n$. An elementary result states that there exists a $d \times d$ integral matrix $U = (u_{i,j})_{1 \leqslant i,j \leqslant d} \in M_d(\mathbb{Z})$ of determinant ± 1 such that $\boldsymbol{c}_i = \sum_{j=1}^d u_{i,j} \boldsymbol{b}_j$ for all $1 \leqslant i \leqslant d$. It follows that the Gram determinants of those two bases are equal:

$$\Delta(\boldsymbol{b}_1, \ldots, \boldsymbol{b}_d) = \Delta(\boldsymbol{c}_1, \ldots, \boldsymbol{c}_d) > 0.$$

The *volume* of the lattice L is then defined as:

$$\mathrm{vol}(L) = \Delta(\boldsymbol{b}_1, \ldots, \boldsymbol{b}_d)^{1/2}$$

which is independent of the choice of the lattice basis $(\boldsymbol{b}_1, \ldots, \boldsymbol{b}_d)$. It corresponds to the d-dimensional volume of the parallelepiped spanned by any basis. In the important case of full-dimensional lattices where $\dim(L) = n = \dim(\mathbb{R}^n)$, the volume is also equal to the absolute value of the determinant of any lattice basis.

If an explicit basis is known, computing the lattice volume amounts to computing a determinant. However, in some cases, we may not know any explicit lattice basis; in such a case, the following elementary result may be useful for computing the volume nonetheless. If L_1 and L_2 are two lattices in $\mathbb{Z}^n$ of the same dimension such that $L_1 \subset L_2$, then L_2/L_1 is a finite group of order denoted by $[L_2 : L_1]$ which satisfies

$$\mathrm{vol}(L_1) = [L_2 : L_1] \cdot \mathrm{vol}(L_2).$$

For example, consider n integers $a_1, \ldots, a_n$, together with a modulus M. We have seen in §10.2.2 that the set L of all $(x_1, \ldots, x_n) \in \mathbb{Z}^n$ such that $\sum_{i=1}^n a_i x_i \equiv 0 \pmod{M}$ is a lattice in $\mathbb{Z}^n$, but it is not obvious how to explicitly write down a basis of L. However, note that $L \subset \mathbb{Z}^n$ and that the dimension of L is n because L contains $M\mathbb{Z}^n$. It follows that $\mathrm{vol}(L) = [\mathbb{Z}^n : L]$. Furthermore, the definition of L says that it is the kernel of the group homomorphism $\mathbb{Z}^n \to \mathbb{Z}/M\mathbb{Z}$ sending $(x_1, \ldots, x_n)$ to $\sum_{i=1}^n a_i x_i$. The image of this morphism is the subgroup generated by $\gcd(M, a_1, a_2, \ldots, a_n)$, and hence:

$$\mathrm{vol}(L) = [\mathbb{Z}^n : L] = \frac{M}{\gcd(M, a_1, a_2, \ldots, a_n)}.$$

10.2.4 Lattice reduction

As previously noted, any lattice has a basis. In fact, any lattice of dimension 2 or more has infinitely many bases. However, some of these bases are "better" than others, in the sense that they give more compact descriptions of the lattice and make it easier to answer various questions about the lattice.

In the case of Euclidean vector spaces, it is usually convenient to work with orthogonal bases. Lattices, on the other hand, do not always admit orthogonal bases. Nevertheless, it can be shown that one can always find so-called *reduced bases*, consisting of "relatively short" vectors that are "not too far" from being orthogonal.

Giving precise definitions for such notions as "short vectors" and "reduced bases", showing that such vectors or such bases exist, and describing methods to find them, is the purpose of the theory of lattice reduction.

Minkowski's successive minima. In order to explain what a reduced basis is, we need to define what is meant by short lattice vectors. Let L be a lattice of dimension $\geqslant 1$ in $\mathbb{R}^n$. Since any closed hyperball centered at zero contains finitely many vectors of L, there exists a non-zero vector in L of minimal length. The Euclidean norm of that shortest non-zero vector is called the *first minimum* of L, and is denoted by $\lambda_1(L) > 0$. Any vector $\boldsymbol{v}$ in L of norm $\lambda_1(L)$ is called a shortest vector of L. It is never unique, since $-\boldsymbol{v}$ is also a shortest vector.

For that reason, one must be careful when defining the *second-to-shortest* vector of a lattice. Minkowski [Min96] defined the other minima as follows. For all $1 \leqslant i \leqslant \dim(L)$, the i-th *minimum* $\lambda_i(L)$ is defined as the minimum of $\max_{1 \leqslant j \leqslant i} \|\boldsymbol{v}_j\|$ over all families of i linearly independent lattice vectors $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_i \in L$. Clearly, the minima are increasing: $\lambda_1(L) \leqslant \lambda_2(L) \leqslant \cdots \leqslant \lambda_d(L)$. And it is not difficult to see that there always exist linearly independent lattice vectors $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_d$ reaching simultaneously the minima, that is $\|\boldsymbol{v}_i\| = \lambda_i(L)$ for all i.

As a side note, perhaps surprisingly, it is not the case in general that such vectors form a basis of L. In fact, one can find lattices (of dimension $\geqslant 5$) in which there is no basis at all formed by vectors of length equal to the successive minima.

Random lattices. It is possible to give a relatively precise description of how Minkowski's minima behave "generically", namely for so-called random lattices. The precise definition of a random lattice is somewhat technical but the idea is that there is a natural way to pick a lattice "uniformly at random" (see e.g. [Ajt02] for details).

It is proved in [Ajt06] that a random lattice of rank n satisfies asymptotically, with overwhelming probability:

$$\forall\, 1 \leqslant i \leqslant n, \quad \lambda_i(L) \approx \sqrt{\frac{n}{2\pi e}}\, \mathrm{vol}(L)^{1/n}. \tag{10.1}$$

This means that all minima are close to each other and to the radius of an n-dimensional ball of volume $\mathrm{vol}(L)$.

Furthermore, [Ajt06] also shows that asymptotically, in a random n-rank lattice L, there exists with overwhelming probability a lattice basis $(\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n)$ such that:

$$\forall\, 1 \leqslant i \leqslant n, \quad \|\boldsymbol{b}_i\| \approx \sqrt{\frac{n}{2\pi e}}\, \mathrm{vol}(L)^{1/n}. \tag{10.2}$$

Such a basis consists of very short vectors, since their norms are close to the successive minima. Thus, random lattices typically have nice bases.

In applications, lattices are often not picked at random in the previous sense but constructed in a particular way. Such non-random lattices can sometimes behave quite differently from random lattices: for example, the first minimum can be arbitrarily small compared to $\mathrm{vol}(L)^{1/n}$ for lattices constructed in an ad hoc way. However, it is often a reasonable heuristic to infer the "typical" behavior of lattices in a given problem from the case of random lattices, and assume that the previous properties hold most often nonetheless.

Reduction notions and LLL. We just mentioned that random lattices always have nice bases: what about general lattices? The goal of lattice reduction is to prove the existence of nice lattice bases in every lattice, and to describe procedures to find such bases. These nice bases are called *reduced*.

There are multiple definitions of a reduced basis: usually, one first defines a notion of reduction, then shows that there exist bases which are reduced in this sense, and

finally proves that bases which are reduced in this sense have interesting properties, mathematical (how short are the vectors of a reduced basis?) and computational (how easy is it to compute the basis?).

One classical notion of reduction, which will be enough for our purposes, is the Lenstra-Lenstra-Lovász reduction [LLL82] (LLL for short). It isn't very strong (in the sense that vectors of an LLL-reduced basis aren't necessarily very short) but it is computationally inexpensive: we can compute LLL-reduced basis of lattices of relatively large dimension quickly.

We will not need a precise definition of LLL reduction, but the following few properties will come in handy. All lattices admit LLL-reduced bases, and any LLL-reduced basis $(\boldsymbol{b}_1, \ldots, \boldsymbol{b}_d)$ of a lattice L satisfies:

1. $\|\boldsymbol{b}_1\| \leqslant 2^{(d-1)/4} (\operatorname{vol} L)^{1/d}$.
2. For all $1 \leqslant i \leqslant d$, $\|\boldsymbol{b}_i\| \leqslant 2^{(d-1)/2} \lambda_i(L)$.
3. $\|\boldsymbol{b}_1\| \times \cdots \times \|\boldsymbol{b}_d\| \leqslant 2^{d(d-1)/4} \operatorname{vol} L$.

The vectors of an LLL-reduced basis are thus at most exponentially far from the minima.

10.3 Modeling Faults on ISO/IEC 9796-2 Signatures

10.3.1 The ISO/IEC 9796-2 signature scheme

As discussed in Chapter 9, ISO/IEC 9796-2 is an encoding standard allowing partial message recovery [ISO9796–2, ISO9796–2:2002], and used in many embedded applications, including EMV smart cards [EMV]. The encoding can be used with hash functions H of various digest sizes k_h. When k_h, the message size and the size of N (denoted k) are all multiples of 8, the ISO/IEC 9796-2 encoding of a message $m = m[1] \,\|\, m[2]$ is

$$\mu(m) = \mathtt{6A}_{16} \,\|\, m[1] \,\|\, H(m) \,\|\, \mathtt{BC}_{16}$$

where $m[1]$ consists of the $k - k_h - 16$ leftmost bits of m and $m[2]$ represents the remaining bits of m. In [ISO9796–2:2002] it is required that $k_h \geqslant 160$.

The ISO/IEC 9796-2 encoding itself is deterministic, but it is often used with messages containing parts which are either secret or picked at random upon signature generation. This is particularly the case in EMV smart cards. The message to be signed can be put in the following general form: $m = m[1] \,\|\, m[2]$ with

$$m[1] = \alpha \,\|\, r \,\|\, \alpha' \quad \text{and} \quad m[2] = \textsc{data}$$

where r is an unknown bit string (e.g. a random nonce), α and α' are bit strings known to the attacker, and DATA is a possibly unknown bit string. Thus, the ISO/IEC 9796-2-encoded message is

$$\mu(m) = \mathtt{6A}_{16} \,\|\, \alpha \,\|\, r \,\|\, \alpha' \,\|\, H(\alpha \,\|\, r \,\|\, \alpha' \,\|\, \textsc{data}) \,\|\, \mathtt{BC}_{16}$$

where both r and the hash value are unknown. In typical use cases, the hash value is a 160-bit long SHA-1 digest, and N is an RSA modulus of around 1024 bits. The unknown string r can be of various sizes depending on the nature of the signed message; the shorter r is, the easier the attack becomes.

More generally, we will consider encoded messages of the form:

$$\mu(m) = A + B \cdot x_0 + C \cdot y_0$$

where B and C are public constants (in the ISO/IEC 9796-2 case, suitable powers of 2), A is a number representing the known part of the encoding, and x_0 and y_0 are the unknown parts (equal to r and $H(m)$ respectively). The signature is then computed as $\sigma = \mu(m)^d \bmod N$ as usual, using the CRT.

10.3.2 Attack model

A fault is injected in the exponentiation modulo q part of the RSA-CRT signature generation, resulting in a faulty signature σ satisfying

$$\sigma^e \equiv A + B \cdot x_0 + C \cdot y_0 \pmod{p} \quad \text{and} \quad \sigma^e \not\equiv A + B \cdot x_0 + C \cdot y_0 \pmod{q}.$$

Dividing by B and subtracting the left-hand side, the faulty signature yields a relation of the form

$$a + x_0 + c \cdot y_0 \equiv 0 \pmod{p} \quad \text{and} \quad a + x_0 + c \cdot y_0 \not\equiv 0 \pmod{q}$$

where $a = B^{-1}(A - \sigma^e) \bmod N$ is a value known to the attacker, and $c = B^{-1} \cdot C \bmod N$ is a public constant.

In other words, the fault attack provides the attacker with an integer a such that for a certain unknown pair (x_0, y_0) of bounded size, the following holds:

$$a + x_0 + c \cdot y_0 \equiv 0 \pmod{p} \quad \text{and} \quad a + x_0 + c \cdot y_0 \not\equiv 0 \pmod{q}.$$

We will write the bounds on x_0 and y_0 as $0 \leqslant x_0 < N^\gamma$ and $0 \leqslant y_0 < N^\delta$. The total fraction of unknown bits in the encoded message is thus $\gamma + \delta$.

Intuitively, in both Coron et al.'s attack and our new attack, the adversary takes advantage of the affine relation in x_0 and $y_0 \bmod p$ to recover the unknown part with lattice techniques, and use it to factor N.

In practice, the fault can be injected using e.g. voltage spikes during the computation modulo q (see [CJK+09, §4]).

The resulting value modulo q (of the faulty signature, or equivalently of $a + x + cy$) is usually modeled as a random element in $\mathbb{Z}_q$: this is the random fault model.

10.4 The Small Root Attack

We now describe the previous attack by Coron et al. [CJK+09] in the single-fault case, and show how it extends to multiple faults (albeit with poor efficiency).

10.4.1 Single-fault attack

After a single fault, the attacker obtains a bivariate linear polynomial $f(x, y) = a + x + c \cdot y$ such that $f(x_0, y_0) \equiv 0 \pmod{p}$ and $f(x_0, y_0) \not\equiv 0 \pmod{q}$ for some unknown pair (x_0, y_0) satisfying known bounds. They can then apply the following Coppersmith-like result by Herrmann and May.

Theorem 10.1 (Herrmann-May [HM08])**.** *Let N be a sufficiently large composite integer with a divisor $p \geqslant N^\beta$. Let $f \in \mathbb{Z}[x, y]$ be a bivariate linear polynomial. Assume that $f(x_0, y_0) = 0 \mod p$ for some (x_0, y_0) such that $|x_0| \leqslant N^\gamma$ and $|y_0| \leqslant N^\delta$. Then for any $\varepsilon > 0$, under the condition*

$$\gamma + \delta \leqslant 3\beta - 2 + 2(1 - \beta)^{3/2} - \varepsilon \tag{10.3}$$

one can find linearly independent polynomials $h_1, h_2 \in \mathbb{Z}[x, y]$ such that $h_1(x_0, y_0) = h_2(x_0, y_0) = 0$ over $\mathbb{Z}$, in time polynomial in $\log N$ and ε^{-1}.

In our case, N is a (presumably balanced) RSA modulus, so $\beta = 1/2$ and condition (10.3) becomes:

$$\gamma + \delta < \frac{\sqrt{2} - 1}{2} \approx 0.207. \tag{10.4}$$

Under that condition, we obtain two linearly independent polynomials $h_1, h_2 \in \mathbb{Z}[x, y]$ such that (x_0, y_0) is a root of both.

Suppose further that h_1, h_2 are in fact *algebraically* independent. It is then easy to compute the finite list of their common roots, e.g. by taking the resultant with respect to y and solving for x, and thus to find (x_0, y_0). Once (x_0, y_0) is computed, one can proceed as in Boneh et al.'s attack to factor N:

$$\gcd\left(f(x_0, y_0) \bmod N, N\right) = p.$$

The algebraic independence assumption makes the attack heuristic, but the experimental validation carried out in [CJK+09] suggests that it works well for $\gamma + \delta$ not too large. For example, with a 2048-bit modulus and 160-bit hash function, a 158-bit long nonce r is recovered in less than a minute with a single fault. The total fraction of unknown message bits is then $\gamma + \delta = (158 + 160)/2048 \approx 0.155$.

However, condition (10.4) is quite restrictive, especially when N is small. Indeed, for a 1024-bit modulus with 160-bit hash function, the theoretical maximum nonce size that can be recovered is 58 bits, and the algorithm is only manageable in practice for much smaller sizes. As a result, the attack performs worse than exhaustive search for r for this modulus size.

10.4.2 Extension to several faults

To tackle larger unknown message parts, Coron et al. suggest [CJK+09, §2.4] taking advantage of $\ell > 1$ faulty signatures obtained with the same signing key. This gives a

family of ℓ linear polynomials $f_i(x, y) = a_i + x + c \cdot y$ each of which has a small root (x_i, y_i) modulo p.

Using independent variables for all of these polynomials and multiplying powers of them together, they get a large number of polynomials in 2ℓ variables, each of which has the small root $(x_1, y_1, \ldots, x_\ell, y_\ell)$ modulo p. If the small root is small enough, it is then possible to construct a lattice from those multivariate polynomials such that lattice reduction yields 2ℓ linearly independent polynomials with the same small root $(x_1, y_1, \ldots, x_\ell, y_\ell)$ *over the integers*. If those polynomials happen to be *algebraically* independent, this reveals the unknown message parts x_i, y_i and we can factor N as before.

The algebraic independence condition in this case is a real hurdle, but more importantly, the dimension of the lattice constructed in that way quickly becomes very large. Theoretically, given a sufficiently large number of faults, the extended attack could tackle cases where $\gamma + \delta$ is asymptotically close to $1/2$. However, since the lattice dimension grows exponentially with the number ℓ of faulty signatures, achieving $\gamma + \delta$ values significantly higher than the single-fault maximum of 0.207 is quite impractical. Table 10.2 illustrates how intractable the problem gets as $\gamma + \delta$ approaches $1/2$. As a result, this approach doesn't extend the scope of the single-fault attack by a large margin.

10.5 Our New Multiple-Fault Attack

We now describe our new multiple-fault attack, which uses lattices in a very different way. We are now given ℓ faulty signatures, and in particular a vector $\boldsymbol{a} = (a_1, \ldots, a_\ell)$ of integers such that, for two short unknown vectors $\boldsymbol{x} = (x_1, \ldots, x_\ell)$, $\boldsymbol{y} = (y_1, \ldots, y_\ell)$, we have:

$$\boldsymbol{a} + \boldsymbol{x} + c \cdot \boldsymbol{y} \equiv \boldsymbol{0} \pmod{p}. \tag{10.5}$$

The attack then proceeds in three steps.

Linearization. The first step is to eliminate $\boldsymbol{a}$ in (10.5) to obtain a linear, rather than affine, relation. To do so, we try to find vectors $\boldsymbol{u}_j$ orthogonal to $\boldsymbol{a}$ modulo N, and take the inner products of (10.5) with the $\boldsymbol{u}_j$'s.

The set $L_{\boldsymbol{a}} = \{\boldsymbol{u} \in \mathbb{Z}^\ell \mid \langle \boldsymbol{a}, \boldsymbol{u} \rangle \equiv 0 \pmod{N}\}$ of vectors $\boldsymbol{u}$ orthogonal to $\boldsymbol{a}$ modulo N is a lattice in $\mathbb{Z}^\ell$ of rank ℓ and volume N, as seen in §10.2.3. It is possible to find a reduced basis of it by standard orthogonal lattice techniques [NS97] as follows. Denote by $L_{\boldsymbol{a}}^{(0)}$ the lattice in $\mathbb{Z}^{\ell+1}$ generated by the rows of the matrix

$$\begin{pmatrix} \kappa N & 0 & \cdots & 0 \\ \kappa a_1 & 1 & & 0 \\ \vdots & & \ddots & \\ \kappa a_\ell & 0 & & 1 \end{pmatrix} \tag{10.6}$$

where κ is a large scaling constant. Now for any vector $\boldsymbol{u}^{(0)} = (u_0, u_1, \ldots, u_\ell)$ in this lattice, one has $u_0 = \kappa(a_1 u_1 + \cdots + a_\ell u_\ell + mN)$ for some integer m. In particular, if

$|u_0| < \kappa$, then $u_0 = 0$, and $\boldsymbol{u} = (u_1, \ldots, u_\ell)$ must satisfy $\langle \boldsymbol{a}, \boldsymbol{u} \rangle \equiv 0 \pmod N$. Thus, any vector $\boldsymbol{u}^{(0)}$ in $L_{\boldsymbol{a}}^{(0)}$ whose norm is less than κ gives rise to a vector $\boldsymbol{u} \in L_{\boldsymbol{a}}$. Conversely, if $\boldsymbol{u}$ is a point in $L_{\boldsymbol{a}}$, then $\boldsymbol{u}^{(0)} = (0, u_1, \ldots, u_\ell)$ is in $L_{\boldsymbol{a}}^{(0)}$.

Since it admits $N\mathbb{Z}^\ell$ as a sublattice, the lattice $L_{\boldsymbol{a}}$ in $\mathbb{Z}^\ell$ contains a linearly independent family consisting of ℓ vectors of length at most N. Taking the corresponding vectors in $L_{\boldsymbol{a}}^{(0)}$, this implies that $\lambda_j(L_{\boldsymbol{a}}^{(0)}) \leqslant N$ for $1 \leqslant j \leqslant \ell$.

Consider then an LLL-reduced basis $(\boldsymbol{u}_1^{(0)}, \ldots, \boldsymbol{u}_{\ell+1}^{(0)})$ of $L_{\boldsymbol{a}}^{(0)}$. Recall from §10.2.4 that a property of LLL reduction is that for $1 \leqslant j \leqslant \ell$,

$$\|\boldsymbol{u}_j^{(0)}\| \leqslant 2^{\ell/2} \lambda_j(L_{\boldsymbol{a}}^{(0)}) \leqslant 2^{\ell/2} N.$$

Hence, if one chooses κ large enough, all vectors $\boldsymbol{u}_j^{(0)}$, $1 \leqslant j \leqslant \ell$, are of norm less than κ and give rise to independent vectors $\boldsymbol{u}_j$ that are orthogonal to $\boldsymbol{a}$ modulo N (it is in fact an LLL-reduced basis of the lattice of such orthogonal vectors). In particular, taking the inner product of (10.5) with $\boldsymbol{u}_j$, we get, for $1 \leqslant j \leqslant \ell$

$$\langle \boldsymbol{x}, \boldsymbol{u}_j \rangle + c \langle \boldsymbol{y}, \boldsymbol{u}_j \rangle \equiv 0 \pmod p. \tag{10.7}$$

Moreover, we can give a heuristic estimate the size of the vectors $\boldsymbol{u}_j$. Assuming that $L_{\boldsymbol{a}}$ behaves like a random lattice, the length of these vectors should be around $\sqrt{\ell/2\pi e} \cdot N^{1/\ell}$ since $\mathrm{vol}(L_{\boldsymbol{a}}) = N$. Neglecting the small multiplicative factor, we can heuristically expect that $\|\boldsymbol{u}_j\| \lesssim N^{1/\ell}$.

Orthogonalization. For all j, let $\alpha_j = \langle \boldsymbol{x}, \boldsymbol{u}_j \rangle$ and $\beta_j = \langle \boldsymbol{y}, \boldsymbol{u}_j \rangle$. We expect to have, for all j except the last few, $|\alpha_j| \lesssim N^{\gamma+1/\ell}$ and $|\beta_j| \lesssim N^{\delta+1/\ell}$. Now recall from (10.7) that

$$\alpha_j + c \cdot \beta_j \equiv 0 \pmod p.$$

If $(\alpha_j, \beta_j) \neq (0, 0)$, this amounts to writing $-c$ as the modular ratio $\alpha_j/\beta_j \bmod p$. But this is only possible in general if the sum of sizes of α_j and β_j is at least the size of p. In other words, if $N^{\gamma+1/\ell} \cdot N^{\delta+1/\ell} \ll p$, that is

$$\gamma + \delta + \frac{2}{\ell} \lesssim \frac{1}{2} \tag{10.8}$$

we should have $\alpha_j = \beta_j = 0$ for all j except perhaps the last few. This means that $\boldsymbol{u}_j$ is orthogonal to $\boldsymbol{x}$ and $\boldsymbol{y}$ in $\mathbb{Z}^\ell$.

Assume that this does in fact hold for $1 \leqslant j \leqslant \ell - 2$, and consider the lattice of vectors in $\mathbb{Z}^\ell$ orthogonal to $\boldsymbol{u}_j$ for $1 \leqslant j \leqslant \ell - 2$. This is a bidimensional lattice containing $\boldsymbol{x}$ and $\boldsymbol{y}$, and we can find a basis $(\boldsymbol{x'}, \boldsymbol{y'})$ of it with LLL. This is done by computing the LLL-reduction of the following lattice in $\mathbb{Z}^{\ell-2+\ell}$ generated by the rows of

$$\begin{pmatrix} \kappa' u_{1,1} & \cdots & \kappa' u_{\ell-2,1} & 1 & & 0 \\ \vdots & & \vdots & & \ddots & \\ \kappa' u_{1,\ell} & \cdots & \kappa' u_{\ell-2,\ell} & 0 & & 1 \end{pmatrix}$$

where κ' is a suitably large constant [NS97].

Factoring. Finally, using LLL again as in the linearization step, we can construct a short vector $\boldsymbol{v}$ orthogonal to both $\boldsymbol{x'}$ and $\boldsymbol{y'}$ modulo N. Then $\boldsymbol{v}$ is also orthogonal mod N to all $\mathbb{Z}$-linear combinations of $\boldsymbol{x'}$ and $\boldsymbol{y'}$, in particular $\boldsymbol{x}$ and $\boldsymbol{y}$. Therefore, taking the inner product of (10.5) with $\boldsymbol{v}$, we get

$$\langle \boldsymbol{a}, \boldsymbol{v} \rangle \equiv 0 \pmod{p}$$

and we can thus factor N as required:

$$\gcd(\langle \boldsymbol{a}, \boldsymbol{v} \rangle, N) = p.$$

Summary. If condition (10.8) is satisfied, the following algorithm should heuristically find a nontrivial factor of N:

1. Find an LLL-reduced basis $(\boldsymbol{u}_1, \ldots, \boldsymbol{u}_\ell)$ of the lattice of vectors in $\mathbb{Z}^\ell$ orthogonal to $\boldsymbol{a}$ modulo N.
2. Find a basis $(\boldsymbol{x'}, \boldsymbol{y'})$ of the lattice of vectors in $\mathbb{Z}^\ell$ orthogonal to $\boldsymbol{u}_j$, $1 \leqslant j \leqslant \ell - 2$.
3. Find a short vector $\boldsymbol{v}$ orthogonal to $\boldsymbol{x'}$ and $\boldsymbol{y'}$ modulo N (for example the first vector of an LLL-reduced basis of the lattice formed by such orthogonal vectors).
4. Return $\gcd(\langle \boldsymbol{a}, \boldsymbol{v} \rangle, N)$.

The attack is heuristic because it might be the case that $\boldsymbol{u}_j$ isn't actually orthogonal to $\boldsymbol{x}$ and $\boldsymbol{y}$ in $\mathbb{Z}^\ell$ for all $j \leqslant \ell - 2$.

Experimentally, however, success rates are very high when condition (10.8) is verified, as we show in the next section.

10.6 Simulation Results

We simulate our new fault attack as follows: we first generate a correct $\sigma_p = \mu(m)^d \bmod p$ and a random $\sigma_q \in \mathbb{Z}_q$ and then compute the faulty signature σ using the Chinese Remainder Theorem. This mimics the process described in [CJK+09] where concrete faults are injected into devices generating ISO/IEC 9796-2 signatures. We then feed the resulting faulty signatures into the algorithm described above and check whether it successfully factors the signing key.

We first collect results on the success rate of the attack $\gamma + \delta = 0.33$. Condition (10.8) predicts that the number of faulty signatures should satisfy $\ell \gtrsim 12$, and we do in fact find a success rate of 100% for $\ell = 13$ or 14, both for balanced and unbalanced (γ, δ) configurations, as shown in Table 10.1.

We then assess the practicality and efficiency of our attack for various unknown message part lengths $\gamma + \delta$, and compare it to the attack by Coron et al. [CJK+09]. The results are presented in Table 10.2, which provides, for several values of $\gamma + \delta$, the following information: the number of faulty signatures ℓ_{new} used in our simulation, the

Number of faults ℓ	12	13	14
Success rate with $\gamma = \delta = \frac{1}{6}$	13%	100%	100%
Success rate with $\gamma = \frac{1}{4}$, $\delta = \frac{1}{12}$	0%	100%	100%
Average CPU time (seconds)	0.19	0.14	0.17

Table 10.1: Attack simulation results using SAGE. Random 1024-bit moduli. Single 2.5 GHz Intel CPU core.

$\gamma + \delta$	ℓ_{new}	ω_{new}	CPU time (new)	ℓ_{old}	ω_{old}	CPU time (old)
0.204	7	8	0.03 s	3	84	49 s
0.214	8	9	0.04 s	2	126	22 min
0.230	8	9	0.04 s	2	462	—
0.280	10	11	0.07 s	6	6188	—
0.330	14	15	0.17 s	8	2^{21}	—
0.400	25	26	1.44 s	—	—	—
0.450	70	71	36.94 s	—	—	—

Table 10.2: Comparison of the new attack with [CJK+09] for a random 1024-bit modulus.

corresponding lattice dimension ω_{new}, and the running time of the new attack, as well as the corresponding values ℓ_{old}, ω_{old} and estimated running time for the multi-fault attack of [CJK+09] (the stated $\gamma + \delta$ being out of reach of the single-fault variant).

As we can see from the table, the attack by Coron et al. has the advantage of requiring fewer faulty signatures to attack the same $\gamma + \delta$ value. However, lattice dimensions are much smaller in our case, making the attack considerably faster, and indeed practical for values very close to the theoretical maximum of 1/2, which are completely out of reach of the previous attack.

10.7 Application to EMV Signatures

10.7.1 The EMV specification

EMV is a collection of industry specifications for the inter-operation of payment cards, POS terminals and ATMs. The EMV specification [EMV] uses ISO/IEC 9796-2 signatures to certify public-keys and to authenticate data. For instance, to authenticate itself, the payment card must issue a signature on data provided by the terminal. The signature algorithm is RSA with ISO/IEC 9796-2 using $e = 3$ or $e = 2^{16} + 1$. The bit length of all moduli is always a multiple of 8. EMV uses special message formats; 7 different formats are used, depending on the message type.

In the following, for clarity's sake, we analyze one of these formats only: the *Offline Dynamic Data Authentication, Dynamic Application Data* format, described in Book 2,

Section 6.5, Table 15, page 67 of the EMV specifications [EMV]. The signing entity is the Card. The message m to be signed has the format $m = m[1] \| m[2]$ with:

$$m[1] = \mathtt{0501}_{16} \parallel \mathsf{L_{DD}} \parallel \mathsf{ICCDD} \parallel \mathtt{BB} \cdots \mathtt{BB}_{16}$$
$$m[2] = \text{DATA}$$

where $\mathsf{L_{DD}}$ is a byte identifying the length (in bytes) of the ICC Dynamic Data string ICCDD, and DATA is some data provided by the terminal. In general, the ICC Dynamic Data string has the following form:

$$\mathsf{ICCDD} = \mathsf{L_{ICCDN}} \parallel \mathsf{ICCDN} \parallel \mathsf{ADD}$$

where $\mathsf{L_{ICCDN}}$ is one byte identifying the length (in bytes) of the time-variant ICC Dynamic Number ICCDN, and ADD consists of $\mathsf{L_{DD}} - \mathsf{L_{ICCDN}} - 1$ bytes of Additional Dynamic Data to be signed. It is specified that one must have $2 \leqslant \mathsf{L_{ICCDN}} \leqslant 8$.

As mentioned in the EMV specifications, the ICC Dynamic Number can be an unpredictable number or a counter incremented for every new signature. In a typical use case (as described, for example, as part of EMV Test 2CC.086.1 Case 07 [EMV TC]), ICCDN is a random 8-byte string generated by the card, and ADD is a variable 8-byte string, encoded according to [ISO8825–1]. In this case, we have:

$$m[1] = \mathtt{0501}_{16} \parallel \mathtt{11}_{16} \parallel \mathtt{08}_{16} \parallel \mathsf{ICCDN} \parallel \mathsf{ADD} \parallel \mathtt{BB} \cdots \mathtt{BB}_{16}$$

which can be rewritten as:

$$m[1] = \mathsf{X} \parallel r \parallel \mathtt{BB} \cdots \mathtt{BB}_{16}$$

where X is a known value and r is a variable byte string of bit-size $k_r = 128$. This gives:

$$\mu(m) = \mathtt{6A}_{16} \parallel \mathsf{X} \parallel r \parallel \mathtt{BB} \cdots \mathtt{BB}_{16} \parallel H(m) \parallel \mathtt{BC}_{16} \tag{10.9}$$

where $H(m)$ is a 160-bit digest of the encoded message m. Note that the no-fault forgery attack from Chapter 9 does not apply because here $m[1]$ cannot be controlled by the adversary.

10.7.2 Fault attack

The EMV format for $\mu(m)$ given in (10.9) is the same as the one recalled in §10.3.1 and considered in our new attack. In the particular use case described above, the string X is known but r and $H(m)$ are unknown to the attacker. Therefore the total number of unknown bits is:

$$k_r + k_h = 128 + 160 = 288.$$

Hence, for a 1024-bit modulus N, we get:

$$\gamma + \delta = \frac{288}{1024} \approx 0.28$$

which is well beyond the range of practical applicability of [CJK+09], as shown in Table 10.2. However, as apparent from the same table, our new attack will factor N in a fraction of a second using about ten faulty signatures.

10.8 Proposed Countermeasures

Since faulty signatures are invalid, a possible countermeasure is to check the signature after generation. Since signature verification is significantly cheaper than signature generation in cases of interest (because of a low public exponent), this is a reasonable option, and preferable to incurring the 4-fold performance penalty of not using the Chinese Remainder Theorem at all. More generally, other usual countermeasures against RSA-CRT fault attacks, as proposed e.g. by Shamir [Sha99, JPY01] or Giraud [Gir06], apply to this setting. See however [KQ07] for an indication that such strategies may prove ineffective in more elaborate attack models.

Furthermore, it should be noted that ad hoc signature paddings have no proof of security even against standard attacks. The ISO/IEC 9796-2 padding scheme, in particular, has a number of known vulnerabilities [CNS99, CNTW09]. It may be advisable to use a probabilistic RSA signature scheme like PSS [BR96] instead, which is actually secure in the random fault model considered here, as proved by Coron and Mandal [CM09].

Chapter 11

Modulus Fault Attacks Against RSA-CRT Signatures

11.1 Introduction

This chapter, like the previous one, is devoted to the description of a new fault attack on RSA signatures using the Chinese Remainder Theorem, and the mathematical tools involved are again the orthogonal lattice techniques of Nguyen and Stern.

However, while most of the work on RSA-CRT fault attacks and countermeasures since the seminal 1997 paper of Boneh, DeMillo and Lipton has concentrated on the perturbation of one of the two half exponentiations in signature generation, in this work we consider the injection of faults in the *public modulus* before CRT interpolation. This is a very different type of fault which does not seem to have been considered before, and which defeats fault countermeasures that concentrate on protecting the exponentiations.

This new attack proves very effective in practice: depending on the fault model, between 5 to 45 faults suffice to recover the RSA factorization within a few seconds. Our simplest attack requires that the adversary knows the faulty moduli, but the most sophisticated variant works even if the moduli are unknown, under reasonable fault models. All our attacks have been fully validated experimentally with fault-injection laser techniques.

This work is to be presented at CHES 2011 [BNNT11a] and will appear in the *Journal of Cryptographic Engineering* [BNNT11b].

11.1.1 Fault attacks on RSA-CRT signatures

To sign a message m with RSA [RSA78], the signer first applies an encoding function μ to m, and then computes the signature $\sigma = \mu(m)^d \bmod N$. To verify the signature σ, the receiver checks that $\sigma^e = \mu(m) \mod N$. The Chinese Remainder Theorem (CRT) is often used to speed up signature generation by a factor of about 4. This is done by

computing:

$$\sigma_p = \mu(m)^{d \bmod p-1} \mod p \quad \text{and} \quad \sigma_q = \mu(m)^{d \bmod q-1} \mod q$$

and deriving σ from (σ_p, σ_q) using the CRT. As we discussed in Chapter 10, Boneh, DeMillo and Lipton [BDL97, BDL01] observed in 1997 that RSA-CRT implementations are vulnerable to fault attacks. Assuming that the attacker can induce a fault when σ_q is computed while keeping the computation of σ_p correct, one gets:

$$\sigma_p = \mu(m)^{d \bmod p-1} \mod p \quad \text{and} \quad \sigma_q \neq \mu(m)^{d \bmod q-1} \mod q$$

hence:

$$\sigma^e = \mu(m) \mod p \quad \text{and} \quad \sigma^e \neq \mu(m) \mod q$$

which allows the attacker to factor N by computing $\gcd(\sigma^e - \mu(m) \bmod N, N) = p$. This attack applies to any deterministic padding function μ, such as Full Domain Hash [BR96], or probabilistic signatures where the randomizer used to generate the signature is sent along with the signature, such as PFDH [Cor02]. Only probabilistic signature schemes such that the randomness remains unknown to the attacker may be safe, though some particular cases have been attacked as well, as we saw in the previous chapter.

In 2005, Seifert [Sei05] introduced a new type of RSA fault attacks, by inducing faults on the RSA public modulus. The initial attack [Sei05] only allowed to bypass RSA verification, but key-recovery attacks were later discovered by Brier et al. [BCMCC06], and improved or extended in [Mui06, BCG08, BCDG09, BCDG10]. These key-recovery attacks only apply to RSA without CRT, and they require many more faults than the attack of Boneh et al.—at least on the order of 1000 faulty signatures.

11.1.2 Our contribution

We present new alternative key-recovery attacks on RSA-CRT signatures: instead of targeting one of the RSA-CRT sub-exponentiations, we inject faults into the *public modulus* like in Seifert's attack. As a result, typical countermeasures against the attack of Boneh et al. do not apply to these new attacks.

Our attacks are based on the orthogonal lattice techniques introduced by Nguyen and Stern [NS97] in 1997. They are very effective in practice: they disclose the RSA factorization within a few seconds using only between 5 to 45 faulty signatures. The exact running time and number of faulty signatures depend on the fault model.

For instance, in our simplest attack, the running time is a fraction of a second using only 5 faulty signatures, but the attacker is assumed to know the faulted moduli for the 5 different messages. However, our attack can be extended to the case where the attacker no longer knows the faulted moduli, using at most 45 faulty signatures, under the following two fault models: either the faulted moduli only differ from the public modulus on a single byte of unknown position and unknown value, or the faulted moduli may differ from the public modulus by many bytes, but the differences are restricted to the least significant bits, up to half of the modulus size.

All our attacks have been fully validated with physical experiments with laser shots on a RISC microcontroller.

11.1.3 Related work

Many countermeasures have been proposed to protect against the attack of Boneh et al. and its numerous generalizations, but they often focus on the exponentiation process. The previously mentioned fault attacks [BCMCC06, Mui06, BCG08, BCDG09, BCDG10] on RSA using faulty moduli only apply to standard RSA without CRT, and they use non-lattice techniques. Our attack seems to be the first attack on RSA-CRT with faulted moduli.

It should be pointed out, however, that a number of protected RSA-CRT implementations also protect the CRT recombination. This is for example the case of [ABF+02, CJ05, Gir06, BNP07, Vig08, Riv09].

More generally, as we observe in §11.5, using the technique known as Garner's formula for CRT recombination does thwart the attack introduced in this chapter. Since this formula is often used in practice, typical implementations conforming to RSA standards like PKCS#1 [PKCS#1 v2.1] and IEEE P1363 [IEEE P1363] should in principle be immune to this attack.

11.1.4 Outline

In §11.2, we describe the basic attack where the faulty moduli are assumed to be known to the attacker. In §11.3, we extend the attack to realistic fault models in which the faulty moduli are no longer known to the attacker. In §11.4, we describe physical experiments with laser shots on a RISC microcontroller to validate the attack. Finally, in §11.5, we suggest possible countermeasures against this attack.

11.2 The New Attack

11.2.1 Overview

Consider again the generation of RSA-CRT signatures. To obtain the signature σ of a message m padded as $\mu(m)$, the signer computes the mod-p and mod-q parts:

$$\sigma_p = \mu(m)^d \mod p \quad \text{and} \quad \sigma_q = \mu(m)^d \mod q$$

and returns the signature:

$$\sigma = \sigma_p \cdot \alpha + \sigma_q \cdot \beta \mod N \tag{11.1}$$

where α, β are the pre-computed Chinese Remainder coefficients $\alpha = q \cdot (q^{-1} \bmod p)$ and $\beta = p \cdot (p^{-1} \bmod q)$.

Assume that an adversary can obtain the correct signature σ, and also a signature σ' of the same padded message $\mu(m)$ after corrupting the modulus N before the CRT step (11.1). In other words, the attacker gets σ as before but also σ' defined as:

$$\sigma' = \sigma_p \cdot \alpha + \sigma_q \cdot \beta \mod N' \quad \text{for some } N' \neq N.$$

Suppose further, for the moment, that the adversary is able to recover the faulty modulus N': we will see in §11.3 how this not-so-realistic hypothesis can be lifted in a more practical setting. Then, by applying the Chinese Remainder Theorem to σ and σ', the adversary can compute

$$v = \sigma_p \cdot \alpha + \sigma_q \cdot \beta \mod N \cdot N'.$$

But if we denote the bit length of N by n, then $N \cdot N'$ is a $2n$-bit integer, whereas α, β are of length n and σ_p, σ_q of length $n/2$, so v is really a linear combination of α and β in $\mathbb{Z}$:

$$v = \sigma_p \cdot \alpha + \sigma_q \cdot \beta.$$

That alone does not suffice to factor N, but several such pairs (σ, σ') provide multiple linear combinations of the (unknown) integers α, β with relatively small coefficients. Then lattice reduction techniques allow us to recover the coefficients σ_p and σ_q, and hence obtain the factorization of N by GCDs. The following sections describe this process in detail.

11.2.2 Applying orthogonal lattice techniques

We refer to §10.2 or [Ngu09, NS01] for basic definitions and results on lattices. The cryptanalytic tool we use here is the orthogonal lattice technique introduced by Nguyen and Stern [NS97] and already used in §10.5. If L is a lattice in $\mathbb{Z}^n$, we let $L^\perp$ be the lattice formed by all vectors in $\mathbb{Z}^n$ which are orthogonal to all vectors of L.

Suppose an attacker obtains ℓ pairs (σ, σ') as above. He can compute as before a vector $\boldsymbol{v} = (v_1, \ldots, v_\ell)$ of $3n/2$-bit integers satisfying an equation of the form:

$$\boldsymbol{v} = \alpha \boldsymbol{x} + \beta \boldsymbol{y} \tag{11.2}$$

where $\boldsymbol{x}, \boldsymbol{y}$ are unknown vectors with $n/2$-bit components and α, β are the (unknown) CRT coefficients relative to p and q. Lattice reduction can exploit such a hidden linear relationship as follows.

Using the results from [NS97, NS98], it is possible to compute a reduced basis $\{\boldsymbol{b}_1, \ldots, \boldsymbol{b}_{\ell-1}\}$ of the lattice $\boldsymbol{v}^\perp \subset \mathbb{Z}^\ell$ of vectors orthogonal to $\boldsymbol{v}$ in $\mathbb{Z}^\ell$. In particular we get:

$$\alpha \langle \boldsymbol{b}_j, \boldsymbol{x} \rangle + \beta \langle \boldsymbol{b}_j, \boldsymbol{y} \rangle = 0 \quad \text{for } j = 1, 2, \ldots, \ell - 1.$$

Now, observe that the smallest nonzero solution $(u, v) \in \mathbb{Z}^2$ of the equation $\alpha \cdot u + \beta \cdot v = 0$ is $\pm(\beta, -\alpha)/g$, where $g = \gcd(\alpha, \beta)$ is heuristically expected to be very small, which implies that $|u|, |v| \geqslant \Omega(N)$ where the $\Omega()$ constant is very small. For each $j = 1, 2, \ldots, \ell - 1$, there are thus two possibilities:

Case 1: $\langle \boldsymbol{b}_j, \boldsymbol{x} \rangle = \langle \boldsymbol{b}_j, \boldsymbol{y} \rangle = 0$, in which case $\boldsymbol{b}_j$ belongs to the lattice $L = \{\boldsymbol{x}, \boldsymbol{y}\}^\perp$ of vectors in $\mathbb{Z}^\ell$ orthogonal to both $\boldsymbol{x}$ and $\boldsymbol{y}$;

Case 2: $\langle \boldsymbol{b}_j, \boldsymbol{x} \rangle$ and $\langle \boldsymbol{b}_j, \boldsymbol{y} \rangle$ have absolute value $\geqslant \Omega(N)$ with a very small $\Omega()$ constant. Since $\boldsymbol{x}, \boldsymbol{y}$ both have norm at most $\sqrt{\ell N}$, this implies $\|\boldsymbol{b}_j\| \geqslant \Omega(\sqrt{N/\ell})$ by Cauchy-Schwarz.

Since the lattice $L = \{\boldsymbol{x}, \boldsymbol{y}\}^\perp$ is of rank $\ell - 2$, Case 1 cannot hold for all $\ell - 1$ linearly independent vectors $\boldsymbol{b}_j$, so that the longest one $\boldsymbol{b}_{\ell-1}$ should be in Case 2, and hence $\|\boldsymbol{b}_{\ell-1}\| \geqslant \Omega(\sqrt{N/\ell})$. On the other hand, the other vectors form a lattice of rank $\ell - 2$ and volume:

$$V = \mathrm{vol}(\mathbb{Z}\boldsymbol{b}_1 \oplus \cdots \oplus \mathbb{Z}\boldsymbol{b}_{\ell-2}) \approx \frac{\mathrm{vol}(\boldsymbol{v}^\perp)}{\|\boldsymbol{b}_{\ell-1}\|} = \frac{\|\boldsymbol{v}\|}{\|\boldsymbol{b}_{\ell-1}\|} \leqslant \frac{\sqrt{\ell} \cdot N^{3/2}}{\Omega(\sqrt{N/\ell})} = O(\ell N)$$

which can heuristically be expected to behave like a random lattice. In particular, we should have:

$$\|\boldsymbol{b}_j\| = O(\sqrt{\ell - 2} \cdot V^{1/(\ell-2)}) = O(\ell^{1/2+1/(\ell-2)} \cdot N^{1/(\ell-2)}) \quad \text{for } j = 1, 2, \ldots, \ell - 2.$$

This length is much smaller than $\sqrt{N/\ell}$ as soon as $\ell \geqslant 5$. Assuming that this is case, $\boldsymbol{b}_j$ should thus be in Case 1 for $j = 1, 2, \ldots, \ell - 2$. This means that those vectors generate a sublattice $L' = \mathbb{Z}\boldsymbol{b}_1 \oplus \cdots \oplus \mathbb{Z}\boldsymbol{b}_{\ell-2}$ of full rank in $L = \{\boldsymbol{x}, \boldsymbol{y}\}^\perp$.

Taking orthogonal lattices, we get $(L')^\perp \supset L^\perp = \mathbb{Z}\boldsymbol{x} \oplus \mathbb{Z}\boldsymbol{y}$. Therefore, $\boldsymbol{x}$ and $\boldsymbol{y}$ belong to the orthogonal lattice $(L')^\perp$ of L'. Let $\{\boldsymbol{x'}, \boldsymbol{y'}\}$ be a reduced basis of that lattice. We can enumerate all the lattice vectors in $(L')^\perp$ of length at most $\sqrt{\ell N}$ as linear combinations of $\boldsymbol{x'}$ and $\boldsymbol{y'}$. The Gaussian heuristic suggests that there should be roughly:

$$\frac{\pi(\sqrt{\ell N})^2}{\mathrm{vol}((L')^\perp)} = \frac{\pi \ell N}{V} = O(1)$$

such vectors, so this is certainly feasible. For all those vectors $\boldsymbol{z}$, we can compute $\gcd(\boldsymbol{v} - \boldsymbol{z}, N)$. We will thus quickly find $\gcd(\boldsymbol{v} - \boldsymbol{x}, N)$ among them, since $\boldsymbol{x}$ is a vector of length $\leqslant \sqrt{\ell N}$ in $(L')^\perp$. But by definition of $\boldsymbol{v}$ we have:

$$\boldsymbol{v} = \boldsymbol{x} \mod p \quad \text{and} \quad \boldsymbol{v} = \boldsymbol{y} \mod q$$

so $\gcd(\boldsymbol{v} - \boldsymbol{x}, N) = p$, which reveals the factorization of N.

11.2.3 Attack summary

Assume that, for $\ell \geqslant 5$ padded messages $\mu(m_i)$, we know a correct signature σ_i and a signature σ_i' computed with a faulty modulus N_i'. Then, we can heuristically recover the factorization of N as follows.

1. For each i, compute the integer $v_i = \mathrm{CRT}_{N,N_i'}(\sigma_i, \sigma_i')$. They form a vector $\boldsymbol{v} = (v_1, \ldots, v_\ell) \in \mathbb{Z}^\ell$.
2. Compute an LLL-reduced [LLL82] basis $\{\boldsymbol{b}_1, \ldots, \boldsymbol{b}_{\ell-1}\}$ of the lattice $\boldsymbol{v}^\perp \subset \mathbb{Z}^\ell$ of vectors in $\mathbb{Z}^\ell$ orthogonal to $\boldsymbol{v}$. This is done by applying LLL to the lattice in $\mathbb{Z}^{1+\ell}$ generated by the rows of the following matrix:

$$\begin{pmatrix} \kappa v_1 & 1 & & 0 \\ \vdots & & \ddots & \\ \kappa v_\ell & 0 & & 1 \end{pmatrix}$$

Number of faulty signatures ℓ	4	5	6
1024-bit moduli	48%	100%	100%
1536-bit moduli	45%	100%	100%
2048-bit moduli	46%	100%	100%

Table 11.1: Attack success probability as a function of the number of faulty signatures and the size of N. Each parameter set was tested with random faults on 500 random moduli of the given size.

where κ is a suitably large constant, and removing the first component of each resulting vector [NS97].

3. The first $\ell - 2$ vectors $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_{\ell-2}$ generate a lattice $L' \subset \mathbb{Z}^\ell$ of rank $\ell - 2$. Compute an LLL-reduced basis $\{\boldsymbol{x'}, \boldsymbol{y'}\}$ of the orthogonal lattice $(L')^\perp$ to that lattice. Again, this is done by applying LLL to the lattice in $\mathbb{Z}^{\ell+2+\ell}$ generated by the rows of

$$\begin{pmatrix} \kappa' b_{1,1} & \cdots & \kappa' b_{\ell-2,1} & 1 & & 0 \\ \vdots & & \vdots & & \ddots & \\ \kappa' b_{1,\ell} & \cdots & \kappa' b_{\ell-2,\ell} & 0 & & 1 \end{pmatrix}$$

and keeping the last ℓ components of each resulting vector.

4. Enumerate the vectors $\boldsymbol{z} = a\boldsymbol{x'} + b\boldsymbol{y'} \in (L')^\perp$ of length at most $\sqrt{\ell N}$, and for each such vector $\boldsymbol{z}$, compute $\gcd(\boldsymbol{v} - \boldsymbol{z}, N)$ using all components, and return any nontrivial factor of N.

11.2.4 Simulation results

Since the attack is heuristic, it is important to evaluate its experimental performances. To do so, we have implemented a simulation of the attack in SAGE [S+10b]: for a given modulus N, we compute the vector $\boldsymbol{v}$ corresponding to a series of ℓ signatures on random messages and apply the lattice attack, attempting to recover a factor of N.

Table 11.1 shows the measured success probabilities for various values of ℓ and modulus sizes. It confirms the heuristic prediction that 5 faulty signatures should always suffice to factor N. It turns out that even 4 signatures are enough in almost half the cases.

Experimental running times are given in Table 11.2. The whole attack takes a few dozen milliseconds on a standard PC. The number of vectors to test as part of the final exhaustive search step is about 20 in practice, which is done very quickly.

11.3 Extending the Attack to Unknown Faulty Moduli

As mentioned in §11.2.1, in its basic form, the attack requires the recovery of the faulty moduli N'_i in addition to the corresponding faulty signatures σ'_i. This is not a very

Modulus size	1024	1536	2048
Average search space $\pi\ell N/V$	24	23	24
Average total CPU time	16 ms	26 ms	34 ms

Table 11.2: Efficiency of the attack with $\ell = 5$ faulty signatures and various modulus sizes. Each parameter set was tested with random faults on 500 random moduli of the given size. Timings for a SAGE implementation, on a single 2.4 GHz Core2 CPU core.

realistic assumption, since a typical implementation does not output the public modulus along with each signature.

To work around this limitation, we would like to reconstruct the vector $\boldsymbol{v}$ of integer values needed to run the attack from signatures alone, without the knowledge of the faulty moduli—possibly at the cost of requiring a few more faulty signatures.

This can actually be achieved in various ways depending on the precise form of the faults inflicted to the modulus. We propose solutions for the following two realistic fault models:

1. The faulty moduli N'_i differ from N on a single (unknown) byte. This is known to be possible using power glitches or laser shots with small aperture.

2. The differences between the faulty moduli N'_i and N are located on the least significant half: the errors on the least significant bits can be up to half of the modulus size. It is easy to obtain such faults with a laser or a cold boot attack.

11.3.1 Single byte faults

In this model, the attacker is able to obtain a certain number $\ell' \geqslant 5$ of pairs (σ_i, σ'_i) where $\sigma_i = \alpha x_i + \beta y_i \bmod N$ is a valid signature and $\sigma'_i = \alpha x_i + \beta y_i \bmod N'_i$ is the same signature computed with a faulty modulus. The faulty moduli N'_i are not known, but they only differ from N on a single byte whose position and value is unknown.

This type of fault can for example occur when attacking the transfer of the modulus to memory on a smart card with an 8-bit processor, or when using a laser attack with a sufficiently focused beam.

For a 1024-bit modulus N, for example, there are $128 \times 255 \approx 2^{15}$ possible faulty moduli. It can thus seem like a reasonable approach to try and run the attack with all possible faults. However, since this should be done with 5 signatures, this results in a search space of size $\approx (2^{15})^5 = 2^{75}$ which is prohibitive.

This kind of exhaustive search can be made practical, though, if we take into account the fact that the CRT value $v_i = \mathrm{CRT}_{N,N'_i}(\sigma_i, \sigma'_i)$ satisfies:

$$v_i = \alpha x_i + \beta y_i \leqslant N \cdot (p+q) = N^{3/2}\left(\sqrt{\frac{p}{q}} + \sqrt{\frac{q}{p}}\right) < (2N)^{3/2}$$

Number of pairs ℓ'	5	7	10	15	20	25
Search space size (bits)	11.6	9.8	7.2	6.2	4.2	2.6
Total attack time (seconds)	49	14	2.4	1.2	0.29	0.10

Table 11.3: Exhaustive search space size for the vector $\boldsymbol{v}$ of CRT values, and average attack running time, depending on the number of pairs (σ_i, σ'_i) available to the attacker. Measured for a family of random single byte faults on a 1024-bit modulus. Timings are given for the SAGE implementation as above.

since $p/q \in (1/2, 2)$. Now, for a given value of σ'_i, there are only very few possible target moduli N_i^* differing from N on a single byte such that $v_i^* = \mathrm{CRT}_{N,N_i^*}(\sigma_i, \sigma'_i) < (2N)^{3/2}$: often only one or two, and almost never more than 20. We only need to run the attack with those target v_i^*'s until we find a factor.

Experimentally, for a 1024-bit modulus, the average base 2 logarithm of the number of possible v_i^*'s is about 2.5, so if an attacker has 5 pairs (σ_i, σ'_i) in this model, they can expect to try all vectors $\boldsymbol{v}$ in a search space of around 12.5 bits, i.e. run the attack a few thousand times, for a total running time of under 2 minutes. This is already quite practical.

If more pairs are available, the attacker can keep the 5 pairs for which the number of possible v_i^*'s is the smallest. This reduces the search space accordingly. In Table 11.3, we show how the exhaustive search space size and the average running time evolve with the number of signatures in a typical example.

11.3.2 Faults on many least significant bits

In this model, the attacker is able to obtain $\ell = 5$ signature families of the form $(\sigma_i, \sigma'_{i,1}, \ldots, \sigma'_{i,k})$, where the σ_i's are correct signatures:

$$\sigma_i = \alpha x_i + \beta y_i \bmod N$$

and the $\sigma'_{i,j}$'s are faulty signatures of the form:

$$\sigma'_{i,j} = \alpha x_i + \beta y_i \bmod N'_{i,j} \quad 1 \leqslant i \leqslant \ell,\ 1 \leqslant j \leqslant k.$$

In other words, for each one of the ℓ different messages, the attacker learns the reduction of the CRT value $v_i = \alpha x_i + \beta y_i$ modulo N, as well as modulo k different unknown faulty moduli $N'_{i,j}$. Additionally, it is assumed that all $N'_{i,j}$ differ from N only on the least significant bits, but the number of distinct bits can be as large as half of the modulus size: we assume that$|N - N'_{i,j}| < N^\delta$ for a certain constant $\delta < 1/2$.

This is a reasonable fault model for a laser attack: it suffices to target a laser beam on the least significant bits of N to produce this type of faults.

To run the attack successfully, the attacker needs to recover the CRT values v_i. This can be done with high probability when the number of available faults k for a given message is large enough. The simplest approach is based on a GCD computation.

k (faults per message)	3	5	7	9
$1/\zeta(k)$	.832	.964	.992	.998
100-bit errors	83.2%	96.8%	99.0%	99.8%
200-bit errors	83.4%	96.2%	99.2%	99.8%
400-bit errors	82.7%	96.6%	99.1%	99.8%
Average CPU time	.73 ms	.75 ms	.79 ms	.85 ms

Table 11.4: Success probabilities of the GCD method for CRT value recovery, depending on the number of available faults on a given message. Tested with random 1024-bit moduli. In the simulation, errors ε_j are modeled as uniformly random signed integers of the given size, and 10,000 of them were generated for each parameter set.

Indeed, fix an index $i \in \{1, \ldots, \ell\}$, and write $N'_{i,j} = N + \varepsilon_j$, $v_i = u$, $\sigma_i = u_0$ and $\sigma'_{i,j} = u_j$. The attacker knows the u_j's and wants to recover u.

Now, observe that there are integers t_j such that u satisfies $u = u_0 + t_0 \cdot N$ and $u = u_j + t_j \cdot (N + \varepsilon_j)$. In particular, for $j = 1, \ldots, k$ we can write:

$$(t_j - t_0) \cdot N + (u_j - u_0) + t_j \cdot \varepsilon_j = 0. \tag{11.3}$$

This implies that $u_j - u_0 \equiv t_j \cdot \varepsilon_j \pmod{N}$. However, we have $t_j \cdot \varepsilon_j < N^{1/2+\delta} \ll N$, so that the congruence is really an equality in $\mathbb{Z}$. In view of (11.3), this implies that all t_j's are in fact equal, and hence:

$$t_0 \cdot \varepsilon_j = u_0 - u_j \quad 1 \leqslant j \leqslant k.$$

If the errors ε_j on the modulus are co-prime, which we expect to happen with probability $\approx 1/\zeta(k)$, we can then deduce t_0 as the GCD of all values $u_0 - u_j$, and this gives:

$$u = u_0 + t_0 \cdot N = u_0 + N \cdot \gcd(u_0 - u_1, \ldots, u_0 - u_k).$$

As seen in Table 11.4, the success probability is in practice very close to $1/\zeta(k)$ regardless of the size of errors.

It is probably possible to further improve the success probability by trying to remove small factors from the computed GCD $g = \gcd(u_0 - u_1, \ldots, u_0 - u_k)$ to find t_0 when $g > \sqrt{N}$, but we find that the number of required faults is already reasonable without this computational refinement.

Indeed, recall that $\ell = 5$ CRT values are required to run the attack. If k faults are obtained for each of the ℓ messages, the probability that these ℓ CRT values can be successfully recovered with this GCD approach is $\zeta(k)^{-\ell}$. This is greater than 95% for $k = 7$, and 99% for $k = 9$.

We can also mention an alternate, lattice-based approach to recovering the CRT value u. The relation between the different quantities above can be written in vector form as:

$$u_0 \mathbf{1} = \boldsymbol{u} + t_0 \boldsymbol{e}$$

where $\mathbf{1} = (1, \ldots, 1)$, $\boldsymbol{u} = (u_1, \ldots, u_k)$ and $\boldsymbol{e} = (\varepsilon_1, \ldots, \varepsilon_k)$.

Then, since $u_0 \approx N$ is much larger than $\|t_0\boldsymbol{e}\| \approx N^{1/2+\delta}$, short vectors orthogonal to $\boldsymbol{u}$ will be orthogonal to both $\mathbf{1}$ and $\boldsymbol{e}$. More precisely, we can heuristically expect that when k is large enough ($k \gtrsim 2/(1-2\delta)$), the first $k-2$ vectors of a reduced basis of $\boldsymbol{u}^\perp$ will be orthogonal to $\mathbf{1}$ and $\boldsymbol{e}$.

Taking orthogonal lattices again, we can thus obtain a reduced basis $\{\boldsymbol{x}, \boldsymbol{y}\}$ of a two-dimensional lattice containing $\mathbf{1}$ and $\boldsymbol{e}$ (and of course $\boldsymbol{u}$). Since $\mathbf{1}$ is really short, we always find that $\boldsymbol{x} = \mathbf{1}$ in practice. Then, it happens quite often that $\boldsymbol{y}$ can be written as $\lambda\mathbf{1} \pm \boldsymbol{e}$, in which case t_0 is readily recovered as the absolute value of the second coordinate of $\boldsymbol{u}$ in the basis $\{\boldsymbol{x}, \boldsymbol{y}\}$.

However, this fails when $\mathbb{Z}\mathbf{1} \oplus \mathbb{Z}\boldsymbol{e}$ is a proper sublattice of $\mathbb{Z}\boldsymbol{x} \oplus \mathbb{Z}\boldsymbol{y} = \mathbb{Z}^k \cap (\mathbb{Q}\mathbf{1} \oplus \mathbb{Q}\boldsymbol{e})$, namely, when there is some integer $d > 1$ such that all errors ε_j are congruent mod d. Thus, we expect the success probability of this alternate approach to be $1/\zeta(k-1)$, which is slightly less than with the GCD approach.

11.4 Practical Experiments

Practical experiments for validating the new attack were done on an 8-bit 0.35μm RISC microcontroller with no countermeasures. As the microprocessor had no arithmetic coprocessor the values σ_p and σ_q were pre-computed by an external program upon each fault-injection experience and fed into the attacked device. The target combined σ_p and σ_q using multiplications and additions (using Formula 11.1) as well as the final modular reduction.

The location and spread of the faults were controlled by careful beam-size and shot-instant tuning. The reader is referred to Appendix app:laser for a description of the physical setting (common to the experiments reported in [MDT+10]).

We conducted several practical experiments corresponding to three different scenarios, roughly corresponding to the fault models considered in §11.2.1, §11.3.1 and §11.3.2 respectively. Let us describe these experiments in order.

11.4.1 First scenario: Known modulus

In this case, we considered 5 messages for a random 1024-bit RSA modulus N. For each message m_i, we obtained a correct signature σ_i, as well as a faulty-modulus signature σ_i' where the faulty modulus N_i' was also read back from the microcontroller.

Therefore, we were exactly in the setting described in §11.2.1, and could apply the algorithm from §11.2.3 directly: apply the Chinese Remainder Theorem to construct the vector $\boldsymbol{v}$ of CRT values and run the lattice-based attack to recover a factor of N.

The implementation of the attack used the same SAGE code as the simulation from §11.2.4. In our experimental case, the ball of radius $\sqrt{N\ell}$ contained only about 10 vectors of the double orthogonal lattice, and the whole attack revealed a factor of N in less than 20 milliseconds.

11.4.2 Second scenario: Unknown single byte faults

In this case, we tried to replicate a setting similar to the one considered in §11.3.1. We considered 20 messages and a random 1024-bit RSA modulus N. For each message m_i, we obtained a correct signature σ_i, as well as faulty-modulus signatures σ'_i with undisclosed faulty modulus N'_i generated by targeting a single byte of N with the laser.

We had to eliminate some signatures, however, because in some cases, errors on the modulus turned out to exceed 8 bits.[1] After discarding those, we had 12 pairs (σ_i, σ'_i) left to carry out the approach described in §11.3.1.

The first step in this approach is to find, for each i, all values v_i^* of the form $\mathrm{CRT}_{N,N_i^*}(\sigma_i, \sigma'_i)$ (N_i^* differing from N only on one byte) that are small enough to be correct candidate CRT values. Unlike the setting of §11.3.1, we could not assume that bit-differences were aligned on byte boundaries: we had to test a whole 1016×255 candidate moduli[2] N_i^* for each i . Therefore, this search step was a bit costly, taking a total of 11 minutes and 13 seconds. Additionally, due to the higher number of candidate moduli, the number of candidate CRT values v_i^* was also somewhat larger than in §11.3.1, namely:

$$7, 17, 3, 9, 15, 5, 14, 44, 44, 17, 10, 55$$

for our 12 pairs respectively. Keeping only the 5 indices with the smallest number of candidates, we obtained $3 \times 5 \times 7 \times 9 \times 10 = 9450$ possible CRT value vectors $\boldsymbol{v}^*$.

We then ran the lattice-based attack on each of these vectors in order until a factor of N was found. The factor was found at iteration number 2120, after a total computation time of 43 seconds.

11.4.3 Third scenario: Least significant bytes faults

In this case, we considered 10 messages for a random 1024-bit N. For each message m_i, we obtained a correct signature σ_i, as well as 10 faulty-modulus signature $\sigma'_{i,j}$ with undisclosed faulty modulus N'_i. The laser beam targeted the lower order bytes of N but with a large aperture, generating multiple faults stretching over as much as 448 modulus bits.

In practice, we only used the data $(\sigma_i, \sigma'_{i,1}, \ldots, \sigma'_{i,10})$ for the first 5 messages, discarding the rest. Then, we reconstructed the CRT values v_i using the GCD technique of §11.3.2:

$$v_i = \sigma_i + N \cdot \gcd(\sigma_i - \sigma'_{i,1}, \ldots, \sigma_i - \sigma'_{i,10}) \quad 1 \leqslant i \leqslant 5$$

and applied the lattice-based attack on the resulting vector $\boldsymbol{v}$. This revealed a factor of N in 16 milliseconds.

[1] Note that in a real-world attack, it might not be possible to detect such overly spread out faults: hence, this particular technique should be used preferably when faults are *known* to affect only single bytes (e.g. in a glitch attack), whereas the technique from the next section is better suited to laser attacks as aperture control is much less of an issue.

[2] There are duplicates among those, corresponding to perturbations of 7 consecutive bits or less, but we did not attempt to avoid testing them several times, as this can only improve the search by a small constant factor while introducing significant complexity in the code.

We also tried the same attack using a fewer number of the $\sigma'_{i,j}$'s, and found that it still worked when taking only 4 of those values in the computation of v_i:

$$v_i = \sigma_i + N \cdot \gcd(\sigma_i - \sigma'_{i,1}, \ldots, \sigma_i - \sigma'_{i,4}) \quad 1 \leqslant i \leqslant 5$$

but failed if we took 3 instead. Considering that $1/\zeta(3)^5 \approx .40$ and $1/\zeta(4)^5 \approx .67$, this is quite in line with expectations.

11.5 Countermeasures and Further Research

Probabilistic and stateful signature schemes are usually secure against this attack, since they make it difficult to obtain two signatures on the same padded message. However, all deterministic schemes are typically vulnerable, including those in which the attacker doesn't have full access to the signed message, provided that the target device can be forced to compute the same signature twice.

A natural countermeasure is to use a CRT interpolation formula that does not require N, such as Garner's formula, computed as follows:

$t \leftarrow \sigma_p - \sigma_q$
if $t < 0$ then $t \leftarrow t + p$
$\sigma \leftarrow \sigma_q + (t \cdot \gamma \bmod p) \cdot q$
return(σ)

where we assume that $p > q$, and γ is the usual CRT coefficient $q^{-1} \bmod p$. Note that the evaluation of σ does not require a modular reduction because

$$\sigma = \sigma_q + (t \cdot \gamma \bmod p) \cdot q \leqslant q - 1 + (p-1)q < N.$$

Besides the obvious countermeasures consisting in checking signatures before release, it would be interesting to devise specific countermeasures for protecting Formula (11.1) (or Garner's formula) taking into account the possible corruption of all data involved.

11.A Laser Fault Injection

Laser (Light Amplification by Stimulated Emission of Radiation) is a stimulated-emission electromagnetic radiation in the visible or the invisible domain. Laser light is monochromatic, unidirectional, coherent and artificial (i.e. laser does not spontaneously exist in nature). Laser light can be generated as a beam of very small diameter (a few μm). The beam can pass through various material obstacles before impacting a target during a very short duration.

Laser impacts on electronic circuits are known to alter functioning. Current chip manufacturing technologies are in the nanometers range. This, and the laser's brief and precise reaction time, makes laser a particularly suitable fault injection means.

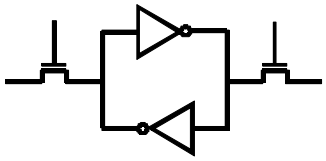


Figure 11.1: Architecture of a typical SRAM cell.

11.A.1 Photoelectric effects of laser on silicon

SRAM (Static Random Access Memory) laser exposure is known to cause bit-flips [SA02], [DBP+02], [BECN+06], [Can09], a phenomenon called *Single Event Upset* (SEU). By tuning the beam's energy level below a destructive threshold, the target will not suffer any permanent damage.

A conventional one-bit SRAM cell (Figure 11.1) is made of two cross-coupled inverters. Every cell has two additional transistors controlling the cell's content access during write and read. As every inverter is made of two transistors, an SRAM cell contains six MOS.

In each cell, the states of four transistors encode the stored value. By design, the cell admits only two stable states: a "0" or a "1". In each stable state, two transistors are at an ON state and two others are OFF.

If a laser beam hits the drain/bulk reversed-biased PN junctions of a blocked transistor, the beam's energy may create pairs of electrons as the beam passes through the silicon. The charge carriers induced in the collection volume of the drain-substrate junction of the blocked transistor are collected and create a transient current that inverts logically the inverter's output voltage. This voltage inversion is in turn applied to the second inverter that switches to its opposite state: all in all, a bit flip happens [DBP+02], [BECN+06].

From the opponent's perspective, an additional advantage of laser fault injection is *reproducibility*. Identical faults can be repeated by carefully tuning the laser's parameters and the target's operating conditions.

11.A.2 Different parameters in a fault attack by laser

In a laser attack, the opponent usually controls the beam's diameter, wavelength, amount of emitted energy, impact coordinates (attacked circuit part) and the exposure's duration. Sometimes, the opponent may also control the synchronicity of the impact with a given clock cycle of the target, the target's clock frequency, V_{cc} and temperature. Finally, laser attacks may indifferently target the chip's front side or back side.

However, the chip's front and back sides have different characteristics when exposed to a laser beam.

Front side attacks are particularly suited to green wavelength ($\sim$ 532nm). The visibility of chips components makes positioning very easy in comparison to backside attacks. But because of the metallic interconnects' reflective effect, it is difficult to target a component with enough accuracy. In addition, progress in manufacturing technolo-

gies results in both a proliferation of metal interconnects and much smaller chips. All in all, it becomes increasingly difficult to hit a target area.

Backside attacks are more successful at the infrared wavelength ($\sim$ 1064nm) as the laser needs to deeply enter the silicon. Positioning is more difficult for lack of visibility. Nonetheless, backside attacks allow to circumvent the reflective problem of metallic surfaces.

11.A.3 Practical CRT fault injection

After chip decapsulation and a mapping of the chip's components, we selected a large target area, given our knowledge of the implementation. Using automated search on the chip's front-side, we modified the impact's coordinates, the beam parameters and timing until a reproducible fault area was obtained.

The target is an 8-bit 0.35 μm 16 MHz RISC microcontroller with an integrated 4KB SRAM and no countermeasures. The device runs SOSSE (Simple Operating System for Smartcard Education [B$^+$03]) to which we added some commands, most notably for feeding-in the data $\{N, p, q, \sigma_p, \sigma_q, p^{-1} \bmod q, q^{-1} \bmod p\}$. Upon reception all these parameters are stored in SRAM. The laser, shown in Figures 11.6 and 11.7, is equipped with a YAG laser emitter in three different wavelengths: green, infrared and ultraviolet.

The spot's diameter can be set between 0 and 2500μm. As the beam passes through a lens, it gets reduced by the lens' zoom factor and loses a big part of its energy. Our experiments were conducted with a 20$\times$ Mitutoyo lens, a green[3] beam of $\varnothing 4\mu$m and $\simeq$ 15pJ per shot[4]. The circuit is installed on a programmable Prior Scientific X-Y positioning table[5]. The X-Y table, card reader, laser and an FPGA trigger board, were connected via RS-232 to a control PC. The FPGA trigger board receives an activation signal from the reader and sends a trigger signal to the laser after a delay defined by the control PC.

Experiments were conducted in ambient temperature and at $V_{\text{cc}} = 5\text{V}$. These parameters are within the device's normal operating conditions $2.7\text{V} \leqslant V_{\text{cc}} \leqslant 5.5\text{V}$.

The chip was decapsulated by chemical etching using a Nisene JetEtch automated acid decapsulator. The decapsulator can be programmed for the chemical opening of different chip types using different ratios of nitric acid (HNO_3) and sulfuric acid (H_2SO_4), at a desired temperature and during a specified time. For opening our chip, we used only nitric acid at 80°C for 40 seconds. The etched chip (Figure 11.2) successfully passed functional tests before and during fault injection.

As it is very difficult to target the chip's ALU (Arithmetic Logic Unit) during a very specific calculation step, we decided to hit the SRAM to corrupt data rather than modify calculations.

[3] 532nm wavelength.

[4] At the laser source emitter, before passing through the lens.

[5] Motorized stepper stage for upright microscopes with 0,1 μm steps.

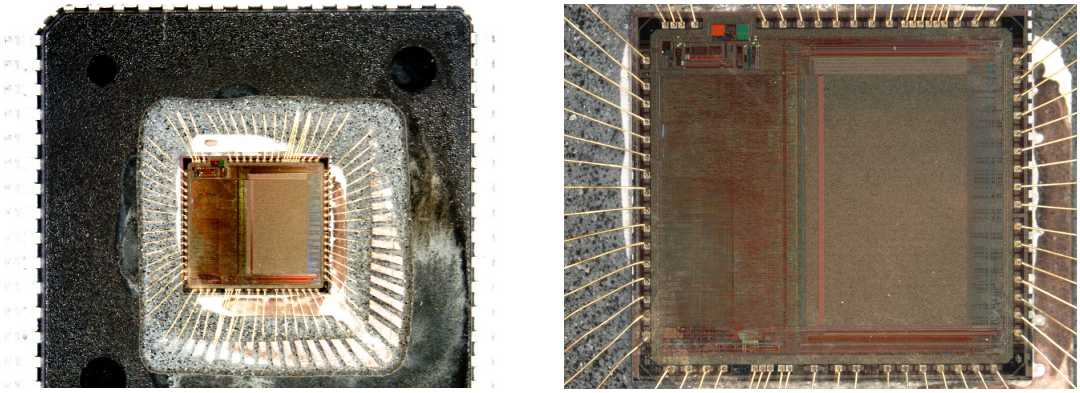


Figure 11.2: Decapsulated chip (SRAM is on the left middle and bottom side).

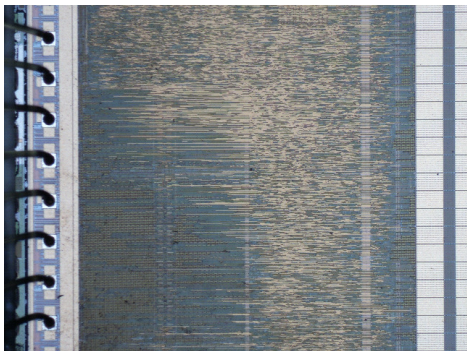


Figure 11.3: Decapsulated chip (closeup on SRAM).

Finding the SRAM area containing each data element and properly tuning the laser's parameters is very time consuming. The number of faults in the read back data, their position and their contents indicate which element has been hit.

Figure 11.5 compares a 1μm laser spot and SRAM cells in different technology sizes. As technology advances, transistor density per μm grows. With several transistors are packed into 1μm areas, single-bit fault injection will require much more precise equipment and are likely to become unfeasible using cheap lasers.

Figure 11.4 shows how we explored the target SRAM space. The method consisted in searching N's precise storage area, shooting into it and reading data back. Figure 11.4 is just a schematic model of the real SRAM (Figure 11.3) to describe our technique and does not correspond to real address allocation. We could successfully inject multiple byte faults into selected parts of N and iterate the process for different moduli and signatures. This sufficed to implement our scenario.

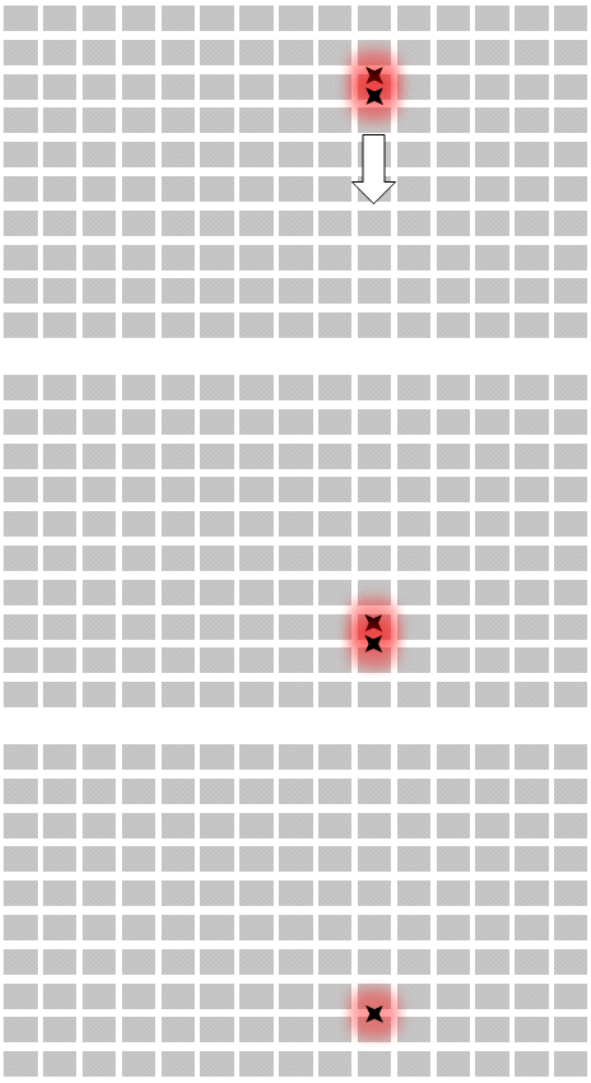


Figure 11.4: Exploration process.

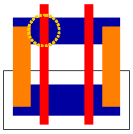
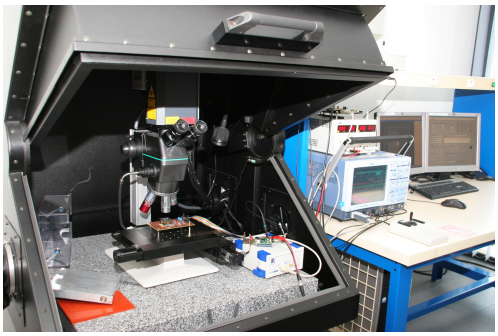
	Transistor	SRAM Cell
350nm		
130nm		
90nm		
65nm		

Figure 11.5: 1μm laser spot (dotted circle) vs. technology sizes [Pou07].



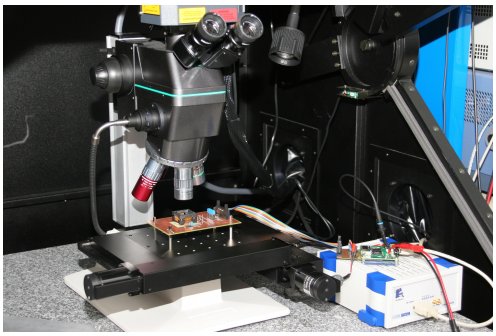


Figure 11.6: Laser and target (general overview).

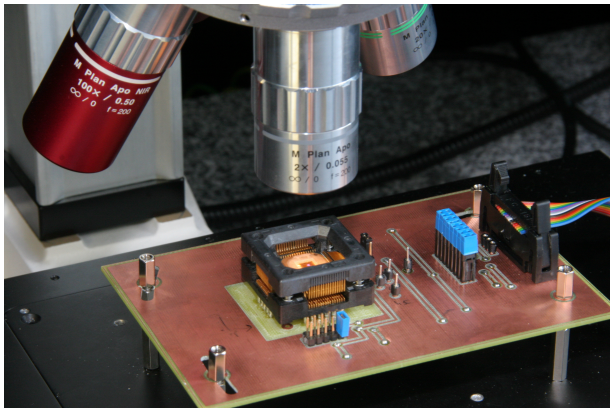


Figure 11.7: Laser and target (closeup).

Chapter 12

On the Broadcast and Validity-Checking Security of PKCS#1 v1.5 Encryption

12.1 Introduction

Returning to black-box model attacks on ad-hoc RSA paddings, this chapter describes new attacks on PKCS#1 v1.5 encryption, a deprecated but still widely used RSA encryption standard.

The first attack predicts, using a single query to a validity checking oracle, which of two chosen plaintexts corresponds to a challenge ciphertext (in other words, it breaks semantic security under validity-checking attacks). The rather simple idea is that a validity-checking oracle makes it possible to check when the least significant bits of the plaintext are zero, an observation that essentially goes back to Bleichenbacher's famous 1998 attack on SSL.

The second attack is a so-called *broadcast attack*, allowing the opponent to reveal an identical plaintext sent to different recipients, which is nontrivial because different random nonces are used for different encryptions (i.e. plaintexts coincide only partially). It relies on multivariate generalizations Coppersmith's technique for finding small roots of polynomials over $\mathbb{Z}_N$.

This work was presented at ACNS 2010 [BCN+10].

12.1.1 The PKCS#1 v1.5 standard

PKCS stands for *Public-Key Cryptography Standards*, and refers to a large body of specifications covering RSA encryption, Diffie-Hellman key agreement, password-based encryption, syntax (extended-certificates, cryptographic messages, private-key information and certification requests) and selected attributes. PKCS was initially developed by RSA Laboratories in collaboration with academia and multiple industry players, and has been

regularly updated since. Today, PKCS has become part of several standards and of a wide range of security products including S/MIME (the standard for encrypted e-mails).

Within the PKCS collection, PKCS#1 v1.5 [PKCS#1 v1.5] describes a particular padding scheme for RSA encryption, in which the secret data (usually a random symmetric key for a block cipher) is prepended with random bytes before applying the RSA function.

At CRYPTO 1998, Bleichenbacher [Ble98] presented an adaptive chosen-ciphertext attack against PKCS#1 v1.5 capable of recovering arbitrary plaintexts from about half a million ciphertexts. Although active adversarial models are generally regarded as theoretical concerns, Bleichenbacher's attack makes use of an oracle that only detects conformance with respect to the padding format, a real-life assumption that resulted in a practical threat. PKCS#1 v1.5 was subsequently updated (release 2.0 [PKCS#1 v2.0]) and patches were issued to users wishing to continue using the old version of the standard. Over a decade later and despite its vulnerabilities, PKCS#1 v1.5 remains widely used in applications. Provably secure algorithms such as RSA-OAEP [BR94, FOPS04] and RSA-KEM [CS03, ISO18033–2, ANSI X9.44] are recommended replacements, but their deployment remains limited.

Another more recent work by Coron, Joye, Naccache and Paillier on the security of PKCS#1 v1.5 [CJNP00] presented a somewhat atypical attack allowing the opponent to retrieve plaintexts ending by enough zero bits.

12.1.2 Our results

In this chapter, we look again at the security of PKCS#1 v1.5 in various adversarial models, and mainly describes two new weaknesses in PKCS#1 v1.5:

- The possibility to predict, using a single validity-checking query and with high probability, which of two chosen plaintexts corresponds to a challenge ciphertext. In other words, we show that PKCS#1 v1.5 encryption in not 1-IND-VCA-secure. The proof is quite elementary.

- A broadcast attack allowing to decrypt an identical message sent to several recipients. The attack is based on multivariate extensions of Coppersmith's technique for finding small roots of polynomials modulo an integer with unknown factorization.

We also obtain some additional results, including a (rather weak) positive security result on the one-wayness of PKCS#1 v1.5, and a proof that PKCS#1 v1.5 encryption is malleable.

12.2 Preliminaries

12.2.1 Public-key encryption

A public-key encryption scheme $\mathscr{P} = (\mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$ is a collection of three efficient algorithms (the first two can be probabilistic):

$\mathsf{KeyGen}(1^k)$: produces a pair (pk, sk) of matching public and private keys depending on the security parameter k (which implicitly define a message space $\mathcal{M}$).

$\mathsf{Encrypt}(pk, m \in \mathcal{M})$: produces a ciphertext $c = \mathsf{Encrypt}(pk, m)$.

$\mathsf{Decrypt}(sk, c)$: produces a plaintext $m = \mathsf{Decrypt}(sk, c)$ or $\perp$.

For all $m \in \mathcal{M}$ and for properly generated $\{pk, sk\}$, the following correctness condition must hold:

$$\mathsf{Decrypt}(sk, \mathsf{Encrypt}(pk, m)) = m.$$

12.2.2 Security definitions

Security is traditionally defined by combining an *adversarial goal* and an *attack model*. We refer to classical texts on provable security, such as [Poi05], for precise statements of security definitions. Intuitively, a public-key encryption scheme $\mathscr{P}$ is:

Unbreakable (UBK): if no efficient adversary $\mathscr{A}$ can compute sk given pk with significant probability;

One-Way (OW): if no efficient adversary $\mathscr{A}$ can recover $m \in \mathcal{M}$ given pk and $c = \mathsf{Encrypt}(pk, m)$ with significant probability;

Indistinguishable (IND): if no efficient adversary $\mathscr{A}$, given pk and a ciphertext computed from one of two messages of his choice, can guess with significant probability which of the two messages was encrypted (this is also referred to as "semantic security", as it implies no information on the plaintext can be recovered from a ciphertext by an efficient adversary);

Non-Malleable (NM): if no adversary $\mathscr{A}$ can produce, given pk and a ciphertext $c = \mathsf{Encrypt}(pk, m)$ corresponding to a message m chosen according to a distribution of his choice (with positive entropy), a new ciphertext $c' \neq c$ corresponding to a plaintext m' meaningfully related to m (in the sense that $\mathcal{R}(m, m') = 1$ with good probability, for some binary relation $\mathcal{R}$ initially chosen by $\mathscr{A}$ as well).

Security notions for encryption schemes are obtained by combining an adversarial goal with an *attack model*, which is essentially the definition of an oracle that $\mathscr{A}$ is given access to. In this chapter, we consider the following attack models:

Chosen-Plaintext Attack (CPA): $\mathscr{A}$ is given nothing more than pk.

Validating-Checking Attack (VCA): $\mathscr{A}$ is given access to a *validity-checking oracle*, indicating if a given ciphertext c is a valid or not (i.e. returning the bit $\mathsf{Decrypt}(sk, c) \stackrel{?}{=} \perp$). Note that this is different from the notion of Plaintext-Checking Attack (PCA) [Poi05].

Chosen-Ciphertext Attack (CCA): $\mathscr{A}$ has access to a decryption oracle.

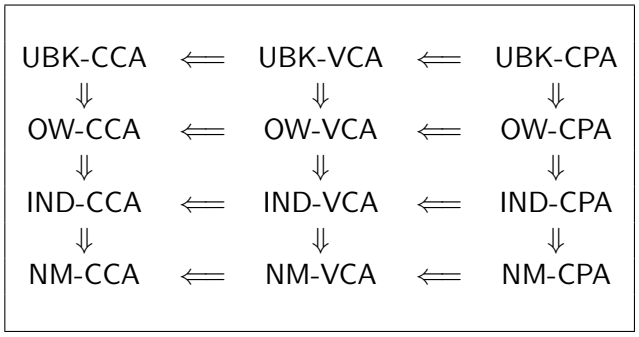


Figure 12.1: Public-key encryption security hierarchy.

We denote security notions positively: e.g. OW-VCA[$\mathscr{P}$] is the problem of contradicting the one-wayness of scheme $\mathscr{P}$ under validity-checking attack. This convention permits the easy description of hierarchies between security notions using reductions, as illustrated in Figure 12.1. When oracle access is permitted (i.e. either for VCA or CCA), we denote by ℓ-GOAL-ATTACK[$\mathscr{P}$] the problem of contradicting GOAL $\in \{\mathsf{UBK}, \mathsf{OW}, \mathsf{IND}, \mathsf{NM}\}$ in less than ℓ oracle queries under an ATTACK $\in \{\mathsf{VCA}, \mathsf{CCA}\}$.

12.2.3 RSA security

Single-user RSA security. RSA encryption as originally described by Rivest, Shamir and Adleman [RSA78] (also known as "textbook RSA encryption") is UBK-CPA-secure assuming the hardness of factoring. Contradicting the OW-CPA security of textbook RSA is known as *the RSA problem* and is believed to be intractable. In fact, no better attack is known on the RSA problem than on factoring.

The validity of a textbook RSA ciphertext can be checked publicly, so in the case of textbook RSA, VCA and CPA are equivalent adversarial models. In particular, the scheme is OW-VCA-secure under the hardness of the RSA problem as well.

Since textbook RSA encryption is deterministic, on the other hand, it cannot be semantically secure: given the encryption c of either m_0 or m_1, the adversary simply computes $c_0 = m_0^e \bmod N$ and checks whether $c_0 = c$.

Many variants of this basic scheme have been proposed in order to achieve better security properties (especially semantic security) through the use of randomized padding schemes. PKCS#1 v1.5 is an early example of such a randomized padding, and this chapter is devoted to studying its security under the various security notions mentioned above.

Multi-user RSA security. The previous definitions do not capture security in a multicast context. In the early 1990s, attacks against low-exponent RSA highlighted the danger of multi-user security threats left uncaptured by single-user models. In independent works, Baudron, Pointcheval and Stern [BPS00] and Bellare, Boldyreva and Micali [BBM00]

proved that for strong security notions (namely IND or NM), schemes with reductionist security in the multi-user setting are exactly those permitting security proofs in the single-user setting. However, encryption one-wayness (OW) in a multi-user setting is not guaranteed by single-user one-wayness.

For textbook RSA encryption, if a common public exponent e is used, then e encryptions of a given message m under different public keys $N_1, \ldots, N_e$ lead to an easy plaintext recovery (indeed, applying the Chinese Remainder Theorem reveals $m^e \bmod N_1 N_2 \cdots N_e$, but this value is just m^e in $\mathbb{Z}$): see [Bon99, §4.2]. Again, several countermeasures have been proposed (e.g. padding the message with random bits) but with an unclear security guarantees. The second part of this chapter explores broadcast attacks on PKCS#1 v1.5.

12.3 PKCS#1 v1.5 Encryption

12.3.1 The PKCS#1 v1.5 encoding function

PKCS#1 v1.5 describes a particular padding scheme for RSA encryption. Consider an RSA modulus N, and let k denote its byte-length (i.e. $2^{8(k-1)} < N < 2^{8k}$). Let m be an $|m|$-byte message with $|m| < k - 11$. A PKCS#1 v1.5 padding $\mu(m)$ of m is obtain as follows. A random nonce r consisting in $k - 3 - |m| \geqslant 8$ nonzero bytes is first generated uniformly at random. Then, $\mu(m) = \mu(m, r)$ is the integer converted from the octet-string:

$$\mu(m, r) = 0002_{16} \| r \| 00_{16} \| m. \tag{12.1}$$

The leading 00 byte guarantees that the encryption block is an integer smaller than N.

The encryption of a message m of $|m| < k - 11$ bytes is then computed as:

$$c = \mu(m, r)^e \bmod N$$

where r is the random nonce chosen as above.

To decrypt $c \in \mathbb{Z}_N^*$, compute $c^d \bmod N$, convert the result to a k-byte octet-string and parse it according to equation (12.1). If the string cannot be parsed correctly or if r is shorter than 8 octets, the decryption algorithm Decrypt outputs $\perp$; otherwise, Decrypt outputs the plaintext m.

12.3.2 Previous attacks on PKCS#1 v1.5

In 1998, Bleichenbacher [Ble98] published an attack on PKCS#1 v1.5 capable of recovering arbitrary plaintexts from a large number ℓ of ciphertexts validation queries. This attack established that PKCS#1 v1.5 is not ℓ-GOAL-ATTACK for a large[1] ℓ, GOAL $\in$ {OW, IND, NM} and ATTACK $\in$ {VCA, CCA} (see Appendix 12.2.2 for definitions of these security notions).

In 2000, Coron, Naccache, Joye and Paillier [CJNP00] introduced two new CPAs on PKCS#1 v1.5. The first attack can be considered as a IND-CPA when e is small (for

[1] $3 \cdot 10^5 < \ell < 2 \cdot 10^6$ for $512 < \log_2 N < 1024$.

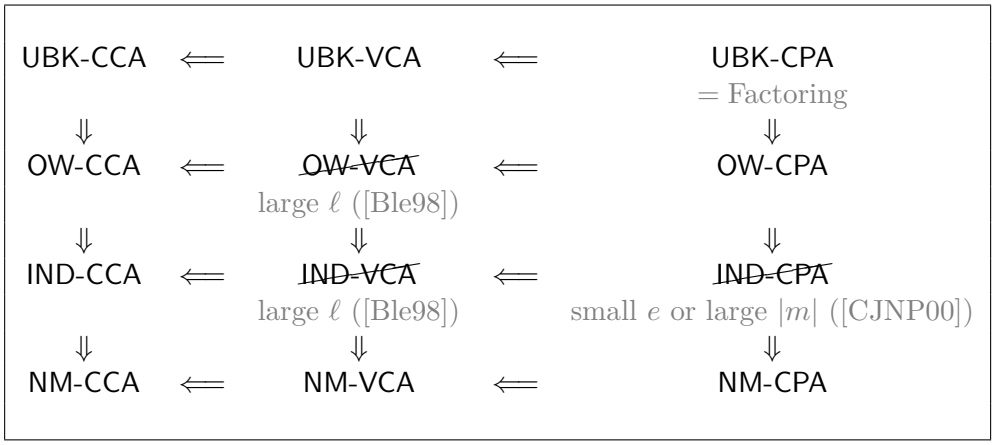


Figure 12.2: PKCS#1 v1.5 security.

plaintext ending by sufficiently many zeros). The second attack applies to arbitrary e, provided that $|m|$ is large and most message bits are zeros. Thus PKCS#1 v1.5 is not GOAL-CPA for a small e or a large $|m|$, for GOAL $\in \{\mathsf{IND}, \mathsf{NM}\}$.

The previous crytanalytic results are summarized in Figure 12.2. In the rest of the chapter, we will study the remaining security notions and prove that PKCS#1 v1.5 is:

- OW-CPA assuming the intractability of the RSA problem (§12.4);
- not OW-CCA for $\ell = 2$ (§12.4);
- not NM-CPA (§12.5.1);
- not IND-VCA for $\ell = 1$ (§12.5.2);
- not OW-CPA in a multi-user setting (§12.6).

12.4 On the OW-CPA-Security of PKCS#1 v1.5

In this paragraph, we prove the following result:

Proposition 12.1. *The OW-CPA security of PKCS#1 v1.5 is equivalent to the RSA problem.*

In [CJNP02, Lemma 2], Coron, Joye, Naccache and Paillier proved that for suitable parameters, the existence of an algorithm that on input $y \in \mathbb{Z}_N^*$ outputs the k_1 least significant bits of $y^d \bmod N$ is equivalent to the existence of an RSA inverter. Following their approach, we prove the following lemma.

Lemma 12.1. *Let $\mathscr{A}$ be a OW-CPA-adversary against PKCS#1 v1.5 with success probability ε within time τ, with uniformly distributed messages of (maximum) length*

$k-11$. *There exists an algorithm $\mathscr{A}'$ that solves the RSA problem with success probability ε' within time τ', where:*

$$\begin{cases} \varepsilon' & \geqslant & \eta^2 \cdot \varepsilon^2 - 2^{-k}\varepsilon \\ \tau' & \leqslant & 2 \cdot \tau + \mathtt{poly}(k) \end{cases}$$

where η is a constant independent of k and $\eta \geqslant 5 \cdot 10^{-8}$.

Proof. Let $\mathscr{A}$ be a OW-CPA-adversary against PKCS#1 v1.5 with success probability ε within time τ. We construct an algorithm $\mathscr{A}'$ that on input $y \in \mathbb{Z}_N^*$ outputs y^d with success probability ε' within time τ':

1. $\mathscr{A}'$ picks $\alpha \in \mathbb{Z}_N^*$ uniformly at random;

2. $\mathscr{A}'$ sets $y_0 = y$ and $y_1 = y \cdot \alpha^e$

3. Let us denote, for $i \in \{0, 1\}$:
$$y_i^d = \omega_i \cdot 2^\beta + m_i$$
with $\beta = 8(k-11)$. y_i is a valid PKCS#1 v1.5 ciphertext if $\omega_i = \mathtt{0002}||r_i||\mathtt{00}$ where r_i is a 8-(nonzero)-octet string. This happens with probability $\eta \geqslant (255/256)^8 \cdot 2^{-24}$. If this happens, then with probability ε, $\mathscr{A}$ will return m_i on input y_i (for $i \in \{0, 1\}$).

4. Therefore with probability at least $(\eta\varepsilon)^2$, we obtain:
$$\alpha(\omega_0 \cdot 2^\beta + m_0) = \omega_1 2^\beta + m_1 \bmod N.$$
Letting $c_1 = 2^{-\beta}(\alpha m_0 - m_1) \mod N$, we get the equation in (ω_0, ω_1):
$$\omega_1 - \alpha \cdot \omega_0 = c_1 \bmod N. \tag{12.2}$$

From [CJNP02, Lemma 3], there exists an algorithm that given this system will output a solution (ω_0, ω_1) (should such a solution exist) with probability at least $1 - 2^{-k}$ on the choice of α. □

Using the same technique, this can be extended to messages of different length, with a possibly higher constant loss in the reduction.

As a byproduct of the previous proof one immediately gets that:

Proposition 12.2. *PKCS#1 v1.5 is not 2-OW-CCA.*

Proof. Thanks to the homomorphic properties of RSA, an adversary can mask the challenge ciphertext c as $c' = cr^e$ for some random r and with two decryption oracle queries, compute the e-th root x' of c' as in the previous proof. Then $x = x'/r$ is the e-th root of c from which, the adversary retrieves readily the plaintext. □

12.5 PKCS#1 v1.5 Malleability and Indistinguishability

We now show in this section that PKCS#1 v1.5 is neither NM-CPA-secure nor IND-VCA-secure. The general idea is the following. Let m be some message to be encrypted, and $\mu(m)$ a corresponding PKCS#1 v1.5 padded encryption block. If m has $Z > 2$ trailing zero bits, then $\mu(m)$ is divisible by 2^Z, and $\mu(m) - \mu(m)/2^Z \bmod N$ has a good probability of still being a valid encryption block. This is not usually the case when the Z least significant bits of m are not all zero.

Let $c = \mu(m)^e \bmod N$ be a ciphertext of some message m. If m has $Z > 2$ trailing zeros, $c' = c \cdot (1 - 2^{-Z})^e \bmod N$ will often be a valid ciphertext of some other message m' which can be related to m: this contradicts non-malleability under chosen plaintext attack. Moreover, if one is granted a single query to a validity oracle, it is possible to distinguish ciphertexts of plaintexts with trailing zeros and ciphertexts from plaintexts whose least significant bits are not all zero: this contradicts indistinguishability under validity checking attack. Note that if i queries are allowed, the distinguishing success odds can quickly approach 1 by iterating the test with $c'_i = c \cdot (i - 2^{-Z})^e \bmod N$ for $i = 1, 2, \ldots$

We develop this idea in further details below.

12.5.1 On the NM-CPA-security of PKCS#1 v1.5

Let k be the byte-size of N, $Z = 4k$, and M a positive integer such that $M + Z + 1$ is a multiple of 8 and $(M + Z + 1)/8 < k - 11$. We consider messages of the following form:

$$m = \underbrace{\bar{m}}_{M \text{ bits}} \| 1_2 \| \underbrace{0 \cdots 0_2}_{Z \text{ zero bits}} .$$

Let $\mathcal{M}$ denote the uniform distribution over messages of this form. Furthermore, we define a relation $\mathcal{R}$ over messages of length $l = M + Z + 1$ as follows: for any l-bit two messages m_1, m_2 (not necessarily of the previous form), $\mathcal{R}(m_1, m_2)$ holds if and only if the M most significant bits of m_1 and m_2 coincide. In particular, for any given message m_2, there is exactly one $m_1 \in \mathcal{M}$ such that $\mathcal{R}(m_1, m_2)$.

Now, consider $m \leftarrow \mathcal{M}$. We can write $\mu(m) \cdot (1 - 2^{-Z})$ as:

$$\begin{array}{cll} & 0002_{16} \| r \| 00_{16} \| \bar{m} \| 1_2 \| & 0 \cdots 0_2 \\ - & & 0002_{16} \| r \| 00_{16} \| \bar{m} \| 1_2 \\ \hline = & 0002_{16} \| r \| 00_{16} \| \bar{m} \| 0_2 \| & \text{some digits} \cdots \text{some digits} \end{array}$$

Hence, $\mu(m) \cdot (1 - 2^{-Z}) = \mu(m')$ for some message $m' \neq m$ such that $\mathcal{R}(m, m')$.

Consider, the NM-CPA adversary $\mathscr{A}$ which outputs sampling algorithm $\mathcal{M}$ in the setup stage, and transforms a challenge ciphertext c into $c' = c \cdot (1 - 2^{-Z})^e \bmod N$. $\mathscr{A}$'s advantage is:

$$\mathrm{Adv}^{\text{NM-CPA}}_{\mathscr{A}} = \Pr[\mathcal{R}(m, m')] - \Pr[m_0 \xleftarrow{\$} \mathcal{M};\ \mathcal{R}(m_0, m')] = 1 - 2^{-M} \geqslant 1/2$$

which is non-negligible. Therefore, PKCS#1 v1.5 encryption is not NM-CPA-secure.

Note that the advantage of $\mathscr{A}$ is, in fact, very close to 1. For a 1024-bit modulus and a 128-bit random nonce, we have $M = 383$, making it exceedingly unlikely that $\mathscr{A}$ will ever fail.

12.5.2 On the IND-VCA-security of PKCS#1 v1.5

We now show how to contradict ciphertext indistinguishability under validity checking attack using a single oracle query. There are two natural types of validity-checking oracles: one which determines whether a given query is a valid ciphertext associated to a plaintext of any length, and the other which also checks message length. We can always contradict IND-VCA-security in the non-length-checking case (which is the one considered in Bleichenbacher's attack [Ble98]) with a single oracle query. Furthermore, if the byte-length of the randomizer is *constant*, as permitted by the PKCS#1 v1.5 standard, it is also possible to break IND-VCA-security with a single query to a length-checking oracle. Both attacks stem from the following result.

Proposition 12.3. *Let $c = \mu(m)^e \bmod N$ be the ciphertext associated to some byte-string message m, ω the byte-length of the random nonce and $c' = c \cdot (1 - 2^{-4})^e \bmod N$.*

1. *If the least significant nibble*[2] *of m is not $\mathtt{0}_{16}$, then c' is never a valid ciphertext.*
2. *If m is a message consisting of a string of $\mathtt{00}_{16}$ bytes, then c' is a valid ciphertext with probability at least 0.47. c' is a valid ciphertext corresponding to a message of the same length as m with probability at least:*

$$\frac{64}{1445}\left(\frac{239}{255}\right)^{\omega-1}.$$

Proof. Starting with the first assertion, consider a message m such that c' is a valid ciphertext. This implies that $\mu(m) - \mu(m) \cdot 2^{-4} \bmod N$ is a valid encoding string that, in particular, begins with the same $\mathtt{0002}_{16}$ pattern as $\mu(m)$.

If, for an integer x, we denote by $\bar{x}$ the only integer in $(-N/2, N/2)$ such that $x \equiv \bar{x} \bmod N$, it follows that:

$$|\overline{\mu(m) \cdot 2^{-4} \bmod N}| < 2^{8k-16}.$$

Consider the set S of residue classes $x \bmod N$ such that $|\bar{x}| < 2^{8k-16}$. Clearly, $\#S = 2^{8k-15} - 1$. On the other hand, let T^+ be the set consisting of k-byte strings of the form $\mathtt{000}_{16}\|u\|\mathtt{0}_{16}$, and T be the union of T^+ and $-T^+$ (where opposites are taken mod N). We also have $\#T = 2^{8k-15} - 1$ and T maps into S under multiplication by $2^{-4} \bmod N$. Since multiplication by 2^{-4} is a permutation of $\mathbb{Z}_N$, we infer that $(y \cdot 2^{-4} \bmod N) \in S$ if and only if $y \in T$.

In particular, if c' is a valid ciphertext, we get $\mu(m) \in T$. By inspection of its top bits, we see that $\mu(m)$ cannot be in $-T^+$, so it has to be in T^+. Its least-significant nibble must thus be $\mathtt{0}_{16}$ as required.

[2] Recall that a nibble is a sequence of four bits, or equivalently a hexadecimal digit.

Turning now to the second assertion, let m be the zero-message of some fixed byte-length. Write the encryption block $\mu(m)$ as follows:

$$\mu(m) = \mathtt{0002}_{16} \parallel r_{2\omega-1} \parallel r_{2\omega-2} \parallel \cdots \parallel r_1 \parallel r_0 \parallel \mathtt{00}_{16} \parallel \mathtt{00}\cdots\mathtt{00}_{16}$$

where $r_0, \ldots, r_{2\omega-1}$ are the nibbles of the random nonce. Recall that the randomizer bytes $r_{2j} \| r_{2j+1}$ are chosen uniformly and independently at random in the range $\mathtt{01}_{16}$, $\ldots$, $\mathtt{FF}_{16}$.

Assuming that $r_{2\omega-1}$ is at least 4 (which happens with probability $(256-4\times16)/255 = 64/85$, and which we will henceforth assume), we can write $(1-2^{-4})\mu(m)$ as:

$$\begin{array}{rc} & \mathtt{0002}_{16} \parallel r_{2\omega-1} \parallel r_{2\omega-2} \parallel \cdots \parallel r_1 \parallel r_0 \parallel \mathtt{00}_{16} \parallel \mathtt{00}\cdots\mathtt{00}_{16} \\ - & \mathtt{0000}_{16} \parallel \mathtt{2}_{16} \parallel r_{2\omega-1} \parallel \cdots \parallel r_2 \parallel r_1 \parallel r_0 \| \mathtt{0}_{16} \parallel \mathtt{00}\cdots\mathtt{00}_{16} \\ \hline = & \mathtt{0002}_{16} \parallel r'_{2\omega-1} \parallel r'_{2\omega-2} \parallel \cdots \parallel r'_1 \parallel r'_0 \parallel s \parallel \mathtt{00}\cdots\mathtt{00}_{16} \end{array}$$

where $r'_j \equiv r_j - r_{j+1} - \kappa_j \mod 16$ for some carry bit κ_j.

Then, $\mu' = (1-2^{-4})\mu(m)$ is a valid encoding block if and only if the first 8 randomizer bytes, namely $r'_{2\omega+1-2j} \| r'_{2\omega-2j}$, $j = 1, \ldots, 8$, are all nonzero. μ' is a valid encoding block for a message of the same length as m (or for short, "strongly valid") if and only if $s = 0$ and all the padding bytes $r'_{2j+1} \| r'_{2j}$, $j = 0, \ldots, \omega-1$, are nonzero. We will find an explicit lower bound for the probability of these events.

Note first that a sufficient condition for $r'_{2j+1} \| r'_{2j}$ to be nonzero is that $r'_{2j+1} \neq 0$. This nibble is zero if and only if $r_{2j+2} \equiv r_{2j+1} - \kappa_{2j+1}$. Now r_{2j+2} is picked independently of r_{2j+1}, since they belong to different bytes; it is also independent of κ_{2j+1}, which only depends on lower order bytes. Consequently:

$$\begin{aligned} \Pr[r'_{2j+1} = 0] &= \sum_{\rho=0}^{15} \Pr[r_{2j+2} = \rho] \cdot \Pr[r_{2j+1} - \kappa_{2j+1} \equiv \rho \pmod{16}] \\ &\leqslant \max_{\rho} \Pr[r_{2j+2} = \rho] = \frac{16}{255} \end{aligned}$$

and this bound still holds conditionally to any assignment of the lower order nibbles r'_{2i+1}, $i < j$. Therefore:

$$\begin{aligned} \Pr[\mu' \text{ is valid}] &\geqslant \Pr[r_{2\omega-1} \geqslant 4 \wedge r'_{2\omega-3} \neq 0 \wedge r'_{2\omega-5} \neq 0 \wedge \cdots \wedge r'_{2\omega-15} \neq 0] \\ &\geqslant \frac{64}{85} \cdot \left(1 - \frac{16}{255}\right)^7 \geqslant 0.47. \end{aligned}$$

Furthermore:

$$\begin{aligned} \Pr[\mu' \text{ is strongly valid}] &\geqslant \Pr[r_{2\omega-1} \geqslant 4 \wedge r'_{2\omega-3} \neq 0 \wedge \cdots \wedge r'_1 \neq 0 \wedge r_0 = 0] \\ &\geqslant \frac{64}{85} \cdot \left(1 - \frac{16}{255}\right)^{\omega-1} \cdot \frac{15}{255} = \frac{64}{1445} \cdot \left(\frac{239}{255}\right)^{\omega-1}. \end{aligned}$$

The corresponding validity assertions for c' follow immediately. □

Consider the IND-VCA adversary $\mathscr{A}$ defined as follows. In the setup stage, $\mathscr{A}$ outputs two messages m_0, m_1 of equal length, with m_1 not zero-terminated (e.g. $\mathtt{00}\cdots\mathtt{0001}_{\mathtt{16}}$) and m_0 consisting of $\mathtt{00}_{\mathtt{16}}$ bytes only. Then, upon receiving a challenge ciphertext $c = \mu(m_b)^e \bmod N$, $\mathscr{A}$ makes a single oracle query and outputs $b' = 0$ or 1 according to whether $c' = c \cdot (1 - 2^{-4}) \bmod N$ is a valid ciphertext or not. Its advantage is then:

$$\begin{aligned}\mathrm{Adv}_{\mathscr{A}}^{\mathsf{IND\text{-}VCA}} &= \Pr[b' = 0 | b = 0] - \Pr[b' = 1 | b = 0] = \Pr[b' = 0 | b = 0] - 0 \\ &\geqslant \begin{cases} 0.47 & \text{if the oracle doesn't check message length} \\ \frac{64}{1445} \cdot \left(\frac{239}{255}\right)^{\omega - 1} & \text{otherwise} \end{cases}\end{aligned}$$

which is non-negligible. In the length-checking case, it is over 2.8% (resp. 1.6%) for 64-bit (resp. 128-bit) nonces.

In the non-length-checking case, we can obtain an even better advantage using $c' = c \cdot (1 - 2^{-8}) \bmod N$ (shifting by 8 bits instead of 4). The proof works similarly provided that N satisfies $N > 2^{8k-7}$, which is not required by the PKCS#1 v1.5 standard but is usually verified in practice. This yields an advantage of at least:

$$\frac{252}{255} \cdot \left(\frac{254}{255}\right)^7 > 0.96.$$

12.6 Broadcast Attack on PKCS#1 v1.5

We now examine the security of PKCS#1 v1.5 in a multiple users context.

In such a scenario, i.e. when broadcast encryption is performed, the sender wishes to transmit the same message m to ℓ parties $P_1, \ldots, P_\ell$. As each party has its own key $pk_i = (e, N_i)$ (with a common public exponent e), the sender encrypts m using all the pk_i's and sends the resulting ciphertexts $c_1, \ldots, c_\ell$ to the corresponding recipients. It has long been known that textbook RSA encryption should not be used in such a context, since an attacker can easily recover the plaintext using the Chinese Remainder Theorem as long as $\ell \geqslant e$. Therefore, m has to be padded before applying the RSA function, and the padding has to be different for each recipient.

In 1988, Håstad [Hås88] showed that using different deterministic linear paddings $\mu_i(m)$ for all parties is not enough to guarantee security. Indeed, when e is small, e ciphertexts are again sufficient to efficiently recover m provided that the encoding functions μ_i are known to the attacker. To achieve this result, Håstad expressed the attack in terms of finding small roots of a univariate modular polynomial, which he accomplishes using Coppersmith's techniques [Cop97].

Håstad's attack does not apply to PKCS#1 v1.5, since the padding used for a given recipient is random, and is thus unknown to an attacker. The following sections will overcome this difficulty. Our main result is as follows.

Proposition 12.4. *Let $c_1, \ldots, c_\ell$ be ℓ PKCS#1 v1.5 ciphertexts of the same message m, of byte length $|m|$. Each c_i is encrypted for a recipient having $pk_i = (e, N_i)$. All N_i are*

k-byte long. Then there exists a heuristic algorithm which, given $c_1, \ldots, c_\ell$, outputs m, provided that the following condition is satisfied:

$$\ell > \frac{e|m|}{k - e(k - |m| - 3)} > 0.$$

Its complexity is polynomial in $e, |m|$ and k but exponential in the number ℓ of recipients.

We describe this algorithm in the coming sections. The core idea is to reduce the problem to finding small modular roots of a multivariate polynomial equation, which can be achieved using a standard generalization of Coppersmith's techniques. As usual, this generalization relies on an assumption of algebraic independence between polynomials, which makes the algorithm heuristic.

12.6.1 The multivariate polynomial of broadcast PKCS#1 v1.5

Recall from §12.3.1 that to encrypt message m for recipient P_i, a PKCS#1 v1.5 sender first generates an $|r_i|$-byte randomizer r_i, and then computes the encoding function:

$$\mu(m, r_i) = \mathtt{0002}_{\mathtt{16}} \parallel r_i \parallel \mathtt{00}_{\mathtt{16}} \parallel m.$$

Numerically, this gives:

$$\mu(m, r_i) = m + 2^{8|m|+8} r_i + 2^{8|m|+8|r_i|+9}.$$

The ciphertext c_i is then computed as $c_i = \mu(m, r_i)^e \bmod N_i$.

Consider then an adversary $\mathscr{A}$ who obtains $c_1, \ldots, c_\ell$. Since the N_i are of the same size, the random nonces r_i have a common byte length $|r|$. Therefore, the ciphertexts collected by $\mathscr{A}$ can be written as:

$$c_1 = (m + Ar_1 + B)^e \mod N_1, \quad \ldots, \quad c_\ell = (m + Ar_\ell + B)^e \mod N_\ell$$

where $A = 2^{8|m|+8}$ and $B = 2^{8|m|+8|r|+9}$. Obviously, the N_i are pairwise co-prime (otherwise, the factorization of some of the N_i could easily be recovered). Thus, the Chinese Remainder Theorem ensures that the previous equations can be rewritten as a single congruence mod $N = N_1 \cdots N_\ell$:

$$u_1 c_1 + \cdots + u_\ell c_\ell = u_1(m + Ar_1 + B)^e + \cdots + u_\ell(m + Ar_\ell + B)^e \mod N$$

where the constants $u_1, \ldots, u_\ell$ are given by the extended Euclidean algorithm. It follows that the tuple $(m, r_1, \ldots, r_\ell)$ is a root of the multivariate modular polynomial:

$$f(x, y_1, \ldots, y_\ell) = u_1(x + Ay_1 + B)^e + \cdots + u_\ell(x + Ay_\ell + B)^e - C \mod N \qquad (12.3)$$

where $C = u_1 c_1 + \cdots + u_\ell c_\ell$. This root is *small* in the sense that all of its components are bounded by quantities that are small compared to N: m is smaller than $2^{8|m|}$ and each r_i is bounded by $2^{8|r|}$. Under suitable conditions on $|m|$ and $|r|$ which will be detailed

below, it will thus become feasible to recover this root, and hence m, in polynomial time using Coppersmith's techniques.

In particular, we show in §12.6.3 that the lattice construction of Jochemsz and May [JM06] yields a heuristic polynomial time algorithm for recovering the small root of f under the following condition:

$$\ell|r| + |m| < \frac{t(\nu)}{s(\nu)} \cdot \ell k$$

where ν is a parameter which determines the complexity of the attack, and $s(\nu)$, $t(\nu)$ are defined as:

$$s(\nu) = \sum_{i=0}^{e\nu} i \binom{e\nu - i + \ell}{\ell} = \binom{\ell + e\nu - 1}{\ell} \frac{(\ell + e\nu)(1 + \ell + e\nu)}{2 + 3\ell + \ell^2}$$

and

$$t(\nu) = \sum_{i=1}^{\nu} \binom{e\nu - ie + \ell + 1}{\ell + 1}.$$

We also prove in the same section that $t(\nu)/s(\nu) \to 1/e$ as ν tends to $+\infty$, so that the best achievable bound on $|m|$ and $|r|$ for which the attack applies is:

$$\ell|r| + |m| < \frac{\ell k}{e}.$$

Since $|r| = k - |m| - 3$ in PKCS#1 v1.5 we obtain the bound announced in Proposition 12.4. As usual when using Coppersmith's techniques, the complexity of the attack is polynomial in the dimension of the constructed lattice and in the size of the entries.

Given the heuristic nature of multivariate Coppersmith-like techniques, we also implement the attack and report practical experiment results in §12.6.4.

12.6.2 Finding small modular roots of a multivariate polynomial

The problem of solving modular polynomial equations is believed to be difficult in the general case. Nevertheless, when we restrict the problem to finding small roots only, the problem becomes easier to solve. Indeed, in 1996, Coppersmith [Cop96] introduced a technique, based on lattice reduction, allowing to recover the root of a univariate modular polynomial provided that this root is small enough. This construction was reformulated in simpler terms by Howgrave-Graham [HG97] and its extensions to more variables found numerous cryptanalytic applications.

Coppersmith's technique. Starting from a polynomial f modulo a known composite integer N, the idea behind Coppersmith's method is to construct a set of polynomials $h_1, \ldots, h_n$ sharing the same sought root over the integers. If the number of these generated polynomials is sufficiently large (greater than the number of variables) and under the assumption that all resultant computations lead to non-zero results, then the root can easily be recovered. Note that this assumption makes the method heuristic.

A sufficient condition ensuring that the polynomials $h_1, \ldots, h_n$ share a common root in $\mathbb{Z}$ was formulated by Howgrave-Graham.

Lemma 12.2 (Howgrave-Graham [HG97]). *Let $h \in \mathbb{Z}[x_1, \ldots, x_n]$ be an integer polynomial that consists of at most ω monomials. Suppose that*

1. *$h(x_{01}, \ldots, x_{0n}) \equiv 0 \bmod N$ for some $|x_{01}| < X_1, \ldots, |x_{0n}| < X_n$.*
2. *$\|h(x_1 X_1, \ldots, x_n X_n)\| < \frac{N}{\sqrt{\omega}}$.*

Then $h(x_{01}, \ldots, x_{0n}) = 0$ holds over the integers.

The problem can thus be reduced to finding polynomials $h_1, \ldots, h_n$ of small norm having the same modular root as f. This can be achieved by representing polynomials as coefficient vectors (using a suitable ordering on monomials) and using lattice reduction techniques such as LLL [LLL82] to search for small vectors in a lattice spanned by polynomials which are known to have the sought modular root.

If L is a lattice of polynomials consisting of at most ω monomials and all having the same modular root as f, then the condition

$$2^{\frac{\omega(\omega-1)}{4(\omega+1-n)}} \det(L)^{\frac{1}{(\omega+1-n)}} < \frac{N}{\sqrt{\omega}} \tag{12.4}$$

ensures first n polynomials obtained by applying LLL to the lattice L match Howgrave-Graham's bound. In the analysis, we let terms that do not depend on N contribute to an error term ε, and simply use the determinant condition $\det(L) \leqslant N^{w+1-n}$.

Lattice construction. A variety of methods for constructing the lattice L have been proposed in the literature. In what follows, we choose to rely on the technique introduced by Jochemsz and May in [JM06].

Recall that we have a polynomial f with an unknown root $\boldsymbol{x_0} = (x_{01}, \ldots, x_{0n})$ modulo some composite integer N whose factorization is unknown. This root is *small* in the sense that each of its components is bounded: $|x_{0i}| < X_i$ for $i \in \{1, \ldots, n\}$. We denote by λ the leading monomial of the polynomial f and by $\mathcal{M}(f)$ the set of monomials appearing in f. Of course, λ can be assumed to be monic as otherwise one simply has to multiply f by the modular inverse of its initial coefficient.

Given $\varepsilon > 0$, we fix an integer $\nu = \nu(\varepsilon)$ and without loss of generality we assume that $\mathcal{M}(f^j) \subseteq \mathcal{M}(f^\nu)$ for $j \in \{1, \ldots, \nu - 1\}$. If k is an integer between 0 and $\nu + 1$, we define the set M_k as $\mathcal{M}(f^\nu) \cap \lambda^k \mathcal{M}(f^{\nu-k})$ (in particular $M_0 = \mathcal{M}(f^\nu)$ and $M_{\nu+1} = \varnothing$). Next, we define the following *shift polynomials*:

$$g_{i_1 \ldots i_n}(x_1, \ldots, x_n) = \frac{x_1^{i_1} \cdots x_n^{i_n}}{\lambda^k} f^k N^{\nu-k}$$

for $k \in \{0, \ldots, \nu\}$ and $x_1^{i_1} \cdots x_n^{i_n} \in M_k \setminus M_{k+1}$. By definition, all polynomials g have the root $(x_{01}, \ldots, x_{0n})$ modulo N^ν. We can now define L as the lattice generated by the

coefficient vectors of all polynomials $g_{i_1 \dots i_n}(x_1 X_1, \dots, x_n X_n)$. If the monomial ordering has been chosen correctly, the matrix corresponding to that lattice is lower triangular and the determinant becomes easy to compute. Indeed, the diagonal elements are those corresponding to the monomial λ^k in f^k for each row. Therefore, the diagonal terms of the matrix are $X_1^{i_1} \cdots X_n^{i_n} N^{\nu-k}$ for $k \in \{0, \dots, \nu\}$ and $x_1^{i_1} \dots x_n^{i_n} \in M_k \setminus M_{k+1}$. By doing a simple computation and neglecting low order terms, one can finally reduce the condition (12.4) to the following new one:

$$\prod_{j=1}^{n} X_j^{s_j} < N^{s_N} \quad \text{for} \quad \begin{cases} s_j = \sum_{x_1^{i_1} \cdots x_n^{i_n} \in M_0} i_j & (1 \leqslant j \leqslant n) \\ s_N = \sum_{k=1}^{\nu} |M_k|. \end{cases} \tag{12.5}$$

This formula expresses an asymptotic condition on the bounds $X_1, \dots, X_n$ allowing to recover the root in polynomial time.

Remark 12.1. The method outlined above is what Jochemsz and May called the "basic strategy"; they also proposed an "extended strategy" in which we can use extra shifts of a certain variable and replace M_k for instance by $M_k = \bigcup_{j=1}^{t} x_1^j \left(\mathcal{M}(f^\nu) \cap \lambda^k \mathcal{M}(f^{\nu-k}) \right)$ for some well-chosen parameter t.

12.6.3 The Jochemsz-May lattice in broadcast PKCS#1 v1.5

Let us examine what the lattice L looks like in the particular setting of broadcast PKCS#1 v1.5 encryption.

Recall from §12.6.1 that recovering m from $c_1, \dots, c_\ell$ reduces to finding the root $(x_0, y_{0,1}, \dots, y_{0,\ell}) = (m, r_1, \dots, r_\ell)$ of the following modular polynomial:

$$f(x, y_1, \dots, y_\ell) = u_1(x + Ay_1 + B)^e + \cdots + u_\ell(x + Ay_\ell + B)^e - C \mod N.$$

We know that this root satisfies the bounds $|x_0| < X$ and $|y_{0i}| < Y$ for all $i \in \{1, \dots, \ell\}$ with $X = 2^{8|m|}$ and $Y = 2^{8|r|}$. We examine how the Jochemsz-May bounds described in the previous section translate into bounds on $|m|$ and $|r|$ allowing the message to be recovered in polynomial time.

Form of the sets M_k. The analysis' first step consists in describing the sets M_k. Note first that the set of monomials $\mathcal{M}(f)$ is included in $\{x^a y_1^{b_1} \cdots y_\ell^{b_\ell} \mid a + b_1 + \cdots + b_\ell \leqslant e\}$. In other words, the geometrical shape of the polynomial f is included in a "pyramid" of dimension $\ell + 1$ of monomials with total degree less than e. We choose the *deglex* monomial order, according to which the leading monomial of f is x^e. The sets M_k can then be described as follows:

$$\begin{aligned} M_0 &= \{x^a y_1^{b_1} \cdots y_\ell^{b_\ell} \mid a + b_1 + \cdots + b_\ell \leqslant e\nu\} \\ M_1 &= \{x^a y_1^{b_1} \cdots y_\ell^{b_\ell} \mid a + b_1 + \cdots + b_\ell \leqslant e\nu \text{ with } a \geqslant e\} \\ &\vdots \\ M_\nu &= \{x^a y_1^{b_1} \cdots y_\ell^{b_\ell} \mid a + b_1 + \cdots + b_\ell \leqslant e\nu \text{ with } a \geqslant e\nu\} \end{aligned}$$

which makes it easy to count the number of monomials in each of them.

Condition on the bounds. Given the above description, we can evaluate the quantities s_j and s_N of equation (12.5) as follows. First, by symmetry, $s_i, s_{j_1}, \ldots s_{j_\ell}$ are all equal to:

$$s(\nu) = \sum_{i=0}^{e\nu} \frac{i(e\nu - i + 1)(e\nu - i + 2)\cdots(e\nu - i + \ell)}{\ell!}.$$

Furthermore, we have:

$$s_N = t(\nu) = \sum_{i=1}^{\nu} \frac{(e\nu - ie + 1)(e\nu - ie + 2)\cdots(e\nu - ie + \ell + 1)}{(\ell+1)!}.$$

Condition (12.5) can then be rewritten as $X^{s(\nu)}Y^{\ell s(\nu)} < N^{t(\nu)}$, and since N is of byte size ℓk, this gives:

$$\ell|r| + |m| < \frac{t(\nu)}{s(\nu)} \cdot \ell k.$$

Asymptotic bound. The functions $s(\nu)$ and $t(\nu)$ are polynomials in ν. Hence, it suffices to evaluate their leading coefficients to obtain an asymptotic estimate as $\nu \to +\infty$. Note further that $s(\nu)$ and $t(\nu)$ are easily expressed in terms of the antidifference operator, which takes a polynomial $P(X)$ to the polynomial $\sigma(P)(X)$ defined by $\sigma(P)(j) = \sum_{i=1}^{j} P(i)$ for $j \in \mathbb{N}$. Indeed:

$$\begin{aligned} s(\nu) &= \sum_{i=0}^{e\nu} (e\nu - i)P(i) = e\nu \cdot \sigma(P)(e\nu) - \sigma(XP)(e\nu) + e\nu \\ &\qquad \text{with} \quad P(X) = \frac{(X+1)\cdots(X+\ell)}{\ell!}; \\ t(\nu) &= \sum_{i=1}^{\nu} Q(i) = \sigma(Q)(\nu) \\ &\qquad \text{with} \quad Q(X) = \frac{(eX - e + 1)\ldots(eX - e + \ell + 1)}{(\ell+1)!}. \end{aligned}$$

Now it is easily seen that if the leading coefficient of P is $c_d X^d$, the leading coefficient of $\sigma(P)$ is $c_d X^{d+1}/(d+1)$. It follows that, as $\nu \to +\infty$, we have:

$$s(\nu) \sim e\nu \cdot \frac{(e\nu)^{\ell+1}}{\ell!(\ell+1)} - \frac{(e\nu)^{\ell+2}}{\ell!(\ell+2)} = \frac{(e\nu)^{\ell+2}}{(\ell+2)!} \quad \text{and} \quad t(\nu) \sim \frac{e^{\ell+1}\nu^{\ell+2}}{(\ell+2)!}.$$

In particular, $t(\nu)/s(\nu) \to 1/e$ when $\nu \to +\infty$. Thus, the best asymptotic bound on $|m|$ and $|r|$ for which the attack is theoretically possible is:

$$\ell|r| + |m| < \frac{\ell k}{e}.$$

12.6.4 Experimental results on the broadcast attack

Given the heuristic nature of Coppersmith's techniques in the multivariate case, it is important to assess the practicality of our attack. In particular, one can ask how many ciphertexts an attacker really needs to recover m. In the particular instance $\{\log_2 N = 1024, e = 3\}$, the number of required ciphertexts is, in principle, really low.

Corollary 12.1. *If a PKCS#1 v1.5 user encrypts the same message m with 64-bit random nonces to multiple recipients using 1024-bit moduli and $e = 3$, then there exists a heuristic polynomial time algorithm that recovers m from $\ell = 4$ ciphertexts.*

Proof. This is a direct consequence of Proposition 12.4, given that, for 1024-bit moduli and 64-bit nonces, message size is equal to 936 bits. □

These parameters, corresponding to optimal message size and encryption speed for 1024-bit moduli, are rather realistic and commonly implemented (although it is much more common to encrypt short messages such as AES symmetric keys than messages of maximal size). However, we will see that the dimension of the lattice that should be reduced to recover m with these parameters can hardly be called practical.

Partial information. Consider an attacker who does not collect all the ℓ required ciphertexts. In that specific case, even if m can not be fully recovered, the attacker can nevertheless obtain partial information on m. In particular, in a scenario where m would not be of full size and would have been previously padded with zero bits, the attack can still be performed.

Practical implementations. To check the applicability of the attack, we investigated three configurations: An attacker having access to two, three and four ciphertexts. Before implementing the attack in each scenario, we first evaluated the dimension of the corresponding lattices (for reasonably small values of the parameter ν) and then expressed the number of bits on m that should be recovered in practice. The results obtained for 1024-bit moduli and $e = 3$ are shown in Table 12.1.

As we can see, the number of bits of m that we are able to recover increases with ν, and approaches 936 bits for $(\ell, \nu) = (4, 10)$. Unfortunately, the dimensions of the constructed lattices are often quite impractical. In many cases, matrix size turns out to exceed 1000, making lattice reduction unfeasible in practice. As a result, and because we had limited processor time at our disposal, we only ran practical experiments in the small cases, namely $\ell = 2$ and $\nu = 2, 3$.

Experiments have been performed on a 7-processor Intel Xeon clocked at 1.86 GHz. Each test was done in the same way: construction of the appropriate lattice, LLL-reduction and then extraction of short vectors. Although theoretically one only needs a number of short vectors equal to the number of variables, in practice we decided to take as many vectors as possible to increase the success odds of the attack. Most of the CPU time was claimed by the LLL-reduction step (approximately 3 hours for $(\ell, \nu) = (2, 3)$). With

	$\ell = 2$		$\ell = 3$		$\ell = 4$	
ν	$\dim(L)$	$\lvert m\rvert$	$\dim(L)$	$\lvert m\rvert$	$\dim(L)$	$\lvert m\rvert$
2	84	213	210	246	462	249
3	220	306	715	395	2002	451
4	455	359	1820	483	6188	578
5	816	394	3876	542	15504	664
6	1330	418	7315	584	33649	726
7	2024	435	12650	615	65780	773
10	5456	469	46376	675	324632	863

Table 12.1: Dimension of the lattice L and number of recoverable bits for different values of (ℓ, ν).

1024-bit moduli and 64-bit random nonces, we managed to recover a 115-bit message (padded with zeros).

Toy example. To better illustrate the process, we present here a toy example for 150-bit moduli and a 5-bit message m. After constructing the polynomial f had and generating the lattice (of dimension 84 in this case), we performed the LLL-reduction. Here the first 50 vectors corresponded to polynomials having the desired root over the integers. We then took all these polynomials and computed a Gröbner basis of the ideal they generated. The results were the following:

$$N_i = 150 \text{ bits},\ m = 5 \text{ bits},\ r = 6 \text{ bits},\ e = 3,\ \nu = 2$$
$$f(x, y, z) \text{ with modular root } (24, 58, 34).$$

$$\begin{cases} p_1 &= z^4 + 512z^3 + 98304z^2 + 86093442y - 94710946z - 1908346880 \\ p_2 &= yz^2 - \frac{89}{81}z^3 + 256yz - \frac{7936}{27}z^2 + 16384y - \frac{573440}{27}z - \frac{33783328}{81} \\ p_3 &= y^2 - \frac{62}{27}yz + \frac{961}{729}z^2 - \frac{1024}{27}y + \frac{31744}{729}z + \frac{262144}{729} \\ p_4 &= x - 55297y + 63489z + 1048576. \end{cases}$$

The Gröbner basis computation does not allow us to directly recover the message m, since the corresponding subvariety is not of dimension zero, as was usually the case in most experiments. In fact, we commonly faced problems of algebraic dependence between the resulting polynomials (hence our choice to take a large number of polynomials, rather than the first few, to compute the Gröbner basis). Nevertheless, it was usually possible to recover the message, as the polynomials in the Gröbner basis had a very simple form. In this particular case, for instance, p_4 is affine and p_3 can be written as $(ay + bz + c)^2$, making it easy to recover the common root.

12.7 Conclusion

Figure 12.3 summarizes our current knowledge of the security of PKCS#1 v1.5.

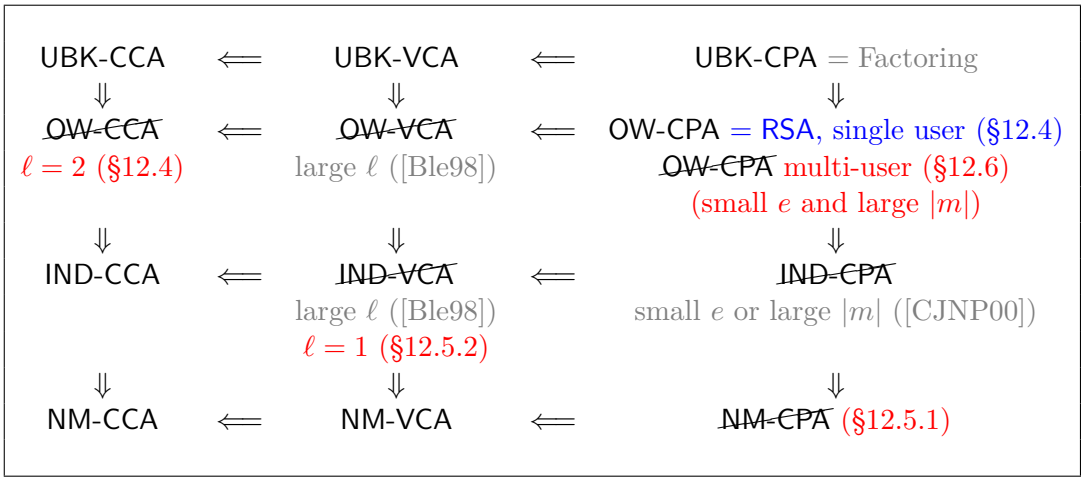


Figure 12.3: Updated security status of PKCS#1 v1.5.

Chapter 13

Cryptanalysis of the RSA Subgroup Assumption

13.1 Introduction

This chapter is devoted to a different type of attack: a "better" exponential algorithm for factoring an RSA modulus of a special form.

At TCC 2005, Groth [Gro05] underlined the usefulness of working in small RSA subgroups of hidden order. In assessing the security of the relevant hard problems, however, the best attack considered for a subgroup of size $2^{2\ell}$ had a complexity of $O(2^{\ell})$. Accordingly, $\ell = 100$ bits was suggested as a concrete parameter.

In this chapter, we exhibit an attack with a complexity of roughly $2^{\ell/2}$ operations, suggesting that Groth's original choice of parameters was overly aggressive. We also discuss the practicality of this new attack and various implementation issues. This work was presented at PKC 2011 [CJM+11].

13.1.1 Groth's small RSA subgroups

In 2005, Groth [Gro05] proposed a collection of cryptographic primitives based on small RSA subgroups of $\mathbb{Z}_N^*$ of hidden orders. The motivation behind these constructions is improved efficiency and tighter security reductions.

The RSA moduli N used by Groth are of the form:

$$N = p \cdot q = (2p'r + 1) \cdot (2q's + 1)$$

where p, p', q, q' are prime integers and r, s are random integers. Then there exists a unique subgroup $\mathbb{G}$ of $\mathbb{Z}_N^*$ of order $p'q'$. Letting g be a random generator of $\mathbb{G}$, the pair (N, g) is made public whereas everything else including the group order $p'q'$ is kept secret.

The best attack considered in [Gro05] has complexity $O(p')$. Therefore, when proposing concrete parameters, the author suggests to take the bit-lengths $\ell_{p'}$ and $\ell_{q'}$ of the primes p' and q' as $\ell_{p'} = \ell_{q'} = 100$.

13.1.2 Our results

We do not consider any specific scheme from [Gro05] in this chapter. Instead, we describe an attack that recovers the secret factors of N from the public data (N, g) in $\widetilde{O}(\sqrt{p'})$, and hence breaks all the proposed schemes with the same complexity. This results in a 2^{50} attack making the choice $\ell_{p'} = \ell_{q'} = 100$ potentially insecure. We analyze the practicality of our attack with an implementation, for which we provide the source code in the Appendix.

Remark 13.1. In [Gro05], Groth also considers RSA subgroups where r and s are smooth integers (i.e. all prime factors of r and s are smaller than some bound B). For this specific variant an attack in complexity $O(\sqrt{p'})$ is given in [Gro05], and consequently larger parameters ($\ell_{p'} = \ell_{q'} = 160$) are suggested. In this chapter we do not consider this variant but directly focus on the general case.

Remark 13.2. Other works have proposed schemes based on small subgroups of $\mathbb{Z}_N^*$. The attack introduced in this chapter applies to some, but not all of them. In particular, the scheme proposed by Damgård et al. in [DGK07] uses a subgroup of *prime* order v of $\mathbb{Z}_N^*$, where v is a factor of both $p-1$ and $q-1$ (of around 160 bits). Since the group has the same order modulo p and q, the attack presented herein does not apply to this scheme. On the other hand, it does, in principle, apply to the subgroup variant of the Paillier cryptosystem [PP99]. The parameter choice from the original paper was more conservative than that of Groth, however (320-bit subgroup), making it out of reach of our new attack.

13.2 The New Attack

Using the notations above, we factor N in time $\widetilde{O}(\sqrt{p'})$ and memory $O(\sqrt{p'})$ as follows. Recall that the RSA modulus $N = pq$ is such that:

$$N = p \cdot q = (2p'r + 1) \cdot (2q's + 1)$$

where p' and q' are prime; besides, g is a generator of the subgroup $\mathbb{G}$ of order $p'q'$. From $g^{p'q'} = 1 \mod N$ we get:

$$g^{p'} = 1 \mod p. \tag{13.1}$$

Let ℓ denote the bit length of p', which we assume is even without loss of generality, and write $\Delta = 2^{\ell/2}$. We then have

$$p' = a + \Delta \cdot b$$

with $0 \leqslant a, b < 2^{\ell/2}$. From (13.1), we get:

$$g^a = (g^\Delta)^{-b} \mod p.$$

If the prime factor p was known, one could carry out a baby-step giant-step attack by generating the following two lists:

$$L_p = \{g^i \bmod p : 0 < i < 2^{\ell/2}\}$$
$$L'_p = \{(g^\Delta)^{-j} \bmod p : 0 \leqslant j < 2^{\ell/2}\}$$

and finding a collision between L_p and L'_p, which would reveal a, b and thus p' in total time and space $O(2^{\ell/2})$.

Obviously p is unknown, so instead of computing L_p and L'_p, we generate the two following lists modulo N:

$$L = \{x_i = g^i \bmod N : 0 < i < 2^{\ell/2}\}$$
$$L' = \{y_j = (g^{\Delta})^{-j} \bmod N : 0 \leqslant j < 2^{\ell/2}\}$$

One could then compute $\gcd(x_i - y_j, N)$ for all $x_i \in L$ and all $y_j \in L'$. Since we have

$$x_a - y_b = 0 \mod p$$

this would reveal the factors of N for $i = a$ and $j = b$. However, the complexity of this naive approach is quadratic in Δ, and will thus require 2^{ℓ} computations, not $2^{\ell/2}$. Hence we proceed as follows instead:

1. Generate the polynomial:

$$f(x) = \prod_{x_i \in L} (x - x_i) \mod N$$

2. For all $y_j \in L'$, evaluate f at y_j and compute $\gcd(f(y_j), N)$.

Since we have

$$f(y_b) = \prod_{x_i \in L} (y_b - x_i) = (y_b - x_a) \cdot R = 0 \mod p$$

computing $\gcd(f(y_j), N)$ reveals the factors of N for $j = b$.

The attack is summarized in Algorithm 13.1. In the next section we show that it can be carried out in time quasi-linear in the cardinalities of L and L'.

13.3 Attack Complexity

This attack involves the computation and evaluation of a polynomial of the form:

$$f(x) = \prod_{i=1}^{d-1} (x - x_i) \mod N$$

with $d = 2^{\ell/2}$. It is a classical fact [Ber08] that the coefficients of such a polynomial can be computed using a product tree, with a total number of operations in $\mathbb{Z}_N$ which is quasilinear in d (namely $O(M(d) \log d)$, where $M(d)$ is the complexity of the multiplication of two polynomials of degree d). Similarly, with a remainder tree, this polynomial can be evaluated at all points y_j, $0 \leqslant j < d$ in $O(M(d) \log d)$ operations.

In our case, however, both (x_i) and (y_j) are geometric progressions, hence even more efficient algorithms exist: the Newton basis conversion algorithms of Bostan and Schost [BS05] make it possible to compute f using $O(d)$ pre-computations and a single

Algorithm 13.1 Attack overview.

1: Let $\Delta \leftarrow 2^{\ell/2}$.
2: **for** $i = 0$ to $\Delta - 1$ **do**
3: $\quad x_i \leftarrow g^i \mod N$
4: $\quad y_i \leftarrow (g^\Delta)^{-i} \mod N$
5: **end for**
6: Generate the polynomial

$$f(x) \leftarrow \prod_{i=1}^{\Delta-1} (x - x_i) \mod N$$

7: **for** $i = 0$ to $\Delta - 1$ **do**
8: $\quad$ Evaluate $f(y_i) \in \mathbb{Z}_N$
9: $\quad$ Attempt to factor N by computing $\gcd(f(y_i), N)$.
10: **end for**

middle product [HQZ04] of polynomials of degree d, and to evaluate $f(y_j)$ for all j using $O(d)$ pre-computations, a product of polynomials of degree d and a middle product of polynomials of degree d. See the next section for details. This results in an overall complexity of $3M(d) + O(d)$ for the complete attack, with a small constant in the big-O. Space requirements are also $O(d)$, to store a few polynomials of degree d.

Thus, for typical parameter sizes, the attack is *essentially linear in $\sqrt{p'}$ both in time and space*.

13.4 Algorithmic Details

As discussed above, we can break down the attack in two steps: first compute the coefficients of the polynomial $f(x) = \prod_{i=1}^{d-1}(x - x_i) \mod N$, and then evaluate $f \mod N$ at each of the points y_j. Since both (x_i) and (y_j) are geometric progressions, both of these steps reduce to a variant of the discrete Fourier transform, called the "chirp transform" (or its inverse) [RSR69, Blu70]. In our implementation, we carry out these computations using the particularly efficient algorithms of Bostan and Schost [BS05], as described in [Bos03, §5.5]. More precisely, Bostan gives pseudocode, reproduced as Algorithms 13.3 and 13.4 in Appendix 13.A, to compute polynomial interpolation and polynomial evaluation at a geometric progression.

In our case, a number of further simplifications are possible in the interpolation stage. Indeed, $f(x_i) = 0$ for $1 \leqslant i \leqslant d - 1$ and $f(1) = \prod_{i=1}^{d-1}(1 - x_i)$, so with the notations of Algorithm 13.3, $v_0 = (-1)^{n-1}s_{n-1}$ and $v_i = 0$ for $i > 0$. This means in particular that the polynomial multiplication of Algorithm 13.3, Step 9 reduces to a simple scalar multiplication, and that the computations of Steps 10–12 can be carried out in the main loop. We can also have a slightly more conservative memory management, with only 4 polynomials of degree $n - 1$ kept in memory for both the interpolation and the

$\ell = \lceil \log_2 p' \rceil$	**running time**
26 bits	1.9 seconds
28 bits	4.0 seconds
30 bits	8.1 seconds
32 bits	16.5 seconds
34 bits	33.5 seconds
36 bits	68.9 seconds

Table 13.1: Experimental attack running times for 1024-bit moduli.

$\ell = \lceil \log_2 p' \rceil$	**running time**	**estimated number of clock cycles**
60 bits	3 days	2^{50}
80 bits	9 years	2^{60}
100 bits	9000 years	2^{70}

Table 13.2: Estimated attack running times for 1024-bit moduli.

evaluation step. Finally, the multiplications by s_i in Algorithm 13.4, Step 14 can be skipped altogether as we search for a factor of N with GCD computations, since the s_i's are prime to N by construction. We obtain the detailed procedure described in Algorithm 13.2.

The attack can again be broken down in three stages: interpolation in Steps 2–18, evaluation in Steps 19–36 and factor search in Steps 37–40. Complexity is dominated by the three quasi-linear multiplication steps: the middle products of Steps 15 and 32, and the polynomial multiplication of Step 36.

13.5 Implementation

We provide the source code of our attack in Appendix 13.B. The implementation of polynomial interpolation and evaluation using Newton basis conversions largely follows the pseudocode from [Bos03] (Figure 5.1 and 5.2), implemented in C using the FLINT library [HH$^+$10].

In Table 13.1, we provide the observed running time of our attack on an Intel Core2 Duo E8500 3.12 GHz, for 1024-bit RSA moduli. The program was linked to the following libraries: FLINT 1.6 (prerelease), MPIR 2.1.3 and MPFR 3.0, and ran on a single CPU core.

From Table 13.1 we see that, as expected, running times are essentially linear in $\sqrt{p'}$. Direct extrapolation yields the estimates given in Table 13.2.

Thus, even the parameter $\ell = 100$ suggested in Groth's paper [Gro05] would require a large but not unachievable amount of computation, even by academic standards. As a

Algorithm 13.2 Detailed attack.

function ATTACK(g, n, N)
  $p \leftarrow 1$; $q \leftarrow 1$; $s \leftarrow 1$; $u \leftarrow 1$; $z \leftarrow 1$; $w \leftarrow 1$
  Initialize U, Z, S, W as zero polynomials of degree $n-1$
  $U_0 \leftarrow u$; $Z_0 \leftarrow z$; $W_0 \leftarrow w$
  for $i = 1$ to $n-1$ **do**
    $p \leftarrow p \cdot g \bmod N$
    $q \leftarrow q \cdot p \bmod N$
    $s \leftarrow s \cdot (p-1) \bmod N$
    $u \leftarrow u \cdot p/(1-p) \bmod N$
    $z \leftarrow (-1)^i u/q \bmod N$
    $w \leftarrow q/(s \cdot u) \bmod N$
    $U_i \leftarrow u$; $Z_i \leftarrow z$; $W_i \leftarrow w$
  end for
  $Z \leftarrow (-1)^{n-1} s_{n-1} \cdot Z \bmod N$
  $W \leftarrow \mathrm{mul}^t(n-1, U, W)$
  for $i = 0$ to $n-1$ **do**
    $W_i \leftarrow W_i \cdot Z_i \bmod N$
  end for
  $g \leftarrow 1/(p \cdot g) \bmod N$ ▷ $g \leftarrow g^{-\Delta}$
  $p \leftarrow 1$; $q \leftarrow 1$; $s \leftarrow 1$; $u \leftarrow 1$; $z \leftarrow 1$; $w \leftarrow 1$
  $U \leftarrow 0$; $Z \leftarrow 0$
  $U_0 \leftarrow u$; $Z_0 \leftarrow z$; $S_0 \leftarrow s$
  for $i = 1$ to $n-1$ **do**
    $p \leftarrow p \cdot g \bmod N$
    $q \leftarrow q \cdot p \bmod N$
    $s \leftarrow s/(p-1) \bmod N$
    $u \leftarrow u \cdot p/(1-p) \bmod N$
    $z \leftarrow (-1)^i u/q \bmod N$
    $S_i \leftarrow s$; $U_i \leftarrow u$; $Z_i \leftarrow z$
    $W_i \leftarrow W_i/z \bmod N$
  end for
  $W \leftarrow \mathrm{mul}^t(n-1, Z, W)$
  for $i = 0$ to $n-1$ **do**
    $W_i \leftarrow (-1)^i W_i \cdot U_i \bmod N$
  end for
  $W \leftarrow W \cdot S$
  for $i = 0$ to $n-1$ **do**
    if $\gcd(W_i, N) \neq 1$ **then return** $\gcd(W_i, N)$ ▷ Factor found!
    end if
  end for
end function

comparison, the recent factorization of RSA 768 [KAF+10] required about 2000 CPU-years.

However, it is not obvious how the algorithm can be efficiently parallelized to distribute the computation. A naive parallelization strategy is to reduce the number of x_i's and increase the number of y_i's by some factor 2^k, but this only reduces time and memory by a factor of about 2^k while requiring 2^{2k} parallel nodes. It would be desirable to be able to distribute the full size computation—both the FFT steps (multiplication and middle product) and the pre-computations—but this appears to be nontrivial.

Most importantly, it is difficult to deal with larger parameters because the attack is heavily memory-bound: the $O(\sqrt{p'})$ memory requirement is a serious hurdle. In experiments, we encountered memory problems as early as $\ell \approx 38$ for a 1024-bit modulus, and even with much more careful memory management and the use of mass storage rather than RAM, it seems unlikely that parameters larger than $\ell \approx 60$ can be attacked unless storage can be efficiently distributed as well.

13.6 Conclusion

We have described an attack against the RSA subgroup of hidden order described in [Gro05] that works in time $\widetilde{O}(\sqrt{p'})$ while the best attack considered in [Gro05] had complexity $O(p')$. We have implemented our attack and assessed its practicality. As expected, our attack exhibits a time complexity quasi-linear in $\sqrt{p'}$. In terms of CPU time alone, the parameters suggested in [Gro05] appear to be within reach for a resourceful attacker. However, due to heavy memory requirements and parallelization problems, these parameters may remain unchallenged.

An interesting open question is to decrease the memory requirement: an algorithm similar to Pollard rho or Pollard lambda with constant memory would be the most convenient type of attack on this problem if it exists. If not, a method for distributing the computation and storage efficiently would be the simplest way to make the attack practical for larger parameters.

13.A Bostan's Algorithms

Algorithm 13.3 Polynomial interpolation: compute the polynomial F of degree $< n$ such that $F(p_i) = v_i$, where $p_i = q^i$, $0 \leqslant i \leqslant n-1$.

function InterpGeom$(p_0, \ldots, p_{n-1}; v_0, \ldots, v_{n-1})$
  $q_0 \leftarrow 1$; $s_0 \leftarrow 1$; $u_0 \leftarrow 1$; $z_0 \leftarrow 1$; $w_0 \leftarrow v_0$
  for $i = 1$ to $n-1$ **do**
    $q_i \leftarrow q_{i-1} \cdot p_i$
    $s_i \leftarrow s_{i-1} \cdot (p_i - 1)$
    $u_i \leftarrow u_{i-1} \cdot p_i/(1-p_i)$
    $z_i \leftarrow (-1)^i u_i/q_i$
  end for
  $H \leftarrow \left(\sum_{i=0}^{n-1} v_i/s_i x^i\right) \cdot \left(\sum_{i=0}^{n-1} (-x)^i q_i/s_i\right)$
  for $i = 1$ to $n-1$ **do**
    $w_i \leftarrow (-1)^i \, \texttt{Coeff}(H, i)/u_i$
  end for
  $G \leftarrow \mathrm{mul}^t(n-1, \sum_{i=0}^{n-1} u_i x^i, \sum_{i=0}^{n-1} w_i x^i)$
  return $\sum_{i=0}^{n-1} z_i \, \texttt{Coeff}(G, i) x^i$
end function

Algorithm 13.4 Polynomial evaluation: evaluate the polynomial F at all points $p_i = q^i$, $0 \leqslant i \leqslant n-1$.

function EvalGeom$(p_0, \ldots, p_{n-1}; F)$
  $q_0 \leftarrow 1$; $s_0 \leftarrow 1$; $u_0 \leftarrow 1$; $z_0 \leftarrow 1$; $g_0 \leftarrow 1$
  for $i = 1$ to $n-1$ **do**
    $q_i \leftarrow q_{i-1} \cdot p_i$
    $s_i \leftarrow s_{i-1} \cdot (p_i - 1)$
    $u_i \leftarrow u_{i-1} \cdot p_i/(1-p_i)$
    $z_i \leftarrow (-1)^i u_i/q_i$
  end for
  $G \leftarrow \mathrm{mul}^t(n-1, \sum_{i=0}^{n-1} z_i x^i, \sum_{i=0}^{n-1} \texttt{Coeff}(F, i)/z_i x^i)$
  for $i = 1$ to $n-1$ **do**
    $g_i \leftarrow (-1)^i u_i \, \texttt{Coeff}(G, i)$
  end for
  $W \leftarrow \left(\sum_{i=0}^{n-1} g_i x^i\right) \cdot \left(\sum_{i=0}^{n-1} s_i^{-1} x^i\right)$
  return $s_0 \, \texttt{Coeff}(W, 0), \ldots, s_{n-1} \, \texttt{Coeff}(W, n-1)$
end function

13.B Source Code of the Attack

```
#include <stdio.h>
#include <time.h>
#include <gmp.h>
#include "F_mpz_poly.h"
#include "F_mpz.h"

void F_mpz_poly_set_coeff_F_mpz(F_mpz_poly_t, ulong, const F_mpz_t);


void attack(F_mpz_t q, F_mpz_t m, unsigned long n)
{
    F_mpz_poly_t polW, polU, polZ, polS;
    F_mpz_t pi, qi, si, ui, zi, wi, x;
    unsigned long i;

    printf("Attack started.\n");

    F_mpz_poly_init2(polW, n);
    F_mpz_poly_init2(polU, n);
    F_mpz_poly_init2(polZ, n);
    F_mpz_poly_init2(polS, n);

    /* Step 1: interpolation */
    F_mpz_init(pi); F_mpz_set_ui(pi, 1);
    F_mpz_init(qi); F_mpz_set_ui(qi, 1);
    F_mpz_init(si); F_mpz_set_ui(si, 1);
    F_mpz_init(ui); F_mpz_set_ui(ui, 1);
    F_mpz_init(zi); F_mpz_set_ui(zi, 1);
    F_mpz_init(wi); F_mpz_set_ui(wi, 1);
    F_mpz_init(x);

    F_mpz_poly_set_coeff_F_mpz(polU, n-1, ui);
    F_mpz_poly_set_coeff_F_mpz(polZ, 0, zi);
    F_mpz_poly_set_coeff_F_mpz(polW, 0, wi);

    for(i=1; i<n; i++) {
        F_mpz_mulmod2(qi, qi, pi, m);
        F_mpz_mulmod2(pi, pi, q, m);

        /* s_i = s_{i-1} * (p_i - 1) */
        F_mpz_sub_ui(x, pi, 1);
        F_mpz_mulmod2(si, si, x, m);

        /* u_i = u_{i-1} * p_i/(1 - p_i) */
        F_mpz_invert(x, x, m);
        F_mpz_mul2(ui, ui, pi);
        F_mpz_mul2(ui, ui, x);
```

```
        F_mpz_neg(ui, ui);
        F_mpz_mod(ui, ui, m);

        F_mpz_poly_set_coeff_F_mpz(polU, n-1-i, ui);

        /* z_i = (-1)^i u_i / q_i */
        F_mpz_invert(x, qi, m);
        F_mpz_mulmod2(x, x, ui, m);
        if(i & 1)
            F_mpz_neg(zi, x);
        else
            F_mpz_set(zi, x);

        F_mpz_poly_set_coeff_F_mpz(polZ, i, zi);

        /* w_i = q_i / (s_i * u_i) */
        F_mpz_mul2(x, x, si);
        F_mpz_invert(wi, x, m);

        F_mpz_poly_set_coeff_F_mpz(polW, i, wi);
    }

    /* W *= (-1)^{n-1} s_{n-1} */
    if(!(n & 1))
        F_mpz_neg(si, si);
    F_mpz_poly_scalar_mul(polW, polW, si);
    F_mpz_poly_scalar_smod(polW, polW, m);

    F_mpz_poly_mul_trunc_left(polW, polU, polW, n-1);
    F_mpz_poly_right_shift(polW, polW, n-1);

    for(i=0; i<n; i++) {
        F_mpz_mulmod2(polW->coeffs + i, polW->coeffs + i, polZ->coeffs + i, m);
    }

    printf("Polynomial interpolation complete.\n");

    /* Step 2: evaluation */
    F_mpz_mul2(q, q, pi);
    F_mpz_invert(q, q, m);

    F_mpz_set_ui(pi, 1);
    F_mpz_set_ui(qi, 1);
    F_mpz_set_ui(si, 1);
    F_mpz_set_ui(ui, 1);
    F_mpz_set_ui(zi, 1);
    F_mpz_set_ui(wi, 1);

    F_mpz_poly_zero(polU);
```

```
F_mpz_poly_zero(polZ);

F_mpz_poly_set_coeff_F_mpz(polU, 0, ui);
F_mpz_poly_set_coeff_F_mpz(polZ, n-1, zi);
F_mpz_poly_set_coeff_F_mpz(polS, 0, si);

for(i=1; i<n; i++) {
    F_mpz_mulmod2(qi, qi, pi, m);
    F_mpz_mulmod2(pi, pi, q, m);

    /* s_i = s_{i-1} / (p_i - 1) */
    F_mpz_sub_ui(x, pi, 1);
    F_mpz_invert(x, x, m);
    F_mpz_mulmod2(si, si, x, m);

    F_mpz_poly_set_coeff_F_mpz(polS, i, si);

    /* u_i = u_{i-1} * p_i/(1 - p_i) */
    F_mpz_mul2(ui, ui, pi);
    F_mpz_mul2(ui, ui, x);
    F_mpz_neg(ui, ui);
    F_mpz_mod(ui, ui, m);

    F_mpz_poly_set_coeff_F_mpz(polU, i, ui);

    /* z_i = (-1)^i u_i / q_i */
    F_mpz_invert(x, qi, m);
    F_mpz_mulmod2(x, x, ui, m);
    if(i & 1)
        F_mpz_neg(zi, x);
    else
        F_mpz_set(zi, x);

    F_mpz_poly_set_coeff_F_mpz(polZ, n-1-i, zi);

    /* w_i /= z_i */
    F_mpz_invert(x, zi, m);
    F_mpz_mulmod2(polW->coeffs + i, polW->coeffs + i, x, m);
}

F_mpz_poly_mul_trunc_left(polW, polZ, polW, n-1);
F_mpz_poly_right_shift(polW, polW, n-1);

F_mpz_poly_clear(polZ);

for(i=0; i<n; i++) {
    if(i & 1)
        F_mpz_neg(polU->coeffs + i, polU->coeffs + i);
    F_mpz_mulmod2(polW->coeffs + i, polW->coeffs + i, polU->coeffs + i, m);
```

```
    }

    F_mpz_poly_clear(polU);
    F_mpz_poly_mul(polW, polW, polS);
    F_mpz_poly_clear(polS);

    printf("Evaluation complete. Searching for a factor.\n");
    for(i=0; i<n; i++) {
        F_mpz_gcd(x, polW->coeffs + i, m);
        if(!F_mpz_is_one(x)) {
            printf("Factor found!\n");
            F_mpz_print(x);
            printf("\n");
            break;
        }
    }

    F_mpz_poly_clear(polW);

    F_mpz_clear(pi);
    F_mpz_clear(qi);
    F_mpz_clear(si);
    F_mpz_clear(ui);
    F_mpz_clear(zi);
    F_mpz_clear(wi);
    F_mpz_clear(x);
}


int main()
{
    F_mpz_t m, g;
    unsigned long d;
    clock_t c0, c1;

    printf("Enter parameters N, g, d.\n");
    F_mpz_init(m);
    F_mpz_init(g);

    F_mpz_read(m);
    F_mpz_read(g);
    scanf("%lu", &d);

    printf("\nParameters:\nN = ");
    F_mpz_print(m);
    printf("\ng = ");
    F_mpz_print(g);
    printf("\nd = %lu\n\n", d);
```

```
    c0 = clock();
    attack(g, m, 1L<<d);
    c1 = clock();

    printf("Elapsed time: %.3f seconds.\n",
            ((float)(c1-c0))/CLOCKS_PER_SEC);
}
```

Contents

List of Figures

List of Tables

List of Algorithms

Bibliography

[ABF+02] Christian Aumüller, Peter Bier, Wieland Fischer, Peter Hofreiter, and Jean-Pierre Seifert. Fault attacks on RSA with CRT: Concrete results and practical countermeasures. In Burton S. Kaliski Jr., Çetin Kaya Koç, and Christof Paar, editors, *CHES*, volume 2523 of *Lecture Notes in Computer Science*, pages 260–275. Springer, 2002. 11.1.3

[Ajt02] Miklós Ajtai. Random lattices and a conjectured 0–1 law about their polynomial time computable properties. In *FOCS*, pages 733–742. IEEE Computer Society, 2002. 10.2.4

[Ajt06] Miklós Ajtai. Generating random lattices according to the invariant distribution. Draft, March 2006. 10.2.4, 10.2.4

[ALNR11] Christophe Arène, Tanja Lange, Michael Naehrig, and Christophe Ritzenthaler. Faster computation of the Tate pairing. *J. Number Theory*, 131(5):842–857, 2011. 8.1.1

[ANSI X9.44] ANSI X9.44:2007. *Public Key Cryptography for the Financial Services Industry – Key Establishment Using Integer Factorization Cryptography.* ANSI, Washington DC, USA, 2007. 12.1.1

[ANSI X9.62] ANSI X9.62:2005. *Public Key Cryptography for the Financial Services Industry, The Elliptic Curve Digital Signature Algorithm (ECDSA).* ANSI, Washington DC, USA, 2005. I

[ANSI X9.63] ANSI X9.63:2001. *Public Key Cryptography for the Financial Services Industry, Key Agreement and Key Transport Using Elliptic Curve Cryptography.* ANSI, Washington DC, USA, 2001. I

[ANSI X9.98] ANSI X9.98:2010. *Lattice-Based Polynomial Public Key Establishment Algorithm for the Financial Services Industry.* ANSI, Washington DC, USA, 2010. 1.2.2

[B+03] Matthias Brüstle et al. *SOSSE: Simple Operating System for Smartcard Education*, 2003. `http://www.mbsks.franken.de/sosse/`. 11.A.3

[BBJ+08] Daniel J. Bernstein, Peter Birkner, Marc Joye, Tanja Lange, and Christiane Peters. Twisted Edwards curves. In Serge Vaudenay, editor, *AFRICACRYPT*, volume 5023 of *Lecture Notes in Computer Science*, pages 389–405. Springer, 2008. 8.1.1

[BBM00] Mihir Bellare, Alexandra Boldyreva, and Silvio Micali. Public-key encryption in a multi-user setting: Security proofs and improvements. In Bart Preneel, editor, *EUROCRYPT*, volume 1807 of *Lecture Notes in Computer Science*, pages 259–274. Springer, 2000. 12.2.3

[BCCN01] Eric Brier, Christophe Clavier, Jean-Sébastien Coron, and David Naccache. Cryptanalysis of rsa signatures with fixed-pattern padding. In Joe Kilian, editor, *CRYPTO*, volume 2139 of *Lecture Notes in Computer Science*, pages 433–439. Springer, 2001. 2.2.7

[BCDG09] Alexandre Berzati, Cécile Canovas, Jean-Guillaume Dumas, and Louis Goubin. Fault attacks on RSA public keys: Left-to-right implementations are also vulnerable. In Marc Fischlin, editor, *CT-RSA*, volume 5473 of *Lecture Notes in Computer Science*, pages 414–428. Springer, 2009. 11.1.1, 11.1.3

[BCDG10] Alexandre Berzati, Cécile Canovas-Dumas, and Louis Goubin. Public key perturbation of randomized RSA implementations. In Stefan Mangard and François-Xavier Standaert, editors, *CHES*, volume 6225 of *Lecture Notes in Computer Science*, pages 306–319. Springer, 2010. 11.1.1, 11.1.3

[BCG08] Alexandre Berzati, Cécile Canovas, and Louis Goubin. Perturbating RSA public keys: An improved attack. In Elisabeth Oswald and Pankaj Rohatgi, editors, *CHES*, volume 5154 of *Lecture Notes in Computer Science*, pages 380–395. Springer, 2008. 11.1.1, 11.1.3

[BCI+10a] Eric Brier, Jean-Sébastien Coron, Thomas Icart, David Madore, Hugues Randriam, and Mehdi Tibouchi. Efficient indifferentiable hashing into ordinary elliptic curves. In Tal Rabin, editor, *CRYPTO*, volume 6223 of *Lecture Notes in Computer Science*, pages 237–254. Springer, 2010. 2.1.3, 2.4.2, 3.5.1, 3.5.3, 3.5.4, 3.5.4, 4.5, 5.1, 6.1.1, 6.4.1, 6.4.3

[BCI10b] Julien Bringer, Hervé Chabanne, and Thomas Icart. Password based key exchange protocols on elliptic curves which conceal the public parameters. In Jianying Zhou and Moti Yung, editors, *ACNS*, volume 6123 of *Lecture Notes in Computer Science*, pages 291–308, 2010. 2.1.7, 3.6.2

[BCMCC06] Eric Brier, Benoît Chevallier-Mames, Mathieu Ciet, and Christophe Clavier. Why one should also secure RSA public key elements. In Louis Goubin and Mitsuru Matsui, editors, *CHES*, volume 4249 of *Lecture Notes in Computer Science*, pages 324–338. Springer, 2006. 11.1.1, 11.1.3

[BCN+10] Aurélie Bauer, Jean-Sébastien Coron, David Naccache, Mehdi Tibouchi, and Damien Vergnaud. On the broadcast and validity-checking security of PKCS#1 v1.5 encryption. In Jianying Zhou and Moti Yung, editors, *ACNS*, volume 6123 of *Lecture Notes in Computer Science*, pages 1–18, 2010. 2.2.4, 2.4.2, 12.1

[BD92] Jørgen Brandt and Ivan Damgård. On generation of probable primes by incremental search. In Ernest F. Brickell, editor, *CRYPTO*, volume 740 of *Lecture Notes in Computer Science*, pages 358–370. Springer, 1992. 2.3.2

[BD00] Dan Boneh and Glenn Durfee. Cryptanalysis of RSA with private key d less than $N^{0.292}$. *IEEE Transactions on Information Theory*, 46(4):1339, 2000. 2.2.6, II

[BDL91] Jørgen Brandt, Ivan Damgård, and Peter Landrock. Speeding up prime number generation. In Hideki Imai, Ronald L. Rivest, and Tsutomu Matsumoto, editors, *ASIACRYPT*, volume 739 of *Lecture Notes in Computer Science*, pages 440–449. Springer, 1991. 2.3.2

[BDL97] Dan Boneh, Richard A. DeMillo, and Richard J. Lipton. On the importance of checking cryptographic protocols for faults (extended abstract). In *EUROCRYPT*, pages 37–51, 1997. 10.1.1, 11.1.1

[BDL01] Dan Boneh, Richard A. DeMillo, and Richard J. Lipton. On the importance of eliminating errors in cryptographic computations. *J. Cryptology*, 14(2):101–119, 2001. 2.2.2, II, 10.1, 10.1.1, 11.1.1

[BECN+06] Hagai Bar-El, Hamid Choukri, David Naccache, Michael Tunstall, and Claire Whelan. The sorcerer's apprentice guide to fault attacks. *Proc. IEEE*, 94(2):370–382, 2006. 11.A.1, 11.A.1

[Ber68] Elwyn R. Berlekamp. *Algebraic Coding Theory*. McGraw-Hill, 1968. 5.3.1

[Ber04a] Daniel J. Bernstein. How to find smooth parts of integers. `http://cr.yp.to/papers.html#smoothparts`, May 2004. 2.2.1, 9.4, 9.4.1

[Ber04b] Daniel J. Bernstein. Scaled remainder trees. `http://cr.yp.to/papers.html#scaledmod`, August 2004. 9.4.1, 9.5.2

[Ber08] Daniel J. Bernstein. Fast multiplication and its applications. In *Algorithmic number theory: lattices, number fields, curves and cryptography*, volume 44 of *Math. Sci. Res. Inst. Publ.*, pages 325–384. Cambridge Univ. Press, Cambridge, 2008. 9.4.1, 13.3

[BF01] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the Weil pairing. In Joe Kilian, editor, *CRYPTO*, volume 2139 of *Lecture Notes in Computer Science*, pages 213–229. Springer, 2001. 1.2.2, 2.1.1, I, 3.1.1, 3.4.2, 5.1.1, 5.1.4, 5.2.2, 5.A.2, 7.1.2, 7.2

[BGLS03] Dan Boneh, Craig Gentry, Ben Lynn, and Hovav Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. In Eli Biham, editor, *EUROCRYPT*, volume 2656 of *Lecture Notes in Computer Science*, pages 416–432. Springer, 2003. 3.1.1, 5.1.1

[BGV11] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. Fully homomorphic encryption without bootstrapping. Cryptology ePrint Archive, Report 2011/277, 2011. `http://eprint.iacr.org/`. 2.3.1

[BJ07] Aurélie Bauer and Antoine Joux. Toward a rigorous variation of coppersmith's algorithm on three variables. In Moni Naor, editor, *EUROCRYPT*, volume 4515 of *Lecture Notes in Computer Science*, pages 361–378. Springer, 2007. 2.2.6

[BK01] Paulo S. L. M. Barreto and Hae Yong Kim. Fast hashing onto elliptic curves over fields of characteristic 3. Cryptology ePrint Archive, Report 2001/098, 2001. `http://eprint.iacr.org/`. 3, 3.5.4

[BL07a] Daniel J. Bernstein and Tanja Lange. Faster addition and doubling on elliptic curves. In Kaoru Kurosawa, editor, *ASIACRYPT*, volume 4833 of *Lecture Notes in Computer Science*, pages 29–50. Springer, 2007. 8.1.1

[BL07b] Daniel J. Bernstein and Tanja Lange. Inverted Edwards coordinates. In Serdar Boztas and Hsiao feng Lu, editors, *AAECC*, volume 4851 of *Lecture Notes in Computer Science*, pages 20–27. Springer, 2007. 8.1.1

[BL08] Daniel J. Bernstein and Tanja Lange. Explicit-formulas database, 2008. `http://www.hyperelliptic.org/EFD/`. 8.1.1

[Ble98] Daniel Bleichenbacher. Chosen ciphertext attacks against protocols based on the RSA encryption standard PKCS #1. In Hugo Krawczyk, editor, *CRYPTO*, volume 1462 of *Lecture Notes in Computer Science*, pages 1–12. Springer, 1998. 2.2.4, II, 9.1.1, 12.1.1, 12.3.2, 12.5.2, 12.7

[BLF08] Daniel J. Bernstein, Tanja Lange, and Reza Rezaeian Farashahi. Binary Edwards curves. In Elisabeth Oswald and Pankaj Rohatgi, editors, *CHES*, volume 5154 of *Lecture Notes in Computer Science*, pages 244–265. Springer, 2008. 8.1.1

[BLP08] Daniel J. Bernstein, Tanja Lange, and Christiane Peters. Attacking and defending the McEliece cryptosystem. In Johannes Buchmann and Jintai Ding, editors, *PQCrypto*, volume 5299 of *Lecture Notes in Computer Science*, pages 31–46. Springer, 2008. 9.E

[BLS01] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the Weil pairing. In Colin Boyd, editor, *ASIACRYPT*, volume 2248 of *Lecture Notes in Computer Science*, pages 514–532. Springer, 2001. 2.1.1, 3.1.1, 3.2.1, 3.2.2, 3.3, 3.3.1, 5.1.1

[BLS03] Paulo S. L. M. Barreto, Ben Lynn, and Michael Scott. On the selection of pairing-friendly groups. In Mitsuru Matsui and Robert J. Zuccherato, editors, *Selected Areas in Cryptography*, volume 3006 of *Lecture Notes in Computer Science*, pages 17–25. Springer, 2003. 8.4.2

[BLS04a] Paulo S. L. M. Barreto, Ben Lynn, and Michael Scott. Efficient implementation of pairing-based cryptosystems. *J. Cryptology*, 17(4):321–334, 2004. 8.4.1

[BLS04b] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the Weil pairing. *J. Cryptology*, 17(4):297–319, 2004. 2

[Blu70] Leo I. Bluestein. A linear filtering approach to the computation of the discrete Fourier transform. *IEEE Trans. Electroacoustics*, 18:451–455, 1970. 13.4

[BMN01] Colin Boyd, Paul Montague, and Khanh Quoc Nguyen. Elliptic curve based password authenticated key exchange protocols. In Vijay Varadharajan and Yi Mu, editors, *ACISP*, volume 2119 of *Lecture Notes in Computer Science*, pages 487–501. Springer, 2001. 3.3.2

[BMP00] Victor Boyko, Philip D. MacKenzie, and Sarvar Patel. Provably secure password-authenticated key exchange using Diffie-Hellman. In Bart Preneel, editor, *EUROCRYPT*, volume 1807 of *Lecture Notes in Computer Science*, pages 156–171. Springer, 2000. 3.1.1, 5.1.1

[BN05] Paulo S. L. M. Barreto and Michael Naehrig. Pairing-friendly elliptic curves of prime order. In Bart Preneel and Stafford E. Tavares, editors, *Selected Areas in Cryptography*, volume 3897 of *Lecture Notes in Computer Science*, pages 319–331. Springer, 2005. 2

[BNNT11a] Éric Brier, David Naccache, Phong Q. Nguyen, and Mehdi Tibouchi. Modulus fault attacks against RSA signatures. In Bart Preneel and Tsuyoshi Takagi, editors, *CHES*, volume 6917 of *Lecture Notes in Computer Science*. Springer, 2011. To appear. 2.2.3, 2.4.2, 11.1

[BNNT11b] Éric Brier, David Naccache, Phong Q. Nguyen, and Mehdi Tibouchi. Modulus fault attacks against RSA signatures. *J. Cryptographic Engineering*, 1(3), 2011. To appear. 2.2.3, 2.4.1, 11.1

[BNP07] Arnaud Boscher, Robert Naciri, and Emmanuel Prouff. CRT-RSA algorithm protected against fault attacks. In Damien Sauveron, Constantinos Markantonakis, Angelos Bilas, and Jean-Jacques Quisquater, editors, *WISTP*, volume 4462 of *Lecture Notes in Computer Science*, pages 229–243. Springer, 2007. 11.1.3

[BNT09] Eric Brier, David Naccache, and Mehdi Tibouchi. Factoring unbalanced moduli with known bits. In Donghoon Lee and Seokhie Hong, editors, *ICISC*, volume 5984 of *Lecture Notes in Computer Science*, pages 65–72. Springer, 2009. 2.2.6, 2.4.2

[Bol03] Alexandra Boldyreva. Threshold signatures, multisignatures and blind signatures based on the gap-Diffie-Hellman-group signature scheme. In Yvo Desmedt, editor, *Public Key Cryptography*, volume 2567 of *Lecture Notes in Computer Science*, pages 31–46. Springer, 2003. 3.1.1, 5.1.1

[Bom66] Enrico Bombieri. On exponential sums in finite fields. In *Les Tendances Géom. en Algèbre et Théorie des Nombres*, pages 37–41. Éditions du Centre National de la Recherche Scientifique, Paris, 1966. 2.1.4, 6.1.2

[Bon99] Dan Boneh. Twenty years of attacks on the RSA cryptosystem. *Notices Amer. Math. Soc.*, 46(2):203–213, 1999. II, 12.2.3

[Bos03] Alin Bostan. *Algorithmique efficace pour des opérations de base en calcul formel*. PhD thesis, École polytechnique, 2003. In English. 13.4, 13.5

[Boy03] Xavier Boyen. Multipurpose identity-based signcryption (a Swiss army knife for identity-based cryptography). In Dan Boneh, editor, *CRYPTO*, volume 2729 of *Lecture Notes in Computer Science*, pages 383–399. Springer, 2003. 3.1.1, 5.1.1

[BP96] Eric Bach and René Peralta. Asymptotic semismoothness probabilities. *Math. Comp.*, 65(216):1701–1715, 1996. 9.4, 9.4.2, 9.6

[BPS00] Olivier Baudron, David Pointcheval, and Jacques Stern. Extended notions of security for multicast public key cryptosystems. In Ugo Montanari, José D. P. Rolim, and Emo Welzl, editors, *ICALP*, volume 1853 of *Lecture Notes in Computer Science*, pages 499–511. Springer, 2000. 12.2.3

[BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *ACM Conference on Computer and Communications Security*, pages 62–73, 1993. 1.2.3, 3.1.1, 5.1.1, 5.2.1, 9.1.1

[BR94] Mihir Bellare and Phillip Rogaway. Optimal asymmetric encryption. In *EUROCRYPT*, pages 92–111, 1994. 1.2.3, 9.1.1, 12.1.1

[BR96] Mihir Bellare and Phillip Rogaway. The exact security of digital signatures: How to sign with RSA and Rabin. In *EUROCRYPT*, pages 399–416, 1996. 1.2.3, 9.1.1, 10.1.1, 10.8, 11.1.1

[BS05] Alin Bostan and Éric Schost. Polynomial evaluation and interpolation on special sets of points. *J. Complexity*, 21(4):420–446, 2005. 2.2.5, 13.3, 13.4

[BSS05] Ian F. Blake, Gadiel Seroussi, and Nigel P. Smart. *Advances in Elliptic Curve Cryptography*, volume 317 of *London Mathematical Society Lecture Note Series*, chapter V. Cambridge University Press, 2005. 8.2.3

[BV11] Zvika Brakerski and Vinod Vaikuntanathan. Fully homomorphic encryption from Ring-LWE and security for key dependent messages. In Phillip Rogaway, editor, *CRYPTO*, volume 6841 of *Lecture Notes in Computer Science*, pages 505–524. Springer, 2011. 2.3.1

[BZ04] Joonsang Baek and Yuliang Zheng. Identity-based threshold decryption. In Feng Bao, Robert H. Deng, and Jianying Zhou, editors, *Public Key Cryptography*, volume 2947 of *Lecture Notes in Computer Science*, pages 262–276. Springer, 2004. 3.1.1, 5.1.1

[Can09] Gaëtan Canivet. *Analyse des effets d'attaques par fautes et conception sécurisée sur plate-forme reconfigurable*. PhD thesis, Institut polytechnique de Grenoble, 2009. 11.A.1

[CC03] Jae Choon Cha and Jung Hee Cheon. An identity-based signature from gap Diffie-Hellman groups. In Yvo Desmedt, editor, *Public Key Cryptography*, volume 2567 of *Lecture Notes in Computer Science*, pages 18–30. Springer, 2003. 3.1.1, 5.1.1

[CCG+08] Don Coppersmith, Jean-Sébastien Coron, François Grieu, Shai Halevi, Charanjit S. Jutla, David Naccache, and Julien P. Stern. Cryptanalysis of ISO/IEC 9796-1. *J. Cryptology*, 21(1):27–51, 2008. II, 9.1.1

[CGH04] Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited. *J. ACM*, 51(4):557–594, 2004. 5.1.1

[CGIP11] Jean-Sébastien Coron, Aline Gouget, Thomas Icart, and Pascal Paillier. Supplemental access control (PACE v2): Security analysis of PACE Integrated Mapping. Cryptology ePrint Archive, Report 2011/058, 2011. `http://eprint.iacr.org/`. 2.1.7, 3.6.2

[CJ05] Mathieu Ciet and Marc Joye. Practical fault countermeasures for Chinese remaindering based cryptosystems. In L. Breveglieri and I. Koren, editors, *FDTC*, pages 124–131, 2005. 11.1.3

[CJK+09] Jean-Sébastien Coron, Antoine Joux, Ilya Kizhvatov, David Naccache, and Pascal Paillier. Fault attacks on RSA signatures with partially unknown messages. In Christophe Clavier and Kris Gaj, editors, *CHES*, volume 5747 of *Lecture Notes in Computer Science*, pages 444–456. Springer, 2009. 2.2.2, 10.1, 10.1.2, 10.1.3, 10.3.2, 10.4, 10.4.1, 10.4.2, 10.6, 10.6, 10.2, 10.7.2, 13.6

[CJM+11] Jean-Sébastien Coron, Antoine Joux, Avradip Mandal, David Naccache, and Mehdi Tibouchi. Cryptanalysis of the RSA subgroup assumption from TCC 2005. In Dario Catalano, Nelly Fazio, Rosario Gennaro, and Antonio Nicolosi, editors, *Public Key Cryptography*, volume 6571 of *Lecture Notes in Computer Science*, pages 147–155. Springer, 2011. 2.2.5, 2.4.2, 13.1

[CJNP00] Jean-Sébastien Coron, Marc Joye, David Naccache, and Pascal Paillier. New attacks on PKCS#1 v1.5 encryption. In Bart Preneel, editor, *EUROCRYPT*, volume 1807 of *Lecture Notes in Computer Science*, pages 369–381. Springer, 2000. II, 9.1.1, 12.1.1, 12.3.2, 12.7

[CJNP02] Jean-Sébastien Coron, Marc Joye, David Naccache, and Pascal Paillier. Universal padding schemes for RSA. In Moti Yung, editor, *CRYPTO*, volume 2442 of *Lecture Notes in Computer Science*, pages 226–241. Springer, 2002. 12.4, 12.4

[CK11] Jean-Marc Couveignes and Jean-Gabriel Kammerer. The geometry of flex tangents to a cubic curve and its parameterizations. Cryptology ePrint Archive, Report 2011/033, 2011. `http://eprint.iacr.org/`. 2.1.7, 3.5.2

[CM05] Benoît Chevallier-Mames. An efficient CDH-based signature scheme with a tight security reduction. In Victor Shoup, editor, *CRYPTO*, volume 3621 of *Lecture Notes in Computer Science*, pages 511–526. Springer, 2005. 3.1.1

[CM09] Jean-Sébastien Coron and Avradip Mandal. PSS is secure against random fault attacks. In Mitsuru Matsui, editor, *ASIACRYPT*, volume 5912 of *Lecture Notes in Computer Science*, pages 653–666. Springer, 2009. 10.8

[CMNT11] Jean-Sébastien Coron, Avradip Mandal, David Naccache, and Mehdi Tibouchi. Fully homomorphic encryption over the integers with shorter public keys. In Phillip Rogaway, editor, *CRYPTO*, volume 6841 of *Lecture Notes in Computer Science*, pages 487–504. Springer, 2011. 2.3.1, 2.4.2

[CND+06] Jean-Sébastien Coron, David Naccache, Yvo Desmedt, Andrew M. Odlyzko, and Julien P. Stern. Index calculation attacks on RSA signature and encryption. *Des. Codes Cryptography*, 38(1):41–53, 2006. 9.3.1

[CNS99] Jean-Sébastien Coron, David Naccache, and Julien P. Stern. On the security of RSA padding. In Michael J. Wiener, editor, *CRYPTO*, volume 1666 of *Lecture Notes in Computer Science*, pages 1–18. Springer, 1999. 2.2.1, 9.1, 9.1.2, 9.3.2, 9.4.1, 10.8

[CNT10] Jean-Sébastien Coron, David Naccache, and Mehdi Tibouchi. Fault attacks against EMV signatures. In Josef Pieprzyk, editor, *CT-RSA*, volume 5985 of *Lecture Notes in Computer Science*, pages 208–220. Springer, 2010. 2.2.2, 2.4.2, 10.1

[CNT11a] Jean-Sébastien Coron, David Naccache, and Mehdi Tibouchi. Another look at RSA signatures with affine padding. Cryptology ePrint Archive, Report 2011/057, 2011. `http://eprint.iacr.org/`. 2.2.7, 2.4.4

[CNT11b] Jean-Sébastien Coron, David Naccache, and Mehdi Tibouchi. Optimization of fully homomorphic encryption. Cryptology ePrint Archive, Report 2011/440, 2011. `http://eprint.iacr.org/`. 2.3.1, 2.4.4

[CNTW09] Jean-Sébastien Coron, David Naccache, Mehdi Tibouchi, and Ralf-Philipp Weinmann. Practical cryptanalysis of ISO/IEC 9796-2 and EMV signatures. In Shai Halevi, editor, *CRYPTO*, volume 5677 of *Lecture Notes in Computer Science*, pages 428–444. Springer, 2009. 2.2.1, 2.4.2, 9.1, 10.8

[Con07] Keith Conrad. Galois groups of cubics and quartics in all characteristics, 2007. `http://www.math.uconn.edu/~kconrad/blurbs/galoistheory/cubicquarticchar2.pdf`. 4.4, 4.A

[Cop94] Don Coppersmith. Solving homogeneous linear equations over GF(2) via block Wiedemann algorithm. *Math. Comp.*, 62(205):333–350, 1994. 9.5.2

[Cop96] Don Coppersmith. Finding a small root of a univariate modular equation. In *EUROCRYPT*, pages 155–165, 1996. 12.6.2

[Cop97] Don Coppersmith. Small solutions to polynomial equations, and low exponent RSA vulnerabilities. *J. Cryptology*, 10(4):233–260, 1997. 2.2.2, II, 10.1, 10.1.2, 12.6

[Cor00] Jean-Sébastien Coron. On the exact security of Full Domain Hash. In Mihir Bellare, editor, *CRYPTO*, volume 1880 of *Lecture Notes in Computer Science*, pages 229–235. Springer, 2000. 1

[Cor02] Jean-Sébastien Coron. Optimal security proofs for PSS and other signature schemes. In Lars R. Knudsen, editor, *EUROCRYPT*, volume 2332 of *Lecture Notes in Computer Science*, pages 272–287. Springer, 2002. 10.1.1, 11.1.1

[Cox04] David A. Cox. *Galois theory*. Pure and Applied Mathematics. Wiley-Interscience, 2004. 4.A

[CS03] Ronald Cramer and Victor Shoup. Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. *SIAM J. Comput.*, 33(1):167–226, 2003. 12.1.1

[CT11] Hervé Chabanne and Mehdi Tibouchi. Securing e-passports with elliptic curves. *IEEE Security & Privacy*, 9(2):75–78, 2011. 2.1.7, 2.4.3, 3.6.2

[DBP+02] Frédéric Darracq, Thomas Beauchene, Vincent Pouget, Hervé Lapuyade, Dean Lewis, Pascal Fouillat, and André Touboul. Single-event sensitivity of a single SRAM cell. *IEEE Trans. Nuclear Sci.*, 49:1486–1490, 2002. 11.A.1, 11.A.1

[DGK07] Ivan Damgård, Martin Geisler, and Mikkel Krøigaard. Efficient and secure comparison for on-line auctions. In Josef Pieprzyk, Hossein Ghodosi, and Ed Dawson, editors, *ACISP*, volume 4586 of *Lecture Notes in Computer Science*, pages 416–430. Springer, 2007. 13.2

[Dic30] Karl Dickman. On the frequency of numbers containing prime factors of a certain relative magnitude. *Arkiv för matematik, astronomi och fysik*, 22A(10):1–14, 1930. 9.4.2

[DJ11] Julien Devigne and Marc Joye. Binary Huff curves. In Aggelos Kiayias, editor, *CT-RSA*, volume 6558 of *Lecture Notes in Computer Science*, pages 340–355. Springer, 2011. 2.1.7, 8.5

[dJC85] Wiebren de Jonge and David Chaum. Attacks on some RSA signatures. In Hugh C. Williams, editor, *CRYPTO*, volume 218 of *Lecture Notes in Computer Science*, pages 18–27. Springer, 1985. 2.2.7

[DO85] Yvo Desmedt and Andrew M. Odlyzko. A chosen text attack on the RSA cryptosystem and some discrete logarithm schemes. In Hugh C. Williams, editor, *CRYPTO*, volume 218 of *Lecture Notes in Computer Science*, pages 516–522. Springer, 1985. 2.2.1, 9.1.2

[DS08] M. Prem Laxman Das and Palash Sarkar. Pairing computation on twisted Edwards form elliptic curves. In Steven D. Galbraith and Kenneth G. Paterson, editors, *Pairing*, volume 5209 of *Lecture Notes in Computer Science*, pages 192–210. Springer, 2008. 8.1.1

[Edw07] Harold M. Edwards. A normal form for elliptic curves. *Bull. Am. Math. Soc., New Ser.*, 44(3):393–422, 2007. 8.1.1

[EGA III.1] Alexander Grothendieck and Jean Dieudonné. Éléments de géométrie algébrique III. Étude cohomologique des faisceaux cohérents, première partie. *Publ. Math. IHES*, 11:5–167, 1961. 5.3.3

[EMV] EMV. Integrated circuit card specifications for payment systems. Book 2. Security and key management. Version 4.2, June 2008. `http://www.emvco.com/`. II, 9.7, 9.7.1, 10.1.2, 10.3.1, 10.7.1

[EMV TC] EMV. EMVCo type approval terminal level 2 test cases. Version 4.2a., April 2009. `http://www.emvco.com/`. 10.7.1

[Far11] Reza Rezaeian Farashahi. Hashing into hessian curves. In Abderrahmane Nitaj and David Pointcheval, editors, *AFRICACRYPT*, volume 6737 of *Lecture Notes in Computer Science*, pages 278–289. Springer, 2011. 2.1.7, 3.5.2, 6.1.1

[FFS+11] Reza R. Farashahi, Pierre-Alain Fouque, Igor E. Shparlinski, Mehdi Tibouchi, and J. Felipe Voloch. Indifferentiable deterministic hashing to elliptic and hyperelliptic curves. *Math. Comput.*, 2011. To appear. 2.1.4, 2.4.1, 6.1.2

[FIPS186–3] FIPS PUB 186-3. *Digital Signature Standard (DSS)*. NIST, USA, 2009. I, 5.5.3

[FJ05] Michael D. Fried and Moshe Jarden. *Field arithmetic*, volume 11 of *Ergebnisse der Mathematik und ihrer Grenzgebiete*. Springer-Verlag, Berlin, second edition, 2005. 4.3.2

[FKT03] Eisaku Furukawa, Mitsuru Kawazoe, and Tetsuya Takahashi. Counting points for hyperelliptic curves of type $y^2 = x^5 + ax$ over finite prime fields. In Mitsuru Matsui and Robert J. Zuccherato, editors, *Selected Areas in Cryptography*, volume 3006 of *Lecture Notes in Computer Science*, pages 26–41. Springer, 2003. 3.5.3, 3.5.4, 7.1.2, 7.2

[FOPS04] Eiichiro Fujisaki, Tatsuaki Okamoto, David Pointcheval, and Jacques Stern. RSA-OAEP is secure under the RSA assumption. *J. Cryptology*, 17(2):81–104, 2004. 12.1.1

[FS11] David Mandell Freeman and Takakazu Satoh. Constructing pairing-friendly hyperelliptic curves using Weil restriction. *J. Number Theory*, 131(5):959–983, 2011. 7.1.2, 7.2

[FSV10] Reza R. Farashahi, Igor E. Shparlinski, and José Felipe Voloch. On hashing into elliptic curves. *J. Math. Cryptology*, 3:353–360, 2010. 4.1.2, 6.4.1, 6.4.1

[FT10a] Pierre-Alain Fouque and Mehdi Tibouchi. Deterministic encoding and hashing to odd hyperelliptic curves. In Marc Joye, Atsuko Miyaji, and Akira Otsuka, editors, *Pairing*, volume 6487 of *Lecture Notes in Computer Science*, pages 265–277. Springer, 2010. 2.1.5, 2.4.2, 3.5.3, 6.4.1, 6.4.3, 7.1

[FT10b] Pierre-Alain Fouque and Mehdi Tibouchi. Estimating the size of the image of deterministic hash functions to elliptic curves. In Michel Abdalla and Paulo S. L. M. Barreto, editors, *LATINCRYPT*, volume 6212 of *Lecture Notes in Computer Science*, pages 81–91. Springer, 2010. 2.1.2, 2.4.2, 4.1

[FT11] Pierre-Alain Fouque and Mehdi Tibouchi. Close to uniform prime number generation with fewer random bits. Cryptology ePrint Archive, Report 2011/481, 2011. `http://eprint.iacr.org/`. 2.3.2, 2.4.4

[Gen09] Craig Gentry. Fully homomorphic encryption using ideal lattices. In Michael Mitzenmacher, editor, *STOC*, pages 169–178. ACM, 2009. 2.3.1

[GH11] Craig Gentry and Shai Halevi. Implementing Gentry's fully-homomorphic encryption scheme. In Kenneth G. Paterson, editor, *EUROCRYPT*, volume 6632 of *Lecture Notes in Computer Science*, pages 129–148. Springer, 2011. 2.3.1

[Gir06] Christophe Giraud. An RSA implementation resistant to fault attacks and to simple power analysis. *IEEE Trans. Computers*, 55(9):1116–1120, 2006. 10.8, 11.1.3

[GKZ07] Pierrick Gaudry, Alexander Kruppa, and Paul Zimmermann. A GMP-based implementation of Schönhage-Strassen's large integer multiplication algorithm. In Dongming Wang, editor, *ISSAC*, pages 167–174. ACM, 2007. 9.4.1

[GM82] Shafi Goldwasser and Silvio Micali. Probabilistic encryption and how to play mental poker keeping secret all partial information. In *STOC*, pages 365–377. ACM, 1982. 1.2.3

[GM84] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *J. Comput. Syst. Sci.*, 28(2):270–299, 1984. 1.2.3

[GM97] Marc Girault and Jean-François Misarsky. Selective forgery of RSA signatures using redundancy. In *EUROCRYPT*, pages 495–507, 1997. 2.2.7

[GMR84] Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A "paradoxical" solution to the signature problem. In *FOCS*, pages 441–448. IEEE, 1984. 1.2.3

[GMR88] Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM J. Comput.*, 17(2):281–308, 1988. 1.2.3

[Gri00] François Grieu. A chosen messages attack on the ISO/IEC 9796-1 signature scheme. In Bart Preneel, editor, *EUROCRYPT*, volume 1807 of *Lecture Notes in Computer Science*, pages 70–80. Springer, 2000. II, 9.1.1

[Gro05] Jens Groth. Cryptography in subgroups of $\mathbb{Z}_N^*$. In Joe Kilian, editor, *TCC*, volume 3378 of *Lecture Notes in Computer Science*, pages 50–65. Springer, 2005. 2.2.5, II, 13.1, 13.1.1, 13.1.2, 13.1, 13.5, 13.6

[Gro08] The PARI Group. *PARI/GP, Version 2.3.4*, 2008. `http://pari.math.u-bordeaux.fr`. 9.5.2

[GS02] Craig Gentry and Alice Silverberg. Hierarchical ID-based cryptography. In Yuliang Zheng, editor, *ASIACRYPT*, volume 2501 of *Lecture Notes in Computer Science*, pages 548–566. Springer, 2002. 3.1.1, 5.1.1

[H+09] William B. Hart et al. *Multi Precision Integers and Rationals library*, 2009. `http://www.mpir.org`. 9.5.2, 9.5.2

[Har77] Robin Hartshorne. *Algebraic Geometry*, volume 52 of *Graduate Texts in Mathematics*. Springer, 1977. 5.3.2, 5.3.3

[Has36] Helmut Hasse. Zur Theorie der abstrakten elliptischen Funktionenkörper III. Die Struktur des Meromorphismenrings; die Riemannsche Vermutung. *J. Reine Angew. Math.*, 175:193–208, 1936. 3.3.1

[Hås88] Johan Håstad. Solving simultaneous modular equations of low degree. *SIAM J. Comput.*, 17(2):336–341, 1988. 2.2.4, II, 12.6

[Hel76] Whitfield Diffie Martin Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976. 1.2.1, 1.2.2, II

[HG97] Nick Howgrave-Graham. Finding small roots of univariate modular equations revisited. In Michael Darnell, editor, *IMA Int. Conf.*, volume 1355 of *Lecture Notes in Computer Science*, pages 131–142. Springer, 1997. 12.6.2, 12.2

[HH+10] William B. Hart, David Harvey, et al. *Fast Library for Number Theory*, 2010. `http://www.flintlib.org`. 13.5

[HKT05] Mitsuhiro Haneda, Mitsuru Kawazoe, and Tetsuya Takahashi. Suitable curves for genus-4 HCC over prime fields. In Luís Caires, Giuseppe F. Italiano, Luís Monteiro, Catuscia Palamidessi, and Moti Yung, editors, *ICALP*, volume 3580 of *Lecture Notes in Computer Science*, pages 539–550. Springer, 2005. 3.5.3, 3.5.4, 7.1.2, 7.2

[HL02] Jeremy Horwitz and Ben Lynn. Toward hierarchical identity-based encryption. In Lars R. Knudsen, editor, *EUROCRYPT*, volume 2332 of *Lecture Notes in Computer Science*, pages 466–481. Springer, 2002. 3.1.1, 5.1.1

[HM08] Mathias Herrmann and Alexander May. Solving linear equations modulo divisors: On factoring given any bits. In Josef Pieprzyk, editor, *ASIACRYPT*, volume 5350 of *Lecture Notes in Computer Science*, pages 406–424. Springer, 2008. 10.1.2, 10.1

[HQZ04] Guillaume Hanrot, Michel Quercia, and Paul Zimmermann. The middle product algorithm I. *Appl. Algebra Eng. Commun. Comput.*, 14(6):415–438, 2004. 13.3

[Huf48] Gerald B. Huff. Diophantine problems in geometry and elliptic ternary forms. *Duke Math. J.*, 15:443–453, 1948. 2.1.6, 8.1.1, 8.1

[HWCD08] Hüseyin Hisil, Kenneth Koon-Ho Wong, Gary Carter, and Ed Dawson. Twisted Edwards curves revisited. In Josef Pieprzyk, editor, *ASIACRYPT*, volume 5350 of *Lecture Notes in Computer Science*, pages 326–343. Springer, 2008. 8.1.1

[Ica09] Thomas Icart. How to hash into elliptic curves. In Shai Halevi, editor, *CRYPTO*, volume 5677 of *Lecture Notes in Computer Science*, pages 303–316. Springer, 2009. 2.1.1, 2.1.2, 3.5.2, 3.6.1, 4.1, 4.2.1, 4.2.2, 4.6, 5.3, 5.3.2, 5.3.3, 6.1.1, 6.4.1

[ICAO10] ISO/IEC JTC1 SC17 WG3/TF5. *Supplemental Access Control for Machine Readable Travel Documents, version 1.01.* International Civil Aviation Organization, 2010. `http://mrtd.icao.int/`. I, 3.6.2

[IEEE P1363] IEEE P1363-2000. *Standard Specifications For Public Key Cryptography.* IEEE, 2000. 11.1.3

[IJ08] Sorina Ionica and Antoine Joux. Another approach to pairing computation in Edwards coordinates. In Dipanwita Roy Chowdhury, Vincent Rijmen, and Abhijit Das, editors, *INDOCRYPT*, volume 5365 of *Lecture Notes in Computer Science*, pages 400–413. Springer, 2008. 8.1.1

[ISO18033–2] ISO/IEC 18033-2:2006. *Information technology – Security techniques – Encryption algorithms – Part 2: Asymmetric ciphers.* ISO, Geneva, Switzerland, 2006. 1.2.3, I, 12.1.1

[ISO8825–1] ISO/IEC 8825-1:2002. *Information technology – ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER).* ISO, Geneva, Switzerland, 2002. 10.7.1

[ISO9796–1] ISO/IEC 9796-1. *Information Technology – Security Techniques – Digital signature scheme giving message recovery, Part 1: Mechanisms using redundancy*. ISO, Geneva, Switzerland, 1999. 1.2.3, 9.1, 9.1.1

[ISO9796–2] ISO/IEC 9796-2. *Information Technology – Security Techniques – Digital signature scheme giving message recovery, Part 2: Mechanisms using a hash-function*. ISO, Geneva, Switzerland, 1997. 1.2.3, 2.2.1, 9.1, 9.1.1, 9.1.2, 9.2, 10.3.1

[ISO9796–2:2010] ISO/IEC 9796-2:2010. *Information Technology – Security Techniques – Digital signature scheme giving message recovery, Part 2: Integer factorization based mechanisms*. ISO, Geneva, Switzerland, 2010. 2.2.1, 9.1, 9.2, 9.8

[ISO9796–2:2002] ISO/IEC 9796-2:2002. *Information Technology – Security Techniques – Digital signature scheme giving message recovery, Part 2: Integer factorization based mechanisms*. ISO, Geneva, Switzerland, 2002. 2.2.1, 2.2.1, II, 9.1, 9.1.1, 9.1.2, 9.2, 10.1.2, 10.3.1

[Jab96] David P. Jablon. Strong password-only authenticated key exchange. *SIGCOMM Comput. Commun. Rev.*, 26:5–26, October 1996. 3.1.1, 3.3.2, 5.1.1

[Jab97] David P. Jablon. Extended password key exchange protocols immune to dictionary attacks. In *WETICE*, pages 248–255. IEEE Computer Society, 1997. 3.3.2

[JLQ99] Marc Joye, Arjen K. Lenstra, and Jean-Jacques Quisquater. Chinese remaindering based cryptosystems in the presence of faults. *J. Cryptology*, 12(4):241–245, 1999. 10.1.1

[JM06] Ellen Jochemsz and Alexander May. A strategy for finding roots of multivariate polynomials with new applications in attacking RSA variants. In Xuejia Lai and Kefei Chen, editors, *ASIACRYPT*, volume 4284 of *Lecture Notes in Computer Science*, pages 267–282. Springer, 2006. 2.2.4, 12.6.1, 12.6.2

[JNT07] Antoine Joux, David Naccache, and Emmanuel Thomé. When e-th roots become easier than factoring. In Kaoru Kurosawa, editor, *ASIACRYPT*, volume 4833 of *Lecture Notes in Computer Science*, pages 13–28. Springer, 2007. II, 9.1.1

[Jou00] Antoine Joux. A one round protocol for tripartite Diffie-Hellman. In Wieb Bosma, editor, *ANTS*, volume 1838 of *Lecture Notes in Computer Science*, pages 385–394. Springer, 2000. 1.2.2, I

[Jou02] Antoine Joux. The Weil and Tate pairings as building blocks for public key cryptosystems. In Claus Fieker and David R. Kohel, editors, *ANTS*, volume 2369 of *Lecture Notes in Computer Science*, pages 20–32. Springer, 2002. 3.5.4, 7.2, 7.2

[JP06] Marc Joye and Pascal Paillier. Fast generation of prime numbers on portable devices: An update. In Louis Goubin and Mitsuru Matsui, editors, *CHES*, volume 4249 of *Lecture Notes in Computer Science*, pages 160–173. Springer, 2006. 2.3.2

[JPV00] Marc Joye, Pascal Paillier, and Serge Vaudenay. Efficient generation of prime numbers. In Çetin Kaya Koç and Christof Paar, editors, *CHES*, volume 1965 of *Lecture Notes in Computer Science*, pages 340–354. Springer, 2000. 2.3.2

[JPY01] Marc Joye, Pascal Paillier, and Sung-Ming Yen. Secure evaluation of modular functions. In R. J. Hwang and C. K. Wu, editors, *2001 International Workshop on Cryptology and Network Security*, pages 227–229, Taipei, Taiwan, 2001. 10.8

[JTV10] Marc Joye, Mehdi Tibouchi, and Damien Vergnaud. Huff's model for elliptic curves. In Guillaume Hanrot, François Morain, and Emmanuel Thomé, editors, *ANTS*, volume 6197 of *Lecture Notes in Computer Science*, pages 234–250. Springer, 2010. 2.1.6, 2.4.2, 8.1

[KAF+10] Thorsten Kleinjung, Kazumaro Aoki, Jens Franke, Arjen K. Lenstra, Emmanuel Thomé, Joppe W. Bos, Pierrick Gaudry, Alexander Kruppa, Peter L. Montgomery, Dag Arne Osvik, Herman J. J. te Riele, Andrey Timofeev, and Paul Zimmermann. Factorization of a 768-bit RSA modulus. In Tal Rabin, editor, *CRYPTO*, volume 6223 of *Lecture Notes in Computer Science*, pages 333–350. Springer, 2010. II, 13.5

[KJJ99] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In Michael J. Wiener, editor, *CRYPTO*, volume 1666 of *Lecture Notes in Computer Science*, pages 388–397. Springer, 1999. II

[KKM11] Ann Hibner Koblitz, Neal Koblitz, and Alfred Menezes. Elliptic curve cryptography: The serpentine course of a paradigm shift. *J. Number Theory*, 131(5):781–814, 2011. I

[KL99] Erich Kaltofen and Austin A. Lobo. Distributed matrix-free solution of large sparse linear systems over finite fields. *Algorithmica*, 24(3-4):331–348, 1999. 9.5.2

[KLR10] Jean-Gabriel Kammerer, Reynald Lercier, and Guénaël Renault. Encoding points on hyperelliptic curves over finite fields in deterministic

polynomial time. In Marc Joye, Atsuko Miyaji, and Akira Otsuka, editors, *Pairing*, volume 6487 of *Lecture Notes in Computer Science*, pages 278–297. Springer, 2010. 2.1.1, 3.5.2, 3.5.3, 3.5.3, 6.4.2, 6.4.2, 6.4.2, 7.1.1

[KM11] Neal Koblitz and Alfred Menezes. Another look at security definitions. Cryptology ePrint Archive, Report 2011/343, 2011. `http://eprint.iacr.org/`. 2

[Kob87] Neal Koblitz. Elliptic curve cryptosystems. *Math. Comp.*, 48:203–209, 1987. 1.2.2, I

[Koc96] Paul C. Kocher. Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In Neal Koblitz, editor, *CRYPTO*, volume 1109 of *Lecture Notes in Computer Science*, pages 104–113. Springer, 1996. II

[KQ07] Chong Hee Kim and Jean-Jacques Quisquater. Fault attacks for CRT based RSA: New attacks, new results, and new countermeasures. In Damien Sauveron, Constantinos Markantonakis, Angelos Bilas, and Jean-Jacques Quisquater, editors, *WISTP*, volume 4462 of *Lecture Notes in Computer Science*, pages 215–228. Springer, 2007. 10.8

[KS00] David R. Kohel and Igor Shparlinski. On exponential sums and group generators for elliptic curves over finite fields. In Wieb Bosma, editor, *ANTS*, volume 1838 of *Lecture Notes in Computer Science*, pages 395–404. Springer, 2000. 6.3, 7.4.2

[KT08] Mitsuru Kawazoe and Tetsuya Takahashi. Pairing-friendly hyperelliptic curves with ordinary jacobians of type $y^2 = x^5 + ax$. In Steven D. Galbraith and Kenneth G. Paterson, editors, *Pairing*, volume 5209 of *Lecture Notes in Computer Science*, pages 164–177. Springer, 2008. 3.5.4, 7.1.2, 7.2

[KW89] Luise-Charlotte Kappe and Bette Warren. An elementary test for the Galois group of a quartic polynomial. *Amer. Math. Monthly*, 96(2):133–137, 1989. 4.5, 4.A

[KW03] Jonathan Katz and Nan Wang. Efficiency improvements for signature schemes with tight security reductions. In Sushil Jajodia, Vijayalakshmi Atluri, and Trent Jaeger, editors, *ACM Conference on Computer and Communications Security*, pages 155–164. ACM, 2003. 1

[LJMP90] Arjen K. Lenstra, Hendrik W. Lenstra Jr., Mark S. Manasse, and John M. Pollard. The number field sieve. In *STOC*, pages 564–572. ACM, 1990. 1.2.2, II

[LL93] Arjen K. Lenstra and Hendrik W. Lenstra, Jr., editors. *The development of the number field sieve*, volume 1554 of *Lecture Notes in Mathematics*. Springer-Verlag, Berlin, 1993. II

[LLL82] Arjen K. Lenstra, Hendrik W. Lenstra, Jr., and L. Lovász. Factoring polynomials with rational coefficients. *Math. Ann.*, 261(4):515–534, 1982. 9.7.2, 10.2.4, 2, 12.6.2

[Lob95] Austin A. Lobo. *WLSS2: an implementation of the homogeneous block Wiedemann algorithm*, 1995. `http://www4.ncsu.edu/~kaltofen/software/wiliss`. 9.5.2

[LQ04] Benoît Libert and Jean-Jacques Quisquater. Efficient signcryption with key privacy from gap Diffie-Hellman groups. In Feng Bao, Robert H. Deng, and Jianying Zhou, editors, *Public Key Cryptography*, volume 2947 of *Lecture Notes in Computer Science*, pages 187–200. Springer, 2004. 3.1.1, 5.1.1

[Mau89] Ueli M. Maurer. Fast generation of secure RSA-moduli with almost maximal diversity. In *EUROCRYPT*, pages 636–647, 1989. 2.3.2

[Mau95] Ueli M. Maurer. Fast generation of prime numbers and secure public-key cryptographic parameters. *J. Cryptology*, 8(3):123–155, 1995. 2.3.2

[McE78] Robert J. McEliece. A public-key cryptosystem based on algebraic coding theory. *The Deep Space Network Progress Report*, 42-44:114–116, 1978. 1.2.2

[MDT+10] Amir Pasha Mirbaha, Jean-Max Dutertre, Assia Tria, Michel Agoyan, Anne-Lise Ribotta, and David Naccache. Study of single-bit fault injection techniques by laser on an AES cryptosystem. In D. Gizopoulos and A. Chatterjee, editors, *IOLTS*, 2010. 11.4

[MH78] Ralph C. Merkle and Martin E. Hellman. Hiding information and signatures in trapdoor knapsacks. *IEEE Transactions on Information Theory*, 24(5):525–530, 1978. 1.2.2

[Mih94] Preda Mihailescu. Fast generation of provable primes using search in arithmetic progressions. In Yvo Desmedt, editor, *CRYPTO*, volume 839 of *Lecture Notes in Computer Science*, pages 282–293. Springer, 1994. 2.3.2

[Mil85] Victor S. Miller. Use of elliptic curves in cryptography. In Hugh C. Williams, editor, *CRYPTO*, volume 218 of *Lecture Notes in Computer Science*, pages 417–426. Springer, 1985. 1.2.2, I

[Mil04] Victor S. Miller. The Weil pairing, and its efficient calculation. *J. Cryptology*, 17(4):235–261, 2004. 8.4.1

[Min96] Hermann Minkowski. *Geometrie der Zahlen.* Teubner-Verlag, 1896. 10.2.4

[Mis98] Jean-François Misarsky. How (not) to design RSA signature schemes. In Hideki Imai and Yuliang Zheng, editors, *Public Key Cryptography*, volume 1431 of *Lecture Notes in Computer Science*, pages 14–28. Springer, 1998. 9.1.2, 9.3.1

[MOV93] Alfred Menezes, Tatsuaki Okamoto, and Scott A. Vanstone. Reducing elliptic curve logarithms to logarithms in a finite field. *IEEE Transactions on Information Theory*, 39(5):1639–1646, 1993. 1.2.2

[MRH04] Ueli M. Maurer, Renato Renner, and Clemens Holenstein. Indifferentiability, impossibility results on reductions, and applications to the random oracle methodology. In Moni Naor, editor, *TCC*, volume 2951 of *Lecture Notes in Computer Science*, pages 21–39. Springer, 2004. 2.1.3, 3.3.1, 5.1.3, 5.2.1, 5.1, 5.2.1

[Mui06] James A. Muir. Seifert's RSA fault attack: Simplified analysis and generalizations. In Peng Ning, Sihan Qing, and Ninghui Li, editors, *ICICS*, volume 4307 of *Lecture Notes in Computer Science*, pages 420–434. Springer, 2006. 11.1.1, 11.1.3

[MVO91] Alfred Menezes, Scott A. Vanstone, and Tatsuaki Okamoto. Reducing elliptic curve logarithms to logarithms in a finite field. In *STOC*, pages 80–89. ACM, 1991. 1.2.2

[MWZ98] Alfred J. Menezes, Yi-Hong Wu, and Robert J. Zuccherato. An elementary introduction to hyperelliptic curves. In Neal Koblitz, editor, *Algebraic Aspects of Cryptography*, volume 3 of *Algorithms and Computation in Mathematics*, pages 155–178. Springer, 1998. 7.4.1

[Ngu09] Phong Q. Nguyen. Public-key cryptanalysis. In I. Luengo, editor, *Recent Trends in Cryptography*, volume 477 of *Contemporary Mathematics*. AMS–RSME, 2009. 10.2, 11.2.2

[NS97] Phong Q. Nguyen and Jacques Stern. Merkle-Hellman revisited: A cryptoanalysis of the Qu-Vanstone cryptosystem based on group factorizations. In Burton S. Kaliski Jr., editor, *CRYPTO*, volume 1294 of *Lecture Notes in Computer Science*, pages 198–212. Springer, 1997. 2.2.2, II, 10.1.3, 10.5, 10.5, 11.1.2, 11.2.2, 11.2.2, 2

[NS98] Phong Q. Nguyen and Jacques Stern. Cryptanalysis of a fast public key cryptosystem presented at SAC '97. In Stafford E. Tavares and Henk Meijer, editors, *Selected Areas in Cryptography*, volume 1556 of *Lecture Notes in Computer Science*, pages 213–218. Springer, 1998. 2.2.2, 10.1.3, 11.2.2

[NS01] Phong Q. Nguyen and Jacques Stern. The two faces of lattices in cryptology. In Joseph H. Silverman, editor, *CaLC*, volume 2146 of *Lecture Notes in Computer Science*, pages 146–180. Springer, 2001. 11.2.2

[NSA05] National Security Agency. The case for elliptic curve cryptography, 2005. `http://www.nsa.gov/business/programs/elliptic_curve.shtml`. 1.2.2, I

[NT11] Phong Q. Nguyen and Mehdi Tibouchi. Lattice-based fault attacks on signatures. In Marc Joye and Michael Tunstall, editors, *Fault Analysis in Cryptography*. Springer, 2011. To appear. 2.4.3

[P+09] Christof Paar et al. *COPACOBANA: A codebreaker for DES and other ciphers*, 2009. `http://www.copacobana.org`. 9.5.2

[Pee54] William D. Peeples, Jr. Elliptic curves and rational distance sets. *Proc. Am. Math. Soc.*, 5:29–33, 1954. 8.1.1

[PKCS#1 v1.5] Burton S. Kaliski et al. *PKCS#1: RSA Encryption Standard, Version 1.5*. RSA Laboratories, November 1993. 1.2.3, II, 9.1.1, 10.1.1, 12.1.1

[PKCS#1 v2.0] Burton S. Kaliski et al. *PKCS#1: RSA Encryption Standard, Version 2.0*. RSA Laboratories, September 1998. 12.1.1

[PKCS#1 v2.1] Burton S. Kaliski et al. *PKCS#1: RSA Encryption Standard, Version 2.1*. RSA Laboratories, June 2002. 1.2.3, 9.2, 11.1.3

[Poi05] David Pointcheval. Provable security for public key schemes. In *Contemporary cryptology*, Adv. Courses Math. CRM Barcelona, pages 133–190. Birkhäuser, Basel, 2005. 12.2.2

[Pou07] Vincent Pouget. Test et analyse par faisceau laser: Plateforme et applications. Technical report, Journée thématique du GDR SOC-SIP, 2007. `http://www.lirmm.fr/soc_sip/6fev/GCT_R1_Pouget.pdf`. 11.5, 13.6

[PP99] Pascal Paillier and David Pointcheval. Efficient public-key cryptosystems provably secure against active adversaries. In Kwok-Yan Lam, Eiji Okamoto, and Chaoping Xing, editors, *ASIACRYPT*, volume 1716 of *Lecture Notes in Computer Science*, pages 165–179. Springer, 1999. 13.2

[PWGP03] Jan Pelzl, Thomas J. Wollinger, Jorge Guajardo, and Christof Paar. Hyperelliptic curve cryptosystems: Closing the performance gap to elliptic curves. In Colin D. Walter, Çetin Kaya Koç, and Christof Paar, editors, *CHES*, volume 2779 of *Lecture Notes in Computer Science*, pages 351–365. Springer, 2003. 3.5.3

[RAD78] Ron L. Rivest, Leonard M. Adleman, and Michael L. Dertouzos. On data banks and privacy homomorphisms. In Richard A. DeMillo, editor, *Foundations of Secure Computation*, pages 169–180. Academic Press, 1978. 2.3.1

[Riv09] Matthieu Rivain. Securing RSA against fault analysis by double addition chain exponentiation. In Marc Fischlin, editor, *CT-RSA*, volume 5473 of *Lecture Notes in Computer Science*, pages 459–480. Springer, 2009. 11.1.3

[Ros02] Michael Rosen. *Number Theory in Function Fields*, volume 210 of *Graduate Texts in Mathematics*. Springer, 2002. 6.3

[RSA78] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2):120–126, 1978. 1.2.1, 1.2.2, II, 9.1.1, 11.1.1, 12.2.3

[RSR69] Lawrence R. Rabiner, Ronald W. Schafer, and Charles M. Rader. The chirp z-transform algorithm and its application. *Bell System Tech. J.*, 48:1249–1292, 1969. 13.4

[RSS11] Thomas Ristenpart, Hovav Shacham, and Thomas Shrimpton. Careful with composition: Limitations of the indifferentiability framework. In Kenneth G. Paterson, editor, *EUROCRYPT*, volume 6632 of *Lecture Notes in Computer Science*, pages 487–506. Springer, 2011. 2.1.3, 4, 3.4.1, 5.1.3, 5.2.1

[S+10a] Nigel P. Smart et al. ECRYPT II yearly report on algorithms and key lengths. Technical report, European Network of Excellence in Cryptology II, March 2010. `http://www.ecrypt.eu.org/documents/D.SPA.13.pdf`. 1.1

[S+10b] W.A. Stein et al. *Sage Mathematics Software (Version 4.4)*. The Sage Development Team, 2010. `http://www.sagemath.org`. 9.5.2, 11.2.4

[SA02] Sergei P. Skorobogatov and Ross J. Anderson. Optical fault induction attacks. In Burton S. Kaliski Jr., Çetin Kaya Koç, and Christof Paar, editors, *CHES*, volume 2523 of *Lecture Notes in Computer Science*, pages 2–12. Springer, 2002. 11.A.1

[Sat09] Takakazu Satoh. Generating genus two hyperelliptic curves over large characteristic finite fields. In Antoine Joux, editor, *EUROCRYPT*, volume 5479 of *Lecture Notes in Computer Science*, pages 536–553. Springer, 2009. 3.5.3, 3.5.4, 7.1.2, 7.2, 7.2

[Sch85] René Schoof. Elliptic curves over finite fields and the computation of square roots mod p. *Math. Comp.*, 44(170):483–494, 1985. 2.1.1, 3.4.3

[Sei05] Jean-Pierre Seifert. On authenticated computing and RSA-based authentication. In Vijay Atluri, Catherine Meadows, and Ari Juels, editors, *ACM Conference on Computer and Communications Security*, pages 122–127. ACM, 2005. 11.1.1

[SH09] Hisayoshi Sato and Keisuke Hakuta. An efficient method of generating rational points on elliptic curves. *J. Math-for-Industry*, 1(A):33–44, 2009. 2.1.1, 3.5.1

[Sha73] Daniel Shanks. Five number-theoretic algorithms. In *Proceedings of the Second Manitoba Conference on Numerical Mathematics (Univ. Manitoba, Winnipeg, Man., 1972)*, pages 51–70. Congressus Numerantium, No. VII. Utilitas Math., 1973. 3.5.1

[Sha95] Adi Shamir. RSA for paranoids. *CryptoBytes Technical Newsletter*, 1(3):1–4, 1995. 2.2.6

[Sha99] Adi Shamir. Method and apparatus for protecting public key schemes from timing and fault attacks. US Patent #5,991,415, November 1999. Presented at the rump session of EUROCRYPT '97. 10.8

[Sho97] Victor Shoup. Lower bounds for discrete logarithms and related problems. In *EUROCRYPT*, pages 256–266, 1997. 2.1.7

[Sil86] Joseph H. Silverman. *The Arithmetic of Elliptic Curves*, volume 106 of *Graduate Texts in Mathematics*, chapter III. Springer-Verlag, 1986. 8.1

[Ska05] Mariusz Skałba. Points on elliptic curves over finite fields. *Acta Arith.*, 117:293–301, 2005. 2.1.1, 3.5.1, 3.2

[SS04] Andrejz Schinzel and Mariusz Skałba. On equations $y^2 = x^n + k$ in a finite field. *Bull. Pol. Acad. Sci. Math.*, 52(3):223–226, 2004. 3.4.3, 3.1, 3.5.1

[SS10] Damien Stehlé and Ron Steinfeld. Faster fully homomorphic encryption. In Masayuki Abe, editor, *ASIACRYPT*, volume 6477 of *Lecture Notes in Computer Science*, pages 377–394. Springer, 2010. 2.3.1

[SV07] Nigel P. Smart and Frederik Vercauteren. On computable isomorphisms in efficient asymmetric pairing-based systems. *Discrete Applied Mathematics*, 155(4):538–547, 2007. 2

[SV10] Nigel P. Smart and Frederik Vercauteren. Fully homomorphic encryption with relatively small key and ciphertext sizes. In Phong Q. Nguyen and David Pointcheval, editors, *Public Key Cryptography*, volume 6056 of *Lecture Notes in Computer Science*, pages 420–443. Springer, 2010. 2.3.1

[SvdW06] Andrew Shallue and Christiaan van de Woestijne. Construction of rational points on elliptic curves over finite fields. In Florian Hess, Sebastian Pauli, and Michael E. Pohst, editors, *ANTS*, volume 4076 of *Lecture Notes in Computer Science*, pages 510–524. Springer, 2006. 2.1.1, 3.5.1, 3.5.4, 4.1, 6.1.1, 7.3.2

[Tib11] Mehdi Tibouchi. A Nagell algorithm in any characteristic. In David Naccache, editor, *Festschrift Jean-Jacques Quisquater*, volume 6805 of *Lecture Notes in Computer Science*. Springer, 2011. To appear. 2.4.3

[Ula07] Maciej Ulas. Rational points on certain hyperelliptic curves over finite fields. *Bull. Pol. Acad. Sci. Math.*, 55(2):97–104, 2007. 2.1.1, 3.5.1, 3.3, 3.5.3, 4.5, 6.1.1, 7.1.1

[vDGHV10] Marten van Dijk, Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. Fully homomorphic encryption over the integers. In Henri Gilbert, editor, *EUROCRYPT*, volume 6110 of *Lecture Notes in Computer Science*, pages 24–43. Springer, 2010. 2.3.1

[Vig08] David Vigilant. RSA with CRT: A new cost-effective solution to thwart fault attacks. In Elisabeth Oswald and Pankaj Rohatgi, editors, *CHES*, volume 5154 of *Lecture Notes in Computer Science*, pages 130–145. Springer, 2008. 11.1.3

[Wei95] André Weil. *Basic number theory*. Classics in Mathematics. Springer-Verlag, Berlin, 1995. Reprint of the second (1973) edition. 2.1.4, 6.1.2, 6.3

[WF10] Hongfeng Wu and Rongquan Feng. Elliptic curves in Huff's model. Cryptology ePrint Archive, Report 2010/390, 2010. `http://eprint.iacr.org/`. 8.5

[Wie90] Michael J. Wiener. Cryptanalysis of short RSA secret exponents. *IEEE Transactions on Information Theory*, 36(3):553–558, 1990. II

[ZK02] Fangguo Zhang and Kwangjo Kim. ID-based blind signature and ring signature from pairings. In Yuliang Zheng, editor, *ASIACRYPT*, volume 2501 of *Lecture Notes in Computer Science*, pages 533–547. Springer, 2002. 3.1.1, 5.1.1