A Leakage-Resilient Pairing-Based Variant of the Schnorr Signature Scheme

David Galindo[1] and Srinivas Vivek[2]

[1] CNRS, Loria, France
david.galindo-chacon@loria.fr
[2] University of Luxembourg, Luxembourg
srinivasvivek.venkatesh@uni.lu

Abstract. Leakage-resilient cryptography aims at capturing side-channel attacks within the provable security framework. Currently there exists a plethora of schemes with provably secure guarantees against a variety of side-channel attacks. However, meeting the strongest security levels (resilience against continual leakage attacks) under the weakest assumptions leads currently to costly schemes. Additionally, recent results show the impossibility to achieve the strongest leakage-resilient security levels for cryptosystems whose secret key is uniquely determined by its public key.

The above justifies the use of stronger assumptions to achieve simpler, more efficient schemes, since most deployed and practical cryptosystems satisfy the above-mentioned uniqueness of the secret key property. In particular, the Schnorr-based leakage-resilient digital signature schemes proposed up to now are built by gluing together ℓ-copies of the basic signature scheme, resulting in a public key that admits exponentially-many secret keys. Furthermore, the space needed to store the secret key material is proportional to the leakage tolerated by these schemes.

We aim at designing a leakage-resilient variant of the Schnorr signature scheme whose secret key's storage space is constant, independently of the amount of leakage that it can tolerate. We assume that at any given time only the parts of the memory in use leak (split-state/only computation leaks information model); we ease the problem of exhibiting a security reduction by relying on generic groups (generic bilinear group model). We proceed by first proposing a pairing analogue of the Schnorr signature scheme, that we next transform to include split signing key updates. We give a leakage-resilience lower bound in generic bilinear groups against continual leakage attacks for the new scheme.

Keywords: Digital signatures, generic group model, leakage-resilient cryptography, continual leakage, efficiency, min-entropy.

1 Introduction

Over the last 30 years the theory of cryptography has been robustly built. It started with the proposal of simple, elegant and sound definitions [19,13,20], it was followed by plausibility results under the weakest assumptions, and currently culminating in the practical constructions used nowadays by the information security community [1].

Concurrently, the theory and practice of cryptanalysis has been no less successful. In particular, the exploitation of the physical nature of the devices where cryptographic primitives are run, pioneered from an academic perspective in [24,25,9], has rendered many of the beautiful and theoretically robust constructions broken. Typical examples of side channel information are the analysis of primitives' running-time, power consumption or electromagnetic radiation leak, to name just but the most well-known.

The area of provable security that provides security reductions even in the presence of secret key leakage is called *leakage-resilient cryptography* and it has been an increasingly active field in recent years. In this work we assume leakage to be *continual leakage*, i.e. the useful leakage data per signature invocation is bounded in length, but unbounded overall; and adhere to the *independent leakage/split-state model*, meaning that the computation can be divided into rounds, where each such round access independent parts of the memory that leak independently.

The continual split-state leakage model has been previously used in the works [15,31,23,16,18,26]. The first assumption is restrictive but overall reasonable; in practice many side-channel attacks only exploit a polylogarithmic amount of information. The second assumption allows us to divide the memory of a device, at every computing step, into two parts - an *active* and a *passive* part. The part of the memory being currently accessed by a computation is the active part, and only the active part leaks information at any given time. We stress that even if leakage is local with respect to each part of the memory, it still captures some global functions of the secret key, for instance any affine leakage function. We refer to the work by Dziembowski and Faust [14] and by Faust *et al.* [17] for a discussion on the significance and limitations of this leakage model.

In the last few years the interplay between provable security and side-channel attacks has experienced great progress, as the works [22,16,8,11,27,7] bear witness for the case of digital signatures. However, the schemes that do not use any idealized assumption (random oracle, generic groups), are much more involved than their non-leakage counterparts and depart significantly from the schemes in the standard cryptography tool-box. Interestingly, recent work by Wichs [38] seems to indicate that it might be impossible to achieve continual leakage-resilience for cryptosystems whose secret key is uniquely determined by its public key, unless we weaken the security model. Furthermore, existing strongly secure proposals are not yet efficient enough. A rough estimation of the efficiency of current leakage-resilient schemes is that they are a linear number of times in the security parameter slower than their non-leakage counterparts.

In this work we study a signature scheme secure against continual leakage in the split-state model that builds on the Schnorr signature scheme [33]. Notice that several works [21,3,16] have already built leakage-resilient signature schemes based on Schnorr. All of these works confirm the finding by Wichs: they are built by gluing together several copies of the basic Schnorr signature scheme (a technique that was first used by Okamoto [30]), and thus given its public key there are exponentially many possible secret keys. The works [21,3] only allow a bounded leakage during the life-time of the protocol, although in their model every part of the memory is susceptible to leak (as opposed to the split-state model); the work [16] uses the split-state model and allows roughly $1/36$ leakage ratio at every signing step, but the number of signature queries is bounded in advance. Our goal is to provide a Schnorr-like signature scheme where the secret key material to be stored is constant at any time, since in the aforementioned works the secret keys' storage is proportional to the leakage ratio allowed. In particular we propose a scheme where the secret key is uniquely determined by its public key, the secret key consists of only two group elements at any given time and it is unforgeable even if the number of adversarial signature queries is not known in advance.

Our positive results are of course far from the ideal achievement, that is, to prove leakage-resilience of the original Schnorr scheme instantiated over any cryptographic group $\mathbb{G}$ where the discrete logarithm problem is assumed to be hard. However, this is presently out of reach using standard techniques [38]. This is why we state our theorems with respect to a transposition of the modified Schnorr signature scheme to pairing groups, where the secret key is no longer $x \in \mathbb{Z}_p$ but $X = g^x \in \mathbb{G}$, where $\mathbb{G}$ is the base pairing group with $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$. This allows us to use an idealized model of computation called the *generic bilinear group* (GBG) model that will ease our analysis. We proceed by first showing that our transposition of the Schnorr signature scheme to pairing groups is existentially unforgeable [20] in the GBG model. This is achieved by showing that the security reduction in the generic group model [36] for elliptic-curve based Schnorr signatures recently given by Neven, Smart and Warinschi [29] can be translated to the GBG and allows to deal with data leakage. Secondly, we modify the pairing-based Schnorr scheme by multiplicatively sharing $X = X_1 \cdot X_2$, where $X_1, X_2 \in \mathbb{G}$, and by breaking the signing scheme into two phases, each one using the corresponding share X_1 or X_2. Again, at each new signature invocation a fresh sharing (X_1', X_2') of X is computed. Our main theorem (Theorem 2) states that allowing λ bits of leakage at each phase of every round overall decreases the security of the scheme by a factor of at most $2^{2\lambda}$ in our leakage model.

The GBG model has been previously used for stating leakage-resilience properties by Kiltz and Pietrzak [23], and Galindo and Vivek [18]. Kiltz and Pietrzak propose a bilinear version of the ElGamal key encapsulation mechanism which enjoys provable leakage-resilience in the presence of continual leakage. Their scheme is very efficient, less than a handful of times slower than standard ElGamal. Galindo and Vivek propose a leakage-resilient existentially unforgeable signature scheme based on the Boneh-Boyen identity-based encryption scheme [4]. Their scheme enjoys efficiency and leakage-resilience properties similar to the scheme by Kiltz and Pietrzak. Our Schnorr-like scheme has efficiency comparable to that of Galindo and Vivek's scheme. Additionally it is so far the only Schnorr-based leakage-resilient scheme whose secret key is uniquely determined by its public key (thus bypassing the impossibility result by Wichs [38] at the cost of the GBG assumption), and the secret key material storage is constant and independent of the leakage rate (two elements in the pairing base group $\mathbb{G}$).

Organization of the Paper. We start in Section 2 by recalling some basic facts and definitions. In Section 3, we introduce a bilinear variant of the Schnorr signature scheme and prove its security in the GBG model. In Section 4, we split the secret state of the bilinear Schnorr scheme and prove its leakage resilience under continual leakage in the GBG model. Finally, we conclude in Section 5 by summarizing the achievements and limitations of our methodology.

2 Definitions

In this section, we recollect some basic notions of security of signature schemes, bilinear groups, and the generic bilinear group model. We also describe the model of leakage we shall consider in this paper and formulate a definition of security of signature schemes in the presence of continual leakage. We adapt the leakage model specified in [23] to signature schemes, exactly as done in [18].

Let $\mathbb{Z}$ denote the set of integers and $\mathbb{Z}_p$ ($p > 0$) denote, depending upon the context, either the set of integers $\{0, 1, \ldots, p-1\}$ or the ring modulo p. We denote a random sampling of an element $a \in A$ from a set A, and also denote a (possibly probabilistic) output of an algorithm A, by $a \leftarrow A$. If we want to explicitly denote the randomness r used during the sampling/output, then we do so by $s \overset{r}{\leftarrow} S$. Unless otherwise mentioned or implicit from the context, any sampling is from an uniform distribution. The symbol ":=" is used to define a notation in an expression, as in $A := \mathbb{Z}$, or to explicitly indicate an output of a deterministic algorithm or a function.

A signature scheme $\Pi = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ consists of three probabilistic polynomial-time algorithms KeyGen, Sign, and Verify. Let κ denote the security parameter. $\mathsf{KeyGen}(\kappa)$ on input κ produces a public- and secret-key pair (pk, sk) along with other public parameters $\mathbb{PP}$. The algorithm $\mathsf{Sign}(sk, m)$ on input a secret key sk and a message $m \in M$, where M is the message space, outputs a signature σ. $\mathsf{Verify}(pk, m, \sigma)$ on input a public key pk, a message $m \in M$ and a signature σ, outputs a bit $b = 1$ meaning *valid*, or $b = 0$ meaning *invalid.* We require the following *correctness* requirement to be satisfied by Π:

$$\Pr[\mathsf{Verify}(pk, m, \mathsf{Sign}(sk, m)) = 1 \;:\; (pk, sk) \leftarrow \mathsf{KeyGen}(\kappa), m \in M] = 1.$$

The standard security notion for signature schemes is existential unforgeability under adaptive chosen-message attacks (EUF-CMA), and it is is defined through the following experiment:

```
Sign-Forge$_\Pi(\mathcal{A},\kappa)$                          | Sign-Oracle $\Omega_{sk}(m)$
    $(pk, sk) \leftarrow \mathsf{KeyGen}(\kappa)$             |     $w := w \cup m$
    $w := \emptyset$                                          |     $\sigma \leftarrow \mathsf{Sign}(sk, m)$
    $(m, \sigma) \leftarrow \mathcal{A}^{\Omega_{sk}(\cdot)}(pk)$ |     Return $\sigma$
    If $m \in w$, then return $b := 0$                        |
    $b \leftarrow \mathsf{Verify}(pk, m, \sigma)$             |
```

Definition 1. [Existential Unforgeability] *A signature scheme* Π *is* existentially unforgeable under adaptive chosen-message attacks, *in short "secure", if* $\Pr[b = 1]$ *is negligible in* Sign-Forge$_\Pi(\mathcal{A},\kappa)$ *for any efficient adversary* $\mathcal{A}$.

2.1 Leakage Model

We split the secret state into two parts that reside in different parts of the memory, and structure any computation that involves access to the secret state into a sequence of steps. Any step accesses only one part of the secret state (*active* part) and the other part (*passive* part) is assumed not to leak in the current step of computation. For simplicity, we define a security notion for leakage-resilient signature schemes assuming that the signing process is carried out in two steps. We also refer to a single invocation of the signature generation algorithm as a *round.*

Let us consider the problem of achieving leakage resilience under continual leakage even when a significant fraction of the bits of the secret state are leaked per round. Then it is necessary that the secret state must be *stateful*, i.e. the secret state must be refreshed during every round [23]. Otherwise, after many rounds the entire secret state will be completely leaked.

Formally, a stateful signature scheme $\Pi^* = (\mathsf{KeyGen}^*, \mathsf{Sign}_1^*, \mathsf{Sign}_2^*, \mathsf{Verify}^*)$ consists of four probabilistic polynomial-time algorithms KeyGen^*, Sign_1^*, Sign_2^* and Verify^*. $\mathsf{KeyGen}^*(\kappa)$ is same as the set-up phase KeyGen of Π except that instead of a "single" secret key sk, it outputs two initial secret states (S_0, S_0'). From the point of view of an adversary, the signing algorithm Sign of Π and $(\mathsf{Sign}_1^*, \mathsf{Sign}_2^*)$ have the same functionality. First, Sign_1^* is executed and later Sign_2^* is executed. That is, the i^{th} round of the signing process is carried out as:

$$(S_i, w_i) \overset{r_i}{\leftarrow} \mathsf{Sign}_1^*(S_{i-1}, m_i)\ ;\ (S_i', \sigma_i) \overset{r_i'}{\leftarrow} \mathsf{Sign}_2^*(S_{i-1}', w_i). \tag{1}$$

In the above expression, r_i and r_i' are the randomness used by Sign_1^* and Sign_2^*, respectively. The parameter w_i is some state information passed onto Sign_2^* by Sign_1^*. The signature σ_i is generated for the message m_i, and the internal state is updated from (S_{i-1}, S_{i-1}') to (S_i, S_i').

We model the leakage during signature generation by giving an adversary $\mathcal{A}$ access to a leakage oracle $\Omega^{\text{leak}}_{(S_{i-1}, S_{i-1}')}(\cdot)$. This oracle, in addition to giving $\mathcal{A}$ signatures for the messages of its choice, also allows $\mathcal{A}$ to obtain leakage from the computation used to generate signatures. More precisely, let λ be a *leakage parameter*. During the i^{th} signing round, $\mathcal{A}$ is allowed to specify two functions

f_i and h_i, each of range $\{0,1\}^\lambda$, that can be efficiently computed. The outputs of the leakage functions are

$$\Lambda_i = f_i(S_{i-1}, r_i) \; ; \; \Lambda'_i = h_i(S'_{i-1}, r'_i, w_i). \tag{2}$$

Since the value of m can be included in the description of f_i and h_i, hence it is not explicitly included as an input. Note that it also possible for $\mathcal{A}$ to specify h_i after obtaining Λ_i. But for simplicity of the exposition, we only describe here the case where f_i and h_i are specified along with the message m_i to the oracle. The security of the signature scheme Π^* in the presence of (continual) leakage is defined through the following experiment Sign-Forge-Leak$_{\Pi^*}(\mathcal{A}, \kappa, \lambda)$. In the description below, $|f_i|$ refers to the length of the output of f_i.

```
Sign-Forge-Leak$_{\Pi^*}(\mathcal{A}, \kappa, \lambda)$
    $(pk, (S_0, S'_0)) \leftarrow \mathsf{KeyGen}^*(\kappa)$
    $i := 1, w := \emptyset$
    $(m, \sigma) \leftarrow \mathcal{A}^{\Omega^{\text{leak}}_{(S_{i-1}, S'_{i-1})}(\cdot)}(pk)$
    If $m \in w$, then return $b := 0$
    $b \leftarrow \mathsf{Verify}^*(pk, m, \sigma)$
```

```
Sign-Leak-Oracle $\Omega^{\text{leak}}_{(S_{i-1}, S'_{i-1})}(m_i, f_i, h_i)$
    If $|f_i| \neq \lambda$ or $|h_i| \neq \lambda$, return $\perp$
    $(S_i, w_i) \overset{r_i}{\leftarrow} \mathsf{Sign}^*_1(S_{i-1}, m_i)$
    $(S'_i, \sigma_i) \overset{r'_i}{\leftarrow} \mathsf{Sign}^*_2(S'_{i-1}, w_i)$
    $\Lambda_i := f_i(S_{i-1}, r_i)$
    $\Lambda'_i := h_i(S'_{i-1}, r'_i, w_i)$
    $i := i + 1$
    $w := w \cup m_i$
    Return $(\sigma_i, \Lambda_i, \Lambda'_i)$
```

Definition 2. [Existential Unforgeability with Leakage] *A signature scheme* Π^* *is* existentially unforgeable under adaptive chosen-message attacks in the presence of (continual) leakage *if* $\Pr[b = 1]$ *is negligible in the Experiment* Sign-Forge-Leak$_{\Pi^*}(\mathcal{A}, \kappa, \lambda)$ *for any efficient adversary* $\mathcal{A}$.

2.2 Bilinear Groups

Let $\mathsf{BGen}(\kappa)$ be a probabilistic bilinear group generator that outputs $(\mathbb{G}, \mathbb{G}_T, p, e, g)$ such that:

1. $\mathbb{G} = \langle g \rangle$ and $\mathbb{G}_T$ are (multiplicatively written) cyclic groups of prime order p with binary operations $\cdot$ and $\star$, respectively. The size of p is κ bits.
2. $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ is a bilinear map that is:
   (a) bilinear: $\forall u, v \in \mathbb{G}$ and $\forall a, b \in \mathbb{Z}$, $e(u^a, v^b) = e(u, v)^{ab}$.
   (b) non-degenerate: $e(g, g) \neq 1$.

Such a group $\mathbb{G}$ is said to be a bilinear group if the above properties hold and the group operations in $\mathbb{G}$ and $\mathbb{G}_T$, and the map e are efficiently computable. The group $\mathbb{G}$ is called as *base group* and $\mathbb{G}_T$ as *target group*.

2.3 Generic Bilinear Group Model

The generic bilinear group (GBG) model [6] is an extension of the generic group model [36]. The encodings of the elements of $\mathbb{G}$ and $\mathbb{G}_T$ are given by random bijective maps $\xi : \mathbb{Z}_p \to \Xi$ and $\xi_T : \mathbb{Z}_p \to \Xi_T$, respectively, where Ξ and Ξ_T are sets of bit-strings. The group operations in $\mathbb{G}$ and $\mathbb{G}_T$, and evaluation of the bilinear map e are performed by three public oracles $\mathcal{O}$, $\mathcal{O}_T$ and $\mathcal{O}_e$, respectively, defined as follows. For all $a, b \in \mathbb{Z}_p$

- $\mathcal{O}(\xi(a), \xi(b)) := \xi(a + b \bmod p)$
- $\mathcal{O}_T(\xi_T(a), \xi_T(b)) := \xi_T(a + b \bmod p)$
- $\mathcal{O}_e(\xi(a), \xi(b)) := \xi_T(ab \bmod p)$

We assume that the (fixed) generator g of $\mathbb{G}$ satisfies $g = \xi(1)$, and also the (fixed) generator g_T of $\mathbb{G}_T$ satisfies $g_T = e(g, g) = \xi_T(1)$. The encoding of g is provided to all users of the group oracles. The users can thus efficiently sample random elements in both $\mathbb{G}$ and $\mathbb{G}_T$.

We further assume that $\Xi \cap \Xi_T = \phi$, $|\Xi| = |\Xi_T| = p$, and that the elements of Ξ and Ξ_T are efficiently recognizable. For instance, the encodings in Ξ can comprise of the binary representation of the set $\{0, 1, \ldots, p-1\}$, where every string begins with '0' and all are of uniform length. The encodings in Ξ_T are similarly defined but instead begin with '1'. Since the encodings are efficiently recognizable, the queries to a group oracle with an invalid encoding can be detected and an error can be raised. For simplicity, we assume that the users' queries to the oracles are all valid.

2.4 Min-Entropy

Let X be a finite random variable with probability distribution Pr. The *min-entropy* of X, denoted $\mathbf{H}_\infty(X)$, is defined as $\mathbf{H}_\infty(X) := -\log_2\left(\max_x \Pr[X = x]\right)$. Min-entropy is a standard measure of the worst-case predictability of a random variable. Let Z be a random variable. The *average conditional min-entropy* of X given Z, denoted $\tilde{\mathbf{H}}_\infty(X \mid Z)$, is defined as $\tilde{\mathbf{H}}_\infty(X \mid Z) := -\log_2\Big(\mathop{\mathbb{E}}_{z \leftarrow Z}\Big[\max_x \Pr[X = x \mid Z = z]\Big]\Big)$. Average conditional min-entropy is a measure of the worst-case predictability of a random variable given a correlated random variable.

Lemma 1. [[12]] *Let $f : X \to \{0,1\}^{\lambda'}$ be a function on X. Then $\tilde{\mathbf{H}}_\infty(X \mid f(X)) \geq \mathbf{H}_\infty(X) - \lambda'$.*

The following result is a variant of the Schwartz-Zippel Lemma [34,39,18].

Lemma 2. [Schwartz-Zippel; min-entropy version] *Let $\mathsf{F} \in \mathbb{Z}_p[\mathsf{X}_1, \ldots, \mathsf{X}_n]$ be a non-zero polynomial of (total) degree at most d. Let P_i $(i = 1, \ldots, n)$ be probability distributions on $\mathbb{Z}_p$ such that $\mathbf{H}_\infty(P_i) \geq \log p - \lambda'$, where $0 \leq \lambda' \leq \log p$. If $x_i \overset{P_i}{\leftarrow} \mathbb{Z}_p$ $(i = 1, \ldots, n)$ are independent, then $Pr[\mathsf{F}(x_1, \ldots, x_n) = 0] \leq 2^{\lambda'} \dfrac{d}{p}$.*

Corollary 1. *If $\lambda' < \log p - \omega(\log \log p)$ in Lemma 2, then $Pr[\mathsf{F}(x_1, \ldots, x_n) = 0]$ is negligible (in $\log p$).*

3 Basic Signature Scheme

We propose a bilinear variant of the Schnorr signature scheme [32,33].

Let $\mathsf{H} : \{0,1\}^* \to \mathbb{Z}_p$ be a hash function. The signature scheme $\Pi_{\mathsf{Sc}} = (\mathsf{KeyGen}_{\mathsf{Sc}}, \mathsf{Sign}_{\mathsf{Sc}}, \mathsf{Verify}_{\mathsf{Sc}})$, defined on the message space $\{0,1\}^*$, is as follows:

1. $\mathsf{KeyGen}_{\mathsf{Sc}}(\kappa)$: Compute $\mathbb{PP} := (\mathbb{G}, \mathbb{G}_T, p, e, g) \leftarrow \mathsf{BGen}(\kappa)$. Choose random $x \leftarrow \mathbb{Z}_p$. Set $X := g^x$, $g_T := e(g,g)$, and $X_T := e(g,X) = g_T^x$. The public key is $pk := (\mathbb{PP}, X_T, \mathsf{H})$ and the secret key is $sk := X$.
2. $\mathsf{Sign}_{\mathsf{Sc}}(sk, m)$: Choose a random $t \leftarrow \mathbb{Z}_p$. Set $\gamma := \mathsf{H}(g_T^t || m)$, $Y := g^t \cdot X^\gamma$ and $\sigma := (Y, \gamma)$. Output the signature σ.
3. $\mathsf{Verify}_{\mathsf{Sc}}(pk, m, \sigma)$: Let $\sigma = (Y, \gamma) \in \mathbb{G} \times \mathbb{Z}_p$. Set $\rho := e(Y,g) \star (g_T^x)^{-\gamma}$. Output the bit $b = 1$ (*valid*) if $\mathsf{H}(\rho || m) = \gamma$. Otherwise output $b = 0$ (*invalid*).

We now prove the security of the above scheme in the GBG model relative to two hardness assumptions about the hash function H that were introduced in [29], and which are recalled below. These two assumptions are *weaker* than collision-resistance [29]. We adapt the proof techniques of [29] to the bilinear setting.

Definition 3. [Random-Prefix (Second-) Preimage problem [29]] *The advantage of an adversary $\mathcal{A}$ in solving the Random-Prefix Preimage (**RPP**) problem (respectively, Random-Prefix Second-Preimage (**RPSP**) problem) for a hash function $\mathsf{H} : \{0,1\}^* \to \mathbb{Z}_p$, with prefix in a set of bit-strings D, is given by*

$$\mathsf{Adv}_{\mathsf{H}}^{\mathsf{RPP[D]}}(\mathcal{A}) = \Pr\left[\mathsf{H}(R||m) = \gamma \ : \ \gamma \leftarrow \mathcal{A}, \text{random } R \leftarrow D,\ m \leftarrow \mathcal{A}(R)\right],$$

$$\mathsf{Adv}_{\mathsf{H}}^{\mathsf{RPSP[D]}}(\mathcal{A}) = \Pr\left[\mathsf{H}(R||m) = \mathsf{H}(R||m') \ : \ m \leftarrow \mathcal{A}, \text{random } R \leftarrow D,\ m' \leftarrow \mathcal{A}(R),\ m' \neq m\right],$$

where the probability is taken over R and the coins of $\mathcal{A}$. The RPP *problem (respectively,* RPSP *problem) for* H *is said to be (t, ϵ) hard if no adversary $\mathcal{A}$ with running time at most t has advantage greater than ϵ in solving it.*

Theorem 1. *The signature scheme Π_{Sc} is* EUF-CMA *secure in the generic bilinear group model if the* RPP$[\Xi_T]$ *and* RPSP$[\Xi_T]$ *problems are hard for* H.

Proof. Let $\mathcal{A}$ be a (Γ-time, q-query) adversary that can break the security of Π_{Sc}. Hence $\mathcal{A}$ can make totally at most q group oracle, pairing oracle and signing oracle queries, and runs in time at most Γ. Let $q_{\mathcal{O}}$ denote the total number of calls to the oracles $\mathcal{O}$, $\mathcal{O}_T$ and $\mathcal{O}_e$, and q_Ω denote the number of calls to the signing oracle Ω_{Sc}. Thus $q_{\mathcal{O}} + q_\Omega \leq q$.

$\Pr^{forge}_{\mathcal{A},\Pi_{\mathsf{Sc}}}$ denote the advantage of the adversary $\mathcal{A}$ in computing a forgery against Π_{Sc}. Also let RPP$[\Xi_T]$ and RPSP$[\Xi_T]$ problems be $(\Gamma', \epsilon_{\text{RPP}})$- and $(\Gamma', \epsilon_{\text{RPSP}})$-hard for the hash function H. We show that

$$\Pr^{forge}_{\mathcal{A},\Pi_{\mathsf{Sc}}} \leq 2q \cdot \epsilon_{\text{RPSP}} + 36q^2 \cdot \epsilon_{\text{RPP}} + \frac{15q^2}{p} + \frac{108q^3}{p}$$

for any (Γ-time, q-query) adversary $\mathcal{A}$ in the GBG model, where $\Gamma' \approx \Gamma$. More precisely, Γ' is the sum of Γ and the time required by simulator to maintain the environment.

The main idea is to use $\mathcal{A}$ to construct an adversary $\mathcal{B}$ that solves both the RPP[Ξ_T] and the RPSP[Ξ_T] problems for the hash function H. $\mathcal{B}$ will simulate EUF-CMA experiment for $\mathcal{A}$ in the naive way, through the game $\mathcal{G}$ that we later describe below. In the game, $\mathcal{B}$ also simulates the generic bilinear group oracles in the usual way by maintaining lists of pairs of encodings and polynomials that represent the relation amongst group elements. Let $\mathcal{C}$ be a challenger trying to prove the hardness of both the RPP[Ξ_T] and the RPSP[Ξ_T] problems for H against $\mathcal{B}$.

There are only two possibilities for $\mathcal{A}$ to output a forgery:

1. $\mathcal{A}$ uses a signature previously obtained to output a forgery (on a distinct message).
2. $\mathcal{A}$ does *not* output a previously obtained signature as a forgery.

Note that in a forgery of type 1, the "random prefix" for the hash function input (during verification) is the same as that for the corresponding previously obtained signature. In this case $\mathcal{B}$ will attempt to solve the RPSP[Ξ_T] problem for H. There are two issues that $\mathcal{B}$ needs to address in this case to solve the RPSP problem. First, it must correctly guess at the beginning of the simulation when $\mathcal{A}$ outputs a forgery of type 1 (and accordingly inform $\mathcal{C}$ that it attempts to solve the RPSP problem). Secondly, $\mathcal{B}$ needs to guess a priori which one of the previous signatures will $\mathcal{A}$ use for the forgery. Then during that step $\mathcal{B}$ needs to forward the corresponding message to the (now RPSP) challenger $\mathcal{C}$ to obtain a random prefix as part of its RPSP challenge. This random prefix will be used as the encoding of the corresponding element g_T^t during the above signing step. $\mathcal{B}$ solves the RPSP problem by forwarding to its (now RPSP) challenger $\mathcal{C}$ the forged message output by $\mathcal{A}$. Note that the probability that $\mathcal{B}$ will succeed in both the guesses is at least $\frac{1}{2q}$.

In the case of forgery of type 2, $\mathcal{B}$ will attempt to solve the RPP[Ξ_T] problem for H. Again, $\mathcal{B}$ must first guess correctly when this type of forgery occurs (and accordingly inform $\mathcal{C}$ that it attempts to solve the RPP problem). Secondly, $\mathcal{B}$ must commit to a value γ to obtain a random prefix $R \in \Xi_T$ as part of a RPP challenge. Eventually when $\mathcal{A}$ outputs a forgery on a (distinct) message m, it must turn out that the encoding of the "corresponding g_T^t" must be R and that $\mathsf{H}(R||m) = \gamma$. The tricky question is how to commit to the value γ before seeing R and m? We overcome this problem by assuming that $\mathcal{A}$ executes the verification algorithm $\mathsf{Verify}_{\mathsf{Sc}}(\cdot)$ before outputting its forgery (Y, γ), as done in [29]. This is w.l.o.g. because for every adversary that does not verify its forgery, we can build an adversary that has the same advantage but verifies its attempted forgery. This step guarantees that the elements $Y \in \mathbb{G}$ and $g_T^t \in \mathbb{G}_T$ appears as outputs of group oracles, with Y appearing before g_T^t. We bound the probability that Y appears later than g_T^t to be ϵ_{RPP}.

Hence $\mathcal{B}$ simply needs to guess a priori which group oracle query outputs g_T^t. During this step, $\mathcal{B}$ recovers the value of γ using the coefficients of the polynomials representing Y and g_T^t, as explained in (7) and proved in Lemma 3. Note that $\mathcal{B}$ also needs to guess a priori which element will be output as Y. $\mathcal{B}$ forwards the value of γ to the (now RPP) challenger $\mathcal{C}$ and obtains the random prefix R, which it uses as the encoding of g_T^t. Note that the probability that $\mathcal{B}$ will succeed in all the three guesses is at least $\frac{1}{2(3q)^2}$, where we later show that the number of elements to choose from is at most $3q$ in both the cases. We would like to note that recovering γ in the proof of [29] (for the original Schnorr signature scheme) is easier than in the bilinear setting. This is because in [29] it involved guessing an element in only one list and the polynomials involved are all binomials.

We now formally describe the game $\mathcal{G}$. The description of the group oracles is typical for proofs in the generic group model (see [36,28,5,18]).

Description of Game $\mathcal{G}$: Initially, $\mathcal{B}$ will choose a random bit $\beta_{\mathcal{C}} \overset{\$}{\leftarrow} \{0,1\}$. This bit decides which of the two problems RPSP (if $\beta_{\mathcal{C}} = 0$) or RPP (if $\beta_{\mathcal{C}} = 1$) will $\mathcal{B}$ attempt to solve using the forgery output by $\mathcal{A}$. If $\beta_{\mathcal{C}} = 0$, then $\mathcal{B}$ randomly chooses $i^* \overset{\$}{\leftarrow} \{1, \ldots, q\}$, else it randomly chooses $i^*, j^* \overset{\$}{\leftarrow} \{1, \ldots, 3q\}$. The quantity i^* indicates the step in which $\mathcal{B}$ interacts with $\mathcal{C}$ to obtain a random prefix $\xi_{T,i^*} \in \Xi_T$. This step may be a signature query (if $\beta_{\mathcal{C}} = 0$) or a group oracle query to $\mathcal{O}_T$ (if $\beta_{\mathcal{C}} = 1$). More on this will be discussed later when describing the simulation of signature queries and queries to the group oracle $\mathcal{O}_T$.

Let X, $\{\mathsf{T}_i : i \geq 1\}$, $\{\mathsf{U}_i : i \geq 1\}$ and $\{\mathsf{V}_i : i \geq 1\}$ be indeterminates, and $\{m_i : i \geq 1\}$ be bit-strings (messages) that are chosen by $\mathcal{A}$. Intuitively, these (or other) polynomials represent the relation amongst the group elements that are output by a group oracle, or guessed by $\mathcal{A}$. The indeterminate X corresponds to the quantity x (discrete logarithm of the secret key), whereas T_i corresponds to the parameter t_i chosen in the i^{th} signing step ($1 \leq i \leq q_\Omega$). Since $\mathcal{A}$ can query the group oracles with representations (from Ξ and Ξ_T) not previously obtained from the group oracles, in order to accommodate this case, we introduce the indeterminates U_i, V_i. The U_i correspond to the guessed elements of $\mathbb{G}$, whereas V_i correspond to the guessed elements of $\mathbb{G}_T$. We denote the lists $\{\mathsf{T}_i : i \geq 1\}$, $\{\mathsf{U}_i : i \geq 1\}$ and $\{\mathsf{V}_i : i \geq 1\}$ by $\{\mathsf{T}\}$, $\{\mathsf{U}\}$ and $\{\mathsf{V}\}$, respectively.

$\mathcal{B}$ maintains three lists of pairs

$$\mathcal{L} = \{(\mathsf{F}_{1,i}\,,\, \xi_{1,i}) \,:\, 1 \leq i \leq \tau_1\}, \tag{3}$$
$$\mathcal{L}_T = \{(\mathsf{F}_{T,i}\,,\, \xi_{T,i}) \,:\, 1 \leq i \leq \tau_T\}, \tag{4}$$
$$\mathcal{L}_\Omega = \{(m_i\,,\, \xi_{\Omega,i}\,,\, \gamma_i) \,:\, 1 \leq i \leq \tau_\Omega\}. \tag{5}$$

The entries $\mathsf{F}_{1,i} \in \mathbb{Z}_p[\mathsf{X}, \{\mathsf{U}\}, \{\mathsf{T}\}]$, $\mathsf{F}_{T,i} \in \mathbb{Z}_p[\mathsf{X}, \{\mathsf{U}\}, \{\mathsf{V}\}, \{\mathsf{T}\}]$ are multivariate polynomials over $\mathbb{Z}_p$, whereas $\xi_{1,i}$, $\xi_{\Omega,i}$, and $\xi_{T,i}$ are bit-strings in the encoding sets Ξ (of $\mathbb{G}$), Ξ, and Ξ_T (of $\mathbb{G}_T$), respectively. We have $m_i \in \{0,1\}^*$ and $\gamma_i \in \mathbb{Z}_p$. The polynomials in lists $\mathcal{L}$ and $\mathcal{L}_T$ correspond to elements of $\mathbb{G}$ and $\mathbb{G}_T$, respectively, that $\mathcal{A}$ will ever be able to compute or guess. The list $\mathcal{L}_\Omega$ records the signatures obtained by $\mathcal{A}$ on the messages m_i of its choice. The values τ_1, τ_T and τ_Ω denote the respective list counters.

Initially, $\tau_1 = 1$, $\tau_T = 1$, $\tau_\Omega = 0$, $\mathcal{L} = \{\ (1, \xi_{1,1})\ \}$, $\mathcal{L}_T = \{\ (\mathsf{X}, \xi_{T,1})\ \}$, and $\mathcal{L}_\Omega = \{\}$. The bit-strings $\xi_{1,1}$, $\xi_{T,1}$ are set to random distinct strings from Ξ and Ξ_T, respectively. We assume that there is some ordering among the strings in the sets Ξ and Ξ_T (say, lexicographic ordering), so that given a string $\xi_{1,i}$ or $\xi_{T,i}$, it is possible to efficiently determine its index in the lists, if it exists. The initial state of the lists $\mathcal{L}$ and $\mathcal{L}_T$ correspond to the generator of $\mathbb{G}$ and the public key, respectively.

The game begins by $\mathcal{B}$ providing $\mathcal{A}$ with the string $\xi_{1,1}$ from $\mathcal{L}$, and the string $\xi_{T,1}$ from $\mathcal{L}_T$.

Signature Query: Signature queries by $\mathcal{A}$ are modeled as follows. $\mathcal{A}$ provides a message $m_{\tau_\Omega} \in \{0,1\}^*$ of its choice to $\mathcal{B}$. In response $\mathcal{B}$ first increments the counters $\tau_1 := \tau_1 + 1$, $\tau_T := \tau_T + 1$ and $\tau_\Omega := \tau_\Omega + 1$, and sets $\mathsf{F}_{T,\tau_T} := \mathsf{T}_{\tau_\Omega}$.

- **(RPSP Challenge)** If $\beta_\mathcal{C} = 0$ and $i^* = \tau_\Omega$, then $\mathcal{B}$ passes on m_{τ_Ω} to $\mathcal{C}$ to obtain a random prefix $\xi_{T,i^*} \in \Xi_T$ as part of an RPSP challenge. If ξ_{T,i^*} is already present in $\mathcal{L}_T$, then $\mathcal{B}$ completes the RPSP challenge with $\mathcal{C}$ by returning arbitrary values, after $\mathcal{A}$ terminates. Denote this event by Abort. Else $\mathcal{B}$ sets $\xi_{T,\tau_T} := \xi_{T,i^*}$.
- Else if $\beta_\mathcal{C} \neq 0$ or $i^* \neq \tau_\Omega$, then $\mathcal{B}$ sets ξ_{T,τ_T} to a random string distinct from those already present in $\mathcal{L}_T$.

Append $\mathcal{L}_T$ with $(\mathsf{F}_{T,\tau_T}, \xi_{T,\tau_T})$. $\mathcal{B}$ computes $\gamma_{\tau_\Omega} := \mathsf{H}(\xi_{T,\tau_T} || m_{\tau_\Omega})$, sets $\mathsf{F}_{1,\tau_1} := \mathsf{T}_{\tau_\Omega} + \gamma_{\tau_\Omega}\mathsf{X}$, sets ξ_{1,τ_1} to a random distinct string, appends $\mathcal{L}$ with $(\mathsf{F}_{1,\tau_1}, \xi_{1,\tau_1})$, sets $\xi_{\Omega,\tau_\Omega} := \xi_{1,\tau_1}$, and appends $\mathcal{L}_\Omega$ and provides $\mathcal{A}$ with $(m_{\tau_\Omega}\ ,\ \xi_{\Omega,\tau_\Omega}\ ,\ \gamma_{\tau_\Omega})$.

Group Operation of $\mathbb{G}$: The calls made by $\mathcal{A}$ to the group oracle $\mathcal{O}$ are modeled as follows. For group operations in $\mathbb{G}$, $\mathcal{A}$ provides $\mathcal{B}$ with two operands (bit-strings) $\xi_{1,i}, \xi_{1,j}$ $(1 \leq i, j \leq \tau_1)$ in $\mathcal{L}$ and also specifies whether to multiply or divide them. $\mathcal{B}$ answers the query by first incrementing the counter $\tau_1 := \tau_1 + 1$, and computes the polynomial $\mathsf{F}_{1,\tau_1} := F_{1,i} \pm F_{1,j}$. If $\mathsf{F}_{1,\tau_1} = \mathsf{F}_{1,k}$ for some $k < \tau_1$, then $\mathcal{B}$ sets $\xi_{1,\tau_1} := \xi_{1,k}$. Otherwise, ξ_{1,τ_1} is set to a random string distinct from those already present in $\mathcal{L}$. The pair $(\mathsf{F}_{1,\tau_1}, \xi_{1,\tau_1})$ is appended to $\mathcal{L}$ and $\mathcal{B}$ provides $\mathcal{A}$ with ξ_{1,τ_1}. Note that the (total) degree of the polynomials $\mathsf{F}_{1,i}$ in $\mathcal{L}$ is at most one.

If $\mathcal{A}$ queries $\mathcal{O}$ with an encoding ξ not previously output by the oracle, then $\mathcal{A}$ increments the counter $\tau_1 := \tau_1 + 1$, sets $\xi_{1,\tau_1} := \xi$, and sets $\mathsf{F}_{1,\tau_1} := \mathsf{U}_{\tau_1}$. The pair $(\mathsf{F}_{1,\tau_1}, \xi_{1,\tau_1})$ is appended to $\mathcal{L}$. This step is carried out for each guessed operand.

Group Operation of $\mathbb{G}_T$: The group oracle $\mathcal{O}_T$ is modeled similar to $\mathcal{O}$, instead appropriately updating the counter τ_T, and appending the list $\mathcal{L}_T$ with the output $(\mathsf{F}_{T,\tau_T}, \xi_{T,\tau_T})$. $\mathcal{B}$ provides $\mathcal{A}$ with ξ_{T,τ_T}. For guessed operands in $\mathbb{G}_T$, a new variable T_{τ_T} is introduced instead.

Pairing Operation: For a pairing operation, $\mathcal{A}$ queries $\mathcal{B}$ with two operands $\xi_{1,i}, \xi_{1,j}$ $(1 \leq i, j \leq \tau_1)$ in $\mathcal{L}$. $\mathcal{B}$ first increments $\tau_T := \tau_T + 1$, and then computes the polynomial $\mathsf{F}_{T,\tau_T} := \mathsf{F}_{1,i} \cdot \mathsf{F}_{1,j}$. Again, if $\mathsf{F}_{T,\tau_T} = \mathsf{F}_{T,k}$ for some $k < \tau_T$, then $\mathcal{B}$ sets $\xi_{T,\tau_T} := \xi_{T,k}$. Otherwise, ξ_{T,τ_T} is set to a random string distinct from those already present in $\mathcal{L}_T$. The pair $(\mathsf{F}_{T,\tau_T}, \xi_{T,\tau_T})$ is appended to $\mathcal{L}_T$, and $\mathcal{B}$

provides $\mathcal{A}$ with ξ_{T,τ_T}. Note that the degree of the polynomials $\mathsf{F}_{T,i}$ in $\mathcal{L}_T$ is at most two.

RPP Challenge: Recollect that $\mathcal{B}$ has earlier sampled $i^*, j^* \xleftarrow{\$} \{1,\ldots,3q\}$. Since $\mathcal{A}$ makes at most $q_{\mathcal{O}} < q$ group oracle queries and that in each query $\mathcal{A}$ can guess at most two new elements, it is easy to see that lists $\mathcal{L}$ and $\mathcal{L}_T$ together have at most $3(q_{\mathcal{O}} + q_{\Omega}) \leq 3q$ elements. Hence

$$|\mathcal{L}| + |\mathcal{L}_T| \leq 3q. \tag{6}$$

If $\beta_{\mathcal{C}} = 1$, then during each of the queries above the counter τ_T is checked while adding an element to the list $\mathcal{L}_T$. If $i^* = \tau_T$, then $\mathcal{B}$ computes

$$\gamma^* = \sum_{i=1}^{q_\Omega} a_i \gamma_i \ - \ a_{\mathsf{X}}, \tag{7}$$

where a_{X} is the coefficient of X in F_{T,i^*}, a_i is the coefficient of T_i in F_{1,j^*} $(1 \leq i \leq q_\Omega)$, and γ_i is, as defined previously, the hash value in the i^{th} signature query.

If F_{1,j^*} does not exist, or $i^* > \tau_T$ at the end of the game $\mathcal{G}$, then $\mathcal{B}$ completes the RPP challenge with $\mathcal{C}$ by returning arbitrary values. Else, $\mathcal{B}$ passes $\gamma^* \in \mathbb{Z}_p$ to $\mathcal{C}$ to obtain a random prefix $\xi_{T,i^*} \in \Xi_T$, as part of an RPP challenge. If $\mathsf{F}_{T,\tau_T} = \mathsf{F}_{T,k}$ for some $k < \tau_T$ and $\xi_{T,i^*} \neq \xi_{T,k}$, then $\mathcal{B}$ completes the RPP challenge with $\mathcal{C}$ by returning arbitrary values (Abort). Else, if there is no such k but ξ_{T,i^*} is already present in $\mathcal{L}_T$, then also Abort. Else $\mathcal{B}$ sets $\xi_{T,\tau_T} := \xi_{T,i^*}$.

If $\mathcal{B}$ has made right guesses for i^* and j^*, then F_{1,j^*} and F_{T,i^*} corresponds to the forgery and satisfy

$$\mathsf{F}_{T,i^*} := \mathsf{F}_{1,j^*} - \gamma\mathsf{X}, \tag{8}$$

where γ is the hash value corresponding to the forgery. Note again that both the polynomials exist (in case of successful forgery) because we assume that $\mathcal{A}$ always verifies its attempted forgery before it is output. Lemma 3 below proves that indeed $\gamma^* = \gamma$. Because $\mathcal{A}$ has access to the oracle $\mathcal{O}_T$, it is easy to see that it is not possible to recover γ from F_{T,i^*} alone.

Lemma 3. *Let $\mathsf{F}_{T,i^*} = \mathsf{F}_{1,j^*} - \gamma\mathsf{X}$, as computed in (8). Let a_X be the coefficient of X in F_{T,i^*}, and a_i be the coefficient of T_i in F_{1,j^*} $(1 \leq i \leq q_\Omega)$. Also let γ_i be the hash value in the i^{th} signature query. Then $\gamma = \sum_{i=1}^{q_\Omega} a_i\gamma_i \ - \ a_{\mathsf{X}}$.*

Proof. Any polynomial in $\mathcal{L}$, in particular F_{1,j^*}, is of the form $\mathsf{F}_{1,j^*} = c_1 + \sum_{i=1} c_{2,i}\mathsf{U}_i + \sum_{i=1}^{q_\Omega} a_i(\mathsf{T}_i + \gamma_i\mathsf{X})$, where $c_1, c_{2,i}, a_i \in \mathbb{Z}_p$ are chosen by $\mathcal{A}$. Hence the lemma follows. □

End of Game $\mathcal{G}$: When $\mathcal{A}$ terminates it outputs $(m, (\xi_{1,\alpha}, \gamma)) \in \{0,1\}^* \times \Xi \times \mathbb{Z}_p$, where $\xi_{1,\alpha} \in \mathcal{L}$ and $1 \leq \alpha \leq \tau_1$. This corresponds to the "forgery" output by $\mathcal{A}$ in the actual interaction. $\mathcal{B}$ simply forwards m to its challenger $\mathcal{C}$.

Let Forge denote the event of successful forgery. Next, $\mathcal{B}$ chooses random values x, $\{u\}$, $\{v\}$, $\{t\} \leftarrow \mathbb{Z}_p$ for the indeterminates X, $\{\mathsf{U}\}$, $\{\mathsf{V}\}$, $\{\mathsf{T}\}$, respectively. Then it evaluates the polynomials in lists $\mathcal{L}$ and $\mathcal{L}_T$. $\mathcal{B}$ will abort if:

1. $\mathsf{F}_{1,i}(x,\{u\},\{t\}) = \mathsf{F}_{1,j}(x,\{u\},\{t\})$ in $\mathbb{Z}_p$, for any $\mathsf{F}_{1,i} \neq \mathsf{F}_{1,j}$ in $\mathcal{L}$.
2. $\mathsf{F}_{T,i}(x,\{u\},\{v\},\{t\}) = \mathsf{F}_{T,j}(x,\{u\},\{v\},\{t\})$ in $\mathbb{Z}_p$, for any $\mathsf{F}_{T,i} \neq \mathsf{F}_{T,j}$ in $\mathcal{L}_T$.

Let $\mathsf{Collide}$ denote either of the above events, i.e. a *collision* occurring in lists $\mathcal{L}$ and/or $\mathcal{L}_T$. This completes the description of game $\mathcal{G}$ and simulator $\mathcal{B}$.

Analysis of $\mathbf{Pr}^{forge}_{\mathcal{A},\Pi_{\mathsf{Sc}}}$: The success probability $\Pr^{forge}_{\mathcal{A},\Pi_{\mathsf{Sc}}}$ of $\mathcal{A}$ in the actual EUF-CMA game satisfies

$$\Pr^{forge}_{\mathcal{A},\Pi_{\mathsf{Sc}}} \leq \Pr[\mathsf{Forge} \mid \overline{\mathsf{Collide}}] + \Pr[\mathsf{Collide}]. \tag{9}$$

This is because the event $\overline{\mathsf{Collide}}$ ensures that $\mathcal{A}$ will get to see only distinct group elements in the actual interaction. In other words, $\mathcal{A}$ is unable to cause *collisions* among group elements. As long as the event $\mathsf{Collide}$ does not occur, then the view of $\mathcal{A}$ is identical in the game $\mathcal{G}$ and the actual interaction. Hence if $\mathcal{A}$ is unable to provoke collisions, then adaptive strategies are no more powerful than non-adaptive ones (see [28, Lemma 2 on pp. 12], also [36]). This observation allows us to choose group elements and their representations independently of the strategy of $\mathcal{A}$.

First we bound $\Pr[\mathsf{Collide}]$. The τ_1 polynomials $\mathsf{F}_{1,i}$ in $\mathcal{L}$ have degree at most one. Note that $\mathsf{F}_{1,i} \neq \mathsf{F}_{1,j} \Leftrightarrow \mathsf{F}_{1,i} - \mathsf{F}_{1,j} \neq 0$ as polynomials. From Lemma 2 (with $\lambda' = 0$), the probability that two distinct polynomials in $\mathcal{L}$ evaluate to the same value for randomly and independently chosen values for the indeterminates is at most $\frac{1}{p}$. Summing up over at most $\binom{\tau_1}{2}$ distinct pairs (i,j), the probability that the condition 1 above holds is at most $\binom{\tau_1}{2} \cdot \frac{2}{p}$. Similarly, the probability that the condition 2 above holds is at most $\binom{\tau_T}{2} \cdot \frac{2}{p}$. Using (6) we obtain

$$\Pr[\mathsf{Collide}] \leq \binom{\tau_1}{2} \cdot \frac{1}{p} + \binom{\tau_T}{2} \cdot \frac{2}{p} \leq \frac{1}{p}(\tau_1 + \tau_T)^2 \leq \frac{9q^2}{p}. \tag{10}$$

Next we bound $\Pr[\mathsf{Forge} \mid \overline{\mathsf{Collide}}]$ in terms of the advantage of $\mathcal{B}$ against $\mathcal{C}$. Whenever $\mathcal{A}$ succeeds in outputting a forgery $(m, (\xi_{1,\alpha}, \gamma))$, there are only two possibilities that can arise:

- **(Solving RPSP Challenge)** There exists an i $(1 \leq i \leq q_\Omega)$ such that $(m_i, (\xi_{1,\alpha}, \gamma)) \in \mathcal{L}_\Omega$. In other words, $\mathcal{A}$ uses a signature previously obtained to output its forgery on a distinct message. Let Forge_1 denote this event. If $\beta_{\mathcal{C}} = 0$ and $i^* = i$, then $\mathcal{B}$ can successfully use the forgery to solve the RPSP$[\Xi_T]$ problem for H. This is because $\mathcal{B}$ will attempt to solve the RPSP problem only when $\beta_{\mathcal{C}} = 0$, the probability of which is $\frac{1}{2}$. Since at the beginning itself $\mathcal{B}$ will decide at which signing step (step i^*) it will interact with $\mathcal{C}$ when $\beta_{\mathcal{C}} = 0$, the probability that $i^* = i$ is at least $\frac{1}{q_\Omega} > \frac{1}{q}$. Hence the advantage of $\mathcal{B}$ in solving RPSP problem is at least $\frac{1}{2q}\Pr[\mathsf{Forge}_1 \mid \overline{\mathsf{Collide}}] - \left(\frac{3q}{p}\right)$, where $\left(\frac{3q}{p}\right)$ is an upper bound on the probability that $\mathcal{B}$ does not Abort due to a repeated entry in $\mathcal{L}_T$ during RPSP challenge step. It may be noted that if $\mathcal{B}$ attempts to solve the RPP problem using this type of forgery, then Abort will occur with overwhelming probability. Therefore, $\Pr[\mathsf{Forge}_1 \,|\, \overline{\mathsf{Collide}}] \leq 2q \cdot \epsilon_{\mathrm{RPSP}} + \frac{6q^2}{p}$.

- **(Solving RPP Challenge)** The complementary event of Forge_1, $\overline{\mathsf{Forge}_1}$. That is, $\mathcal{A}$ does not use a signature previously obtained to output its forgery. Since $\mathcal{A}$ verifies its forgery before it is output, then there exists some i^{th} entry $(\mathsf{F}_{T,i}, \xi_{T,i})$ in the list $\mathcal{L}_T$ such that $\mathsf{H}(\xi_{T,i}||m) = \gamma$. Also let this entry be the first occurrence of this pair in $\mathcal{L}_T$. If $\beta_{\mathcal{C}} = 1$, $i^* = i$ and $j^* = \alpha$, then $\mathcal{B}$ can successfully use the forgery to solve the RPP$[\Xi_T]$ problem for H. Hence the advantage of $\mathcal{B}$ in solving the RPP problem is at least $\frac{1}{2(3q)^2}\Pr[\overline{\mathsf{Forge}_1} \,|\, \overline{\mathsf{Collide}}] - \left(\frac{3q}{p} + \epsilon_{\mathrm{RPP}}\right)$, where again $\left(\frac{3q}{p}\right)$ is an upper bound on the probability that $\mathcal{B}$ does not Abort due to a repeated entry in $\mathcal{L}_T$ during RPP challenge step.
  The quantity ϵ_{RPP} appearing above is an upper bound on the probability that the entry $(\mathsf{F}_{T,i}, \xi_{T,i})$ does not appear before $(\mathsf{F}_{1,\alpha}, \xi_{1,\alpha})$. Because $\mathsf{F}_{T,i} = \mathsf{F}_{1,\alpha} - \gamma\mathsf{X}$ (c.f. (8)) and that encodings are random, this means that $\mathcal{A}$ is able to compute the value γ even before getting an encoding $\xi_{T,i}$ such that $\mathsf{H}(\xi_{T,i}||m) = \gamma$. In other words, $\mathcal{A}$ has solved the RPP$[\Xi_T]$ problem for H. Therefore, $\Pr[\overline{\mathsf{Forge}_1} \,|\, \overline{\mathsf{Collide}}] \leq 36q^2 \cdot \epsilon_{\mathrm{RPP}} + \frac{108q^3}{p}$.

Since $\Pr[\mathsf{Forge} \,|\, \overline{\mathsf{Collide}}] = \Pr[\mathsf{Forge}_1 \,|\, \overline{\mathsf{Collide}}] + \Pr[\overline{\mathsf{Forge}_1} \,|\, \overline{\mathsf{Collide}}]$, we obtain

$$\Pr[\mathsf{Forge} \,|\, \overline{\mathsf{Collide}}] \leq 2q \cdot \epsilon_{\mathrm{RPSP}} + 36q^2 \cdot \epsilon_{\mathrm{RPP}} + \frac{6q^2}{p} + \frac{108q^3}{p}. \tag{11}$$

From (9), (10) and (11), we have $\Pr^{forge}_{\mathcal{A},\Pi_{\mathsf{Sc}}} \leq 2q \cdot \epsilon_{\mathrm{RPSP}} + 36q^2 \cdot \epsilon_{\mathrm{RPP}} + \frac{15q^2}{p} + \frac{108q^3}{p}$. Hence if $q = poly(\log p)$, then $\Pr^{forge}_{\mathcal{A},\Pi_{\mathsf{Sc}}}$ is negligible provided $(\epsilon_{\mathrm{RPSP}} + \epsilon_{\mathrm{RPP}})$ is negligible. This completes the proof of Theorem 1. $\square$

4 A Leakage-Resilient Signature Scheme

In this section, we describe a leakage-resilient variant Π^*_{Sc} of the scheme Π_{Sc}. We use the techniques of [23] to transform Π_{Sc} to Π^*_{Sc}. A major difference between the two variants is that the secret key $X = g^x$ of Π_{Sc} is now split into two parts as $(S_0 := g^{l_0}, S'_0 := g^{x-l_0})$ for a random $l_0 \leftarrow \mathbb{Z}_p$. The two shares reside in different parts of the memory. The key generation step $\mathsf{KeyGen}^*_{\mathsf{Sc}}$ of Π^*_{Sc} is obtained by suitably modifying the $\mathsf{KeyGen}_{\mathsf{Sc}}$ step of Π_{Sc}. The signing step of Π^*_{Sc} is also split into two steps $\mathsf{Sign}^*_{\mathsf{Sc1}}$ and $\mathsf{Sign}^*_{\mathsf{Sc2}}$. After every signature query, the two shares of the secret key are randomly refreshed. This is required because, as seen in Section 2.1, if the secret state is not stateful, then the scheme cannot be secure in the presence of continual leakage.

Let $\mathsf{H} : \{0,1\}^* \to \mathbb{Z}_p$ be a hash function. The stateful signature scheme $\Pi^*_{\mathsf{Sc}} = (\mathsf{KeyGen}^*_{\mathsf{Sc}}, \mathsf{Sign}^*_{\mathsf{Sc1}}, \mathsf{Sign}^*_{\mathsf{Sc2}}, \mathsf{Verify}^*_{\mathsf{Sc}})$, defined on $\{0,1\}^*$, is as follows:

1. $\mathsf{KeyGen}^*_{\mathsf{Sc}}(\kappa)$: Compute $\mathbb{PP} := (\mathbb{G}, \mathbb{G}_T, p, e, g) \leftarrow \mathsf{BGen}(\kappa)$. Choose random $x, l_0 \leftarrow \mathbb{Z}_p$. Set $X := g^x$ and $X_T := e(g, X) = e(g, g)^x$. The public key is $pk := (\mathbb{PP}, X_T, \mathsf{H})$ and the secret key is $sk^* := (S_0 := g^{l_0}, S'_0 := g^{x-l_0} = X \cdot g^{-l_0}) \in \mathbb{G}^2$.
2. $\mathsf{Sign}^*_{\mathsf{Sc1}}(S_{i-1}, m_i)$: Choose random $t_i, l_i \leftarrow \mathbb{Z}_p$. Set $S_i := S_{i-1} \cdot g^{l_i}$, $\gamma_i := \mathsf{H}(g_T^{t_i} || m_i)$, and $Y'_i := g^{t_i} \cdot S_i^{\gamma_i}$.
3. $\mathsf{Sign}^*_{\mathsf{Sc2}}(S'_{i-1}, (Y'_i, \gamma_i, l_i))$: Set $S'_i := S'_{i-1} \cdot g^{-l_i}$, $Y_i := Y'_i \cdot (S'_i)^{\gamma_i}$, and $\sigma_i := (Y_i, \gamma_i)$. Output the signature σ_i. .
4. $\mathsf{Verify}^*_{\mathsf{Sc}}(pk, m, \sigma)$: Let $\sigma = (Y, \gamma) \in \mathbb{G} \times \mathbb{Z}_p$. Set $\rho := e(Y, g) \star (g_T^x)^{-\gamma}$. Output the bit $b = 1$ (*valid*) if $\mathsf{H}(\rho || m) = \gamma$. Otherwise output $b = 0$ (*invalid*).

The index i used above refers to the number of times the signing algorithm has been invoked. For $i \geq 1$, let $Z_i := \sum_{j=0}^{i} l_j$. The correctness property of Π^*_{Sc} follows from Π_{Sc} since $S_i \cdot S'_i = g^{Z_i} \cdot g^{x - Z_i} = X$. The leakage functions $f_i()$ and $h_i()$ that the adversary specifies to the signing oracle would take the form $f_i(S_{i-1}, (l_i, t_i))$ and $h_i(S'_{i-1}, (Y'_i, \gamma_i, l_i))$ (cf. (1) and (2)).

The signing step of Π^*_{Sc} requires totally six exponentiations - four for $\mathsf{Sign}^*_{\mathsf{Sc1}}$ and two for $\mathsf{Sign}^*_{\mathsf{Sc2}}$. This quantity can be reduced to five if g^{l_i} is also passed on from $\mathsf{Sign}^*_{\mathsf{Sc1}}$ to $\mathsf{Sign}^*_{\mathsf{Sc2}}$. Note that the $\mathsf{Sign}_{\mathsf{Sc}}$ step of Π_{Sc} requires only three exponentiations.

Since the input/output behaviour of Π^*_{Sc} and Π_{Sc} is identical, from Theorem 1 we obtain that Π^*_{Sc} is secure in the GBG model in a non-leakage setting.

Lemma 4. *The signature scheme* Π^*_{Sc} *is* EUF-CMA *secure in the generic bilinear group model if the* RPP$[\Xi_T]$ *and* RPSP$[\Xi_T]$ *problems are hard for* H.

The following theorem shows that Π^*_{Sc} is resilient to continual leakage in the GBG model if RPP$[\Xi_T]$ and RPSP$[\Xi_T]$ problems are hard for the hash function H, and $\lambda < \frac{\log p}{2} - \omega(\log \log p)$, where λ is the leakage parameter.

Theorem 2. *The signature scheme* Π^*_{Sc} *is secure with leakage w.r.t. Definition 2 in the generic bilinear group model relative to the hardness of RPP*$[\Xi_T]$ *and RPSP*$[\Xi_T]$ *problems for* H. *Let the RPP and RPSP problems be* $(\Gamma, \epsilon_{\mathrm{RPP}})$ *and* $(\Gamma, \epsilon_{\mathrm{RPSP}})$*-hard, respectively. Then the advantage of a (*Γ*-time, q-query) adversary who gets at most* λ *bits of leakage per each invocation of* $\mathsf{Sign}^*_{\mathsf{Sc1}}$ *or* $\mathsf{Sign}^*_{\mathsf{Sc2}}$ *is* $O\left(q^2\, \epsilon_{\mathrm{RPP}} + q\, \epsilon_{\mathrm{RPSP}} + \frac{q^3}{p} + \frac{q^2}{p} 2^{2\lambda}\right)$.

Let us briefly sketch the main ideas of the proof. Working on the lines of (9), the advantage of $\mathcal{A}$ is bounded by its success probabilities conditioned on the event whether or not a collision has occurred in the lists consisting of elements of $\mathbb{G}$ and $\mathbb{G}_T$. It is important to note that the proofs for the non-leakage setting (i.e. proof of Theorem 1) and the leakage setting would be the same conditioned on the fact that a collision has not occurred. The reason is that in the event of no collision, the adversary must either solve the RPP or the RPSP problem for the hash function in order to output a forgery (let us recall that a solution to either the RPP or the RPSP problem implies a collision for the hash function). Hence leakage on the secret state will not be useful in this case. Hence the success probability of $\mathcal{A}$ against Π_{Sc} and Π^*_{Sc} is the same in the event of no collision (that includes the event of guessing the representations of group elements using partial information about them).

However the probability that a collision occurs in the leakage setting is increased by a factor of at most $2^{2\lambda}$. This is because when $\mathcal{A}$ has access to leakage output $f_i(S_{i-1},(l_i,t_i))$ and $h_i(S'_{i-1},(Y'_i,\gamma_i,l_i))$ during i^{th} signature query, then in adversary's view the parameters t_i, l_i $(i \geq 1)$ are no longer uniformly distributed even though they are still independent. Hence $\mathcal{A}$ can now cause collisions among polynomials (in Conditions 1-2 on page 12) with increased probability. Each value t_i can only be leaked by f_i, hence at most λ bits of t_i can be leaked. Since l_i appears in both $f_i()$ and $h_i()$, at most 2λ bits of l_i can be leaked.

The only useful information that the leakage functions can provide to $\mathcal{A}$ is about the secret key X and the values t_i. This is because the values l_i are independent of the signatures generated. However $\mathcal{A}$ can use the leakages of l_i to eventually leak X. If $\mathcal{A}$ is able to compute X, then it can trivially forge a signature on a distinct message. The event of no collision, and the fact that X is not a "linear combination" of the inputs to the leakage functions, guarantees that $\mathcal{A}$ is unable to compute X.

Proof. Let $\mathcal{A}$ be a ($\mathit{\Gamma}$-time, q-query) adversary that can break the security of Π^*_{Sc}. Hence $\mathcal{A}$ can make totally at most q group oracle, pairing oracle and signing oracle queries, and runs in time at most $\mathit{\Gamma}$. In the count of q, even group oracle queries by leakage functions f_i, h_i $(i \geq 1)$ specified by $\mathcal{A}$ are also included. Let the adversary $\mathcal{A}$ play the game $\mathcal{G}'$ described below. This game is an extension of game $\mathcal{G}$ described in the proof of Theorem 1. To avoid repetition, we only describe here the extensions that are not part of game $\mathcal{G}$. Let $\{L\}$ denote the list of indeterminates $\{L_i : 1 \leq i \leq q_\Omega\}$ that correspond to the values l_i in Π^*_{Sc}.

Game $\mathcal{G}'$: For each leakage function $f_i(S_{i-1},(l_i,t_i))$ and $h_i(S'_{i-1},(Y'_i,\gamma_i,l_i))$, $\mathcal{A}$ maintains a pair of lists $\left(\mathcal{L}^{f_i}, \mathcal{L}^{f_i}_T\right)$ and $\left(\mathcal{L}^{h_i}, \mathcal{L}^{h_i}_T\right)$, respectively. These lists contain polynomial and bit-string pairs. The polynomials in $\mathcal{L}^{f_i}$ and $\mathcal{L}^{h_i}$ belong to $\mathbb{Z}_p[\mathsf{X},\{\mathsf{U}\},\{\mathsf{T}\},\{\mathsf{L}\}]$, and the corresponding bit-strings are from the encoding set $\mathit{\Xi}$ of group $\mathbb{G}$. The polynomials in $\mathcal{L}^{f_i}_T$ and $\mathcal{L}^{h_i}_T$ are in the ring $\mathbb{Z}_p[\mathsf{X},\{\mathsf{U}\},\{\mathsf{V}\},\{\mathsf{T}\},\{\mathsf{L}\}]$, and the corresponding bit-strings are from the encoding set $\mathit{\Xi}_T$ of group $\mathbb{G}_T$. Intuitively, the polynomials in lists $\mathcal{L}^{f_i}$ and $\mathcal{L}^{h_i}$ correspond to the elements of group $\mathbb{G}$ that can be computed by f_i and h_i, respectively, whereas the lists $\mathcal{L}^{f_i}_T$ and $\mathcal{L}^{h_i}_T$ correspond to the elements of $\mathbb{G}_T$.

Every polynomial in $\mathcal{L}^{f_i}$ is of the form $c_{1,i}\mathsf{L}_i + c_{2,i}\sum_{j=0}^{i-1}\mathsf{L}_j + c_{3,i}\mathsf{D}_i$, where $c_{1,i}$, $c_{2,i}$, $c_{3,i} \in \mathbb{Z}_p$ are chosen by $\mathcal{A}$ and $\mathsf{D}_i \in \mathbb{Z}_p[\mathsf{X},\{\mathsf{U}\},\{\mathsf{T}\}]$ is in $\mathcal{L}$ (cf. (3)). Every polynomial in $\mathcal{L}^{h_i}$ is of the form

$$d_{1,i}\mathsf{L}_i + d_{2,i}\left(\mathsf{X} - \sum_{j=0}^{i-1}\mathsf{L}_j\right) + d_{3,i}\left(\mathsf{T}_i + \gamma_i\left(\sum_{j=0}^{i}\mathsf{L}_j\right)\right) + d_{4,i}\mathsf{W}_i, \qquad (12)$$

where $d_{1,i}$, $d_{2,i}$, $d_{3,i}$, $d_{4,i} \in \mathbb{Z}_p$ are also chosen by $\mathcal{A}$ and $\mathsf{W}_i \in \mathbb{Z}_p[\mathsf{X}, \mathsf{X}_0, \mathsf{X}_1, \{\mathsf{U}\}, \{\mathsf{T}\}]$ is in the list $\mathcal{L}$. Note that the polynomials in lists $\mathcal{L}^{f_i}$ and $\mathcal{L}^{h_i}$ are of degree at most one, and that they do not contain the monomial X. The polynomials in lists $\mathcal{L}_T^{f_i}$ and $\mathcal{L}_T^{h_i}$ are of degree at most two.

The game $\mathcal{G}'$ proceeds exactly as game $\mathcal{G}$ except that $\mathcal{A}$ can also obtain leakage through functions f_i and h_i in the i^{th} signature query. In particular, when $\mathcal{A}$ terminates it outputs $(m, (\xi_{1,\alpha}, \gamma)) \in \{0,1\}^* \times \varXi \times \mathbb{Z}_p$, where $\xi_{1,\alpha} \in \mathcal{L}$ and $1 \le \alpha \le \tau_1$. Let us denote by Forge^* the event of successful forgery by $\mathcal{A}$. Let $\mathsf{Collide}^*$ denote the event of a collision occurring in lists $\mathcal{L}$, $\mathcal{L}_T$, $\mathcal{L}^{f_i}$, $\mathcal{L}^{h_i}$, $\mathcal{L}_T^{f_i}$, $\mathcal{L}_T^{h_i}$ $(1 \le i \le q_\Omega)$. The polynomials are now evaluated with values chosen from independent distributions with min-entropy $\log p - 2\lambda$, not necessarily from an uniform distribution. The exact distribution depends on the leakage functions chosen by $\mathcal{A}$. Since we are only interested to upper bound the collision probability, we can safely assume that the simulator chooses the right distribution. Note that even in the leakage setting, adaptive strategies are no more powerful than non-adaptive ones, as observed in [2, pp. 691]. This completes the description of the game $\mathcal{G}'$.

Let $\Pr^{forge}_{\mathcal{A},\mathsf{\Pi}^*_{\mathsf{Sc}}}$ denote the advantage of $\mathcal{A}$ in computing a forgery against $\mathsf{\Pi}^*_{\mathsf{Sc}}$. On the lines of (9), we can write

$$\Pr^{forge}_{\mathcal{A},\mathsf{\Pi}^*_{\mathsf{Sc}}} \;\le\; \Pr[\mathsf{Forge}^* \mid \overline{\mathsf{Collide}^*}] + \Pr[\mathsf{Collide}^*]. \tag{13}$$

As mentioned before, conditioned on the event $\overline{\mathsf{Collide}^*}$, the view of the adversary $\mathcal{A}$ will be same in both the games $\mathcal{G}'$ and $\mathcal{G}$. This is because in both the cases $\mathcal{A}$ will get to see only distinct group elements. Hence, from (11), we have

$$\Pr[\mathsf{Forge}^* \mid \overline{\mathsf{Collide}^*}] \;\le\; O\left(q^2\,\epsilon_{\text{RPP}} + q\,\epsilon_{\text{RPSP}} + \frac{q^3}{p}\right). \tag{14}$$

Lemma 5. $\Pr[\mathsf{Collide}^*] \le O\left(\frac{q^2}{p} 2^{2\lambda}\right)$.

Proof. To compute the required probability, the polynomials in lists $\mathcal{L}$, $\mathcal{L}_T$, $\mathcal{L}^{f_i}$, $\mathcal{L}^{h_i}$, $\mathcal{L}_T^{f_i}$, $\mathcal{L}_T^{h_i}$ $(1 \le i \le q_\Omega)$ are evaluated by choosing values from $\mathbb{Z}_p$ according to (independent) distributions with min-entropy at least $\log p - 2\lambda$. This is because $\mathcal{A}$ can obtain at most 2λ bits of leakage about l_i $(i = 0, \ldots, q_\Omega)$, and at most λ bits of t_i $(i = 1, \ldots, q_\Omega)$. According to Lemma 1, the values l_i, t_i have min-entropy at least $\log p - 2\lambda$ in the view of $\mathcal{A}$. The total length of all the lists is at most $O(q_\Omega + q_\mathcal{O}) = O(q)$. Hence there can be at most $O(q^2)$ pairs of distinct polynomials (of degree at most two) evaluating to the same value. From Lemma 2 (with $\lambda' = 2\lambda$), we obtain $\Pr[\mathsf{Collide}^*] \le O\left(\frac{q^2}{p} 2^{2\lambda}\right)$. $\square$

From (13), (14) and Lemma 5, we have $\Pr^{forge}_{\mathcal{A},\mathsf{\Pi}^*_{\mathsf{Sc}}} \le O\Big(q^2\,\epsilon_{\text{RPP}} + q\,\epsilon_{\text{RPSP}} + \frac{q^3}{p} + \frac{q^2}{p} 2^{2\lambda}\Big)$. This completes the proof of Theorem 2. $\square$

5 Conclusions

In this work we presented the pairing-based split Schnorr scheme and quantified its security against independent and continual leakage in the generic bilinear group model. In particular, we showed that allowing λ bits of leakage at each of the two phases of every round in the proposed scheme can be compared to decreasing the security of the pairing-based Schnorr scheme (without leakage) by a factor of at most $2^{2\lambda}$ in our leakage model.

Undoubtedly, the main advantage of our approach lies on its practicality: signing takes at most 5 exponentiations in $\mathbb{G}$ plus 1 exponentiation in $\mathbb{G}_T$; verification takes 1 pairing plus 1 exponentiation in $\mathbb{G}_T$. A suitable bilinear pairing group to implement our modification of the Schnorr scheme is the pairing-friendly curve BN-128 studied by Scott in [35]. Thus while our scheme offers continual leakage-resilience, its efficiency is comparable to that of standard pairing-based signature schemes [37]. This is currently out of reach for schemes that offer EUF-CMA security against continual leakage and dispense with the generic group model, be it in the standard or the random oracle models.

It is interesting to compare the relative efficiency and strength of our scheme and the FKPR scheme by Faust *et al.* [16]. The latter has a weak form of EUF-CMA security against continual independent leakages in the random oracle model, where the adversary can ask at most for D signatures queries, for D fixed before the key generation phase. The main advantage of that construction with respect to ours is that it can be implemented over any group $\mathbf{G}$ where the DL problem is conjectured to be hard (our scheme needs pairing-based groups). Let us now examine its disadvantages against our scheme, which are all related to its practicality. The signer in the FKPR scheme needs to maintain a state consisting on roughly d Schnorr signatures and d public and corresponding secret keys, with the length of a signature being proportional to d and $D = 2^{d+1} - 2$; signing takes 9 exponentiations in the group $\mathbf{G}$, while verification time is proportional to d. FPKR only tolerates a leakage rate of roughly 1/36. Thus, for reasonable values of d, e.g. $d = 20$, our scheme is more efficient in storage, computing time and leakage ratio than the FPKR scheme, while offering standard existential unforgeability against continual leakage in the split-state model. Finally both our scheme and the FPKR scheme use an idealized model of computation to prove security, namely the former uses the random oracle model, while ours uses generic groups.

Acknowledgements. The research leading to these results has received funding from the European Research Council under the European Union's Seventh Framework Programme (FP7/2007-2013) / ERC grant agreement n° 258865.

References

1. ISO/IEC 18033-2:2006 - Information technology – security techniques – encryption algorithms – Part 2: Asymmetric ciphers.

2. Divesh Aggarwal and Ueli Maurer. The leakage-resilience limit of a computational problem is equal to its unpredictability entropy. In Dong Hoon Lee and Xiaoyun Wang, editors, *ASIACRYPT*, volume 7073 of *LNCS*, pages 686–701. Springer, 2011.
3. Joël Alwen, Yevgeniy Dodis, and Daniel Wichs. Leakage-resilient public-key cryptography in the bounded-retrieval model. In Shai Halevi, editor, *CRYPTO*, volume 5677 of *LNCS*, pages 36–54. Springer, 2009.
4. Dan Boneh and Xavier Boyen. Efficient selective-id secure identity-based encryption without random oracles. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT*, volume 3027 of *LNCS*, pages 223–238. Springer, 2004.
5. Dan Boneh and Xavier Boyen. Short signatures without random oracles and the sdh assumption in bilinear groups. *J. Cryptology*, 21(2):149–177, 2008.
6. Dan Boneh, Xavier Boyen, and Eu-Jin Goh. Hierarchical identity based encryption with constant size ciphertext. In Cramer [10], pages 440–456.
7. Elette Boyle, Gil Segev, and Daniel Wichs. Fully leakage-resilient signatures. In Kenneth G. Paterson, editor, *EUROCRYPT*, volume 6632 of *LNCS*, pages 89–108. Springer, 2011.
8. Zvika Brakerski, Yael Tauman Kalai, Jonathan Katz, and Vinod Vaikuntanathan. Overcoming the hole in the bucket: Public-key cryptography resilient to continual memory leakage. In *FOCS*, pages 501–510. IEEE Computer Society, 2010.
9. Jean-Sébastien Coron. Resistance against differential power analysis for elliptic curve cryptosystems. In Çetin Kaya Koç and Christof Paar, editors, *CHES*, volume 1717 of *LNCS*, pages 292–302. Springer, 1999.
10. Ronald Cramer, editor. *Advances in Cryptology - EUROCRYPT 2005, 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005, Proceedings*, volume 3494 of *LNCS*. Springer, 2005.
11. Yevgeniy Dodis, Kristiyan Haralambiev, Adriana López-Alt, and Daniel Wichs. Efficient public-key cryptography in the presence of key leakage. In Masayuki Abe, editor, *ASIACRYPT*, volume 6477 of *LNCS*, pages 613–631. Springer, 2010.
12. Yevgeniy Dodis, Rafail Ostrovsky, Leonid Reyzin, and Adam Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. *SIAM J. Comput.*, 38(1):97–139, 2008.
13. Danny Dolev, Cynthia Dwork, and Moni Naor. Non-malleable cryptography. *SIAM J. Computing*, 30(2):391–437, 2000.
14. Stefan Dziembowski and Sebastian Faust. Leakage-resilient cryptography from the inner-product extractor. In Dong Hoon Lee and Xiaoyun Wang, editors, *ASIACRYPT*, volume 7073 of *LNCS*, pages 702–721. Springer, 2011.
15. Stefan Dziembowski and Krzysztof Pietrzak. Leakage-resilient cryptography. In *FOCS*, pages 293–302. IEEE, 2008.
16. Sebastian Faust, Eike Kiltz, Krzysztof Pietrzak, and Guy N. Rothblum. Leakage-resilient signatures. In Daniele Micciancio, editor, *TCC*, volume 5978 of *LNCS*, pages 343–360. Springer, 2010.
17. Sebastian Faust, Krzysztof Pietrzak, and Joachim Schipper. Practical leakage-resilient symmetric cryptography. In Emmanuel Prouff and Patrick Schaumont, editors, *CHES*, volume 7428 of *LNCS*, pages 213–232. Springer, 2012.
18. David Galindo and Srinivas Vivek. A practical leakage-resilient signature scheme in the generic group model. In *SAC 2012*, volume 7707 of *LNCS*, pages 50–65. Springer, 2013.
19. Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, 1984.

20. 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.
21. Jonathan Katz. Signature schemes with bounded leakage resilience. Cryptology ePrint Archive, Report 2009/220, 2009. `http://eprint.iacr.org/`.
22. Jonathan Katz and Vinod Vaikuntanathan. Signature schemes with bounded leakage resilience. In Mitsuru Matsui, editor, *ASIACRYPT*, volume 5912 of *LNCS*, pages 703–720. Springer, 2009.
23. Eike Kiltz and Krzysztof Pietrzak. Leakage resilient elgamal encryption. In Masayuki Abe, editor, *ASIACRYPT*, volume 6477 of *LNCS*, pages 595–612. Springer, 2010.
24. Paul C. Kocher. Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems. In Neal Koblitz, editor, *CRYPTO*, volume 1109 of *LNCS*, pages 104–113. Springer, 1996.
25. Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In Michael J. Wiener, editor, *CRYPTO*, volume 1666 of *LNCS*, pages 388–397. Springer, 1999.
26. Feng-Hao Liu and Anna Lysyanskaya. Tamper and leakage resilience in the split-state model. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO*, volume 7417 of *LNCS*, pages 517–532. Springer, 2012.
27. Tal Malkin, Isamu Teranishi, Yevgeniy Vahlis, and Moti Yung. Signatures resilient to continual leakage on memory and computation. In Yuval Ishai, editor, *TCC*, volume 6597 of *LNCS*, pages 89–106. Springer, 2011.
28. Ueli M. Maurer. Abstract models of computation in cryptography. In Nigel P. Smart, editor, *IMA Int. Conf.*, volume 3796 of *LNCS*, pages 1–12. Springer, 2005.
29. Gregory Neven, Nigel P. Smart, and Bogdan Warinschi. Hash function requirements for schnorr signatures. *J. Mathematical Cryptology*, 3(1):69–87, 2009.
30. Tatsuaki Okamoto. Provably secure and practical identification schemes and corresponding signature schemes. In Ernest F. Brickell, editor, *CRYPTO*, volume 740 of *LNCS*, pages 31–53. Springer, 1992.
31. Krzysztof Pietrzak. A leakage-resilient mode of operation. In Antoine Joux, editor, *EUROCRYPT*, volume 5479 of *LNCS*, pages 462–482. Springer, 2009.
32. Claus-Peter Schnorr. Efficient identification and signatures for smart cards. In Gilles Brassard, editor, *CRYPTO*, volume 435 of *LNCS*, pages 239–252. Springer, 1989.
33. Claus-Peter Schnorr. Efficient signature generation by smart cards. *J. Cryptology*, 4(3):161–174, 1991.
34. Jacob T. Schwartz. Fast probabilistic algorithms for verification of polynomial identities. *J. ACM*, 27(4):701–717, 1980.
35. Michael Scott. On the efficient implementation of pairing-based protocols. In Liqun Chen, editor, *IMA Int. Conf.*, volume 7089 of *LNCS*, pages 296–308. Springer, 2011.
36. Victor Shoup. Lower bounds for discrete logarithms and related problems. In Walter Fumy, editor, *EUROCRYPT*, volume 1233 of *LNCS*, pages 256–266. Springer, 1997.
37. Brent Waters. Efficient identity-based encryption without random oracles. In Cramer [10], pages 114–127.
38. Daniel Wichs. Barriers in cryptography with weak, correlated and leaky sources. In Robert D. Kleinberg, editor, *ITCS*, pages 111–126. ACM, 2013.
39. Richard Zippel. Probabilistic algorithms for sparse polynomials. In Edward W. Ng, editor, *EUROSAM*, volume 72 of *LNCS*, pages 216–226. Springer, 1979.