PhD-FSTM-2024-006
The Faculty of Science, Technology and Medicine

DISSERTATION

Defence held on 24/01/2024 in Esch-sur-Alzette

to obtain the degree of

DOCTEUR DE L'UNIVERSITÉ DU LUXEMBOURG
EN INFORMATIQUE

by

Reynaldo GIL PONS

Born on 17 October 1992 in Santiago de Cuba (Cuba)

ANALYSIS OF SECURITY PROTOCOLS WITH SECURITY PROPERTIES BASED ON DISTANCE

Dissertation defence committee

Abstract

Security protocols are commonplace in current digital communications. Achieving secure, private and efficient communications has sparked the expansion of many fields in Computer Science at the intersection between cryptography, formal methods and the theory of computation. History has shown the first steps to achieve security are to model the problem, state its desired properties, and prove them correct with mathematical rigour. Several methods have been demonstrated useful for such tasks, from pen and paper proofs in the standard model, to semi-automated proofs in the symbolic model.

Recently, protocols that depend on proximity have gained much popularity and importance. They are used in contactless technologies such as contactless payment, Near Field Communication (NFC) on smartphones and e-passports. These protocols assume the agents participating in the protocol are physically close during the execution. When this condition is not enforced, they become vulnerable to the so-called relay attacks. This type of attack occurs when an attacker relays the communication between agents, thus making them believe that they are communicating directly. In some cases this is a vulnerability, as direct communication indicates acceptance to execute a transaction. Distance-bounding protocols are an important class of protocols which aim to guarantee that the agents executing the protocol are physically close. These have been thoroughly studied in the last decade. Nevertheless, there are several protocols which are not in this class, but still use proximity notions in order to achieve security. One example is peer-to-peer key-exchange protocols where authentication is achieved solely through physical proximity. The first objective of this thesis is to study these protocols.

First, we develop a symbolic framework to generically analyse protocols that consider attackers far away from the agents executing the protocol. We call this condition the distant-attacker assumption. These protocols usually utilize time measurements to guarantee that the partner of communication is nearby, similar to what is done in distance bounding. The key difference though, is that the security property sought is different, as their aim is to achieve other properties such as secrecy, authentication or memory-erasure. Using the proposed framework we show how these properties can be proved semi-automatically using standard tools. Finally, we evaluate the security of several protocols in the literature that had not been formally verified yet.

Second, we study software-based memory-erasure protocols. These protocols are useful to guarantee small devices are in a safe state (for example free of malware) without having to physically access the device. In these protocols the prover aims to convince the verifier that it has erased its memory. In the literature, software-based erasure protocols have typically assumed that the prover is isolated during the execution. We show how this restriction can be lifted using distance-bounding techniques. To this end, we propose new protocols and prove them secure within a computational model assuming an attacker that is distant rather than absent.

Finally, we solve an open problem related to the probability of success of attackers against lookup-based distance-bounding protocols. We propose a new protocol and prove it optimal with respect to mafia-fraud and distance-fraud attacks using probabilistic analysis in a computational model.

Acknowledgements

Personal achievements are seldom obtained in solitude. Most human beings, no matter their origin or culture, depend on others for the most part of their life. These words are my sincere attempt to say *thank you* to everyone who deserves it.

I would like to thank Sjouke for his accurate advice and guidance in the last four years, specially for sharing insights about the human side of research and work. It has been a great pleasure to be his teaching assistance in TCS I. Thanks Rolando for the uncountable hours spent making this thesis move forward from the start to the end. Specially for having the patience to work with me continuously for such a long period of time.

I want to deeply express my gratitude to the rest of the defence committee: Prof. Dr. Peter Ryan, Dr. Sasa Radomirovic, and Dr. Ralf Sasse. Thank you all for taking the time to review my thesis, your inputs made the thesis and its defence a much nicer and interesting work.

I would certainly not be here without having had the family I had. The teachings and love of my grandparents Aurora, Antonio, Isabel and Reinaldo gave meaning to my life from the very start. I will always remember the joyful moments shared with and the support given by my sister Aurorita, my father "Pepe", my aunts, my uncles and my cousins. Special thanks to my partner Bego, who helped to make this PhD a complete life experience, and not just an academic one. Without doubt, this thesis is dedicated to my mother, who still today guides my thinking and my hopes for the future, permanently showing me the correct path to take and the best choice to make in every circumstance.

Working on a PhD thesis would have been probably quite boring without colleagues with which share daily moments and thoughts. I want to thank my friends and colleagues: Sergiu, Yunior, Sevdenur, Semyon, Xihui, Aoran, Manel, Chenxiang, Ross, Tsz Pan, Tomer, Hailong, Ninghan, Zhiqiang, Adrien, Mike, Aleksandr, Cui and Zachary.

Finally, I want to thank the Luxembourg National Research Fund (FNR) for the financial support.

Thank you very much,

Reynaldo Gil Pons

Contents

List of Figures

List of Tables

1 Introduction

Computer networks have changed how people communicate. Internet, the global computer network, is necessarily present in private communications, payments, news distribution, knowledge dissemination and government services. By their nature, these services require means of communication that guarantee the secrecy, integrity and authenticity of the data exchanged. But obtaining certainty about the security of communication methods has not been easily achieved. Furthermore, the ideas necessary for designing these methods were considered military secrets or inexistent, making most classical means of communication basically insecure. Scientifically proven methods came to solve this issue in the XX century through the field of cryptography.

Cryptography has made secure communication systems possible. Its main research methods are in the intersection between mathematics and computer science. Results from number theory [136], computational complexity [126], information theory [94] and quantum computing [97] have been applied (and in many cases discovered) in order to create the so-called cryptographic primitives. These are functions with characteristics that enable security properties. For example, encryption functions map strings to strings in such a way that the reverse map cannot be computed in reasonable time unless a secret is known. Likewise, hash functions compress any string into a fixed size string, and have the property that finding a single collision (two strings with the same image) is "impossible" with current technology.

In modern applications cryptographic primitives are seldom used in isolation, but as part of security protocols. These protocols are, in general, interactive algorithms executed between a set of parties which communicate using a network. These algorithms use cryptographic primitives to achieve a predefined set of goals. Examples of widely deployed security protocols are: TLS [118] (used to provide security to the Hypertext Transfer Protocol (HTTP)), Signal [112] (used by WhatsApp and others), SSH [138] (used to remotely access computers), WireGuard [56] (remote access to networks), InterPlanetary File System [23] and BitTorrent [39] (file sharing), and Bitcoin [104] (decentralized payments). The security goals of these protocols are quite varied. For example, TLS aims to provide a communication channel between two parties with confidentiality (messages sent through the channel are secret), integrity (messages cannot be modified by third-parties), and authenticity (messages come from the expected party). When these properties are valid, eavesdropping and tampering are impossible, even when attackers are in full control of the network.

Showing that a security protocol is secure against all possible attacks is an extremely difficult task. Case in point, even the omnipresent TLS protocol, after years of scrutiny in dozens of research papers, has been shown to be vulnerable to attacks because of weaknesses at the protocol level. We refer in particular to the triple handshake attack in [26]. The attack followed from the unexpected composition of standard features of the protocol: session resumption followed by client authentication during renegotiation. It affected all major implementations of the protocol, but luckily was relatively easy to fix.

1.1 Protocols with proximity measurements

In general, there are protocol that are cryptographically secure, but still can be attacked in practice. The reason is that cryptography usually overlooks variables such as computational resources and time. For example, many important protocols depend on time for protecting against distributed denial-of-service (DDoS) attacks. These have caused the downtime of very popular services such as search engines and social networks. An attack begins when the number of clients starting new sessions with the server grows too fast in a very short period of time. Without the right measures in place, the amount of data that needs to be processed goes beyond what the server can handle, and then it becomes effectively disconnected from the rest of the network. DDoS attacks can be prevented by using time-aware detection techniques. Another example in which time is necessary is packet retransmission. Given that networks are usually unreliable, dropping messages is a common situation for which low level protocols must have an efficient recovery path. In this case, if a message is not acknowledged after a certain period of time, it must be assumed lost and has to be retransmitted.

Contactless payments are another example where time is important to achieve some level of security. When this technology is used to pay in a supermarket, a protocol is executed between the payment terminal and the user card. Normally, no user interaction is required for the payment to be successful, but the security of this transaction depends on the proximity between the card and the terminal. If this condition is not enforced, then so-called relay attacks become feasible: a card in the pocket of a user could be used to pay on a terminal in another location by using an attacker's controlled network to relay messages between them. Relay attacks were mostly a theoretical possibility [33] until they were successfully used against contactless systems [66]. Another application where proximity is required are passive keyless entry and start systems, which allow cars to be opened and started when its key is nearby [64].

The relay attacks on payment systems described before can be prevented if the terminal ensures the card is physically close at the time of payment. We call this process proximity verification, which is an essential part of protocols which depend on the agents locations. The most common way of proximity verification is to measure the round-trip-time (RTT) of messages to estimate the distance between agents. In payment systems, for example, a terminal should only accept a payment from a card if the responses to its challenge messages arrive within a predetermined time-bound, which corresponds to the card being physically close while executing the protocol.

This thesis focuses on the security analysis of protocols with proximity measurements. These are two-party protocols (verifier and prover), where the parties are able to measure the time that passes between sending certain messages (challenges) and receiving a message back from the prover (responses). Such protocols usually include a fast phase, in which several rounds of challenge-response messages occur. In each round the verifier sends a challenge and receives a response from the prover. The verifier measures the RTT of each round, starting when the challenge is sent and ending when the response is received. The intuition behind these measurements is that if each response is received very fast, then the entity sending it must be very close (thus preventing relay attacks), and must not have done much computation in between. This follows from the fact that communication is constrained by the maximum speed at which information can propagate (speed of light), and so is the computation speed (although in this case we are still very far from that limit[1]).

Protocols with proximity measurements can be classified according to the security property they aim to achieve. Several examples can be found in the literature: distance bounding [33, 75], authentication [135], key exchange [123], and software-based memory erasure [129]. Distance-bounding protocols are a special case deserve special attention because their security goal is proximity verification. Hence, their verification mechanisms and proof techniques serve as inspiration to other protocols with proximity measures that aim at other security properties, such authentication, memory-erasure, etc. In the next few paragraphs we introduce these families of protocols in more detail.

Distance-bounding protocols [25, 33] were the first to include RTT measurements in order to prevent relay attacks. Pioneer work formally verifying these protocols with tool support did not appear until two decades later [21]. A distance-bounding protocol is executed between a verifier and a prover. It consists of three phases in this order: initial phase, fast phase and final phase. Messages are exchanged normally during the initial and the final phase, while the RTT is measured during the fast phase.

Depending on the relative positions of agents during the execution of the protocol, the attacks against distance-bounding protocols have been traditionally classified in: distance fraud (when the prover is malicious far away from the verifier), mafia fraud (when the prover is honest and far way from the verifier), distance hijacking (like distance fraud but requiring another nearby honest prover) and terrorist fraud (a far away dishonest prover convinces the verifier it is nearby with the help of a nearby attacker without revealing its secret keys). There have been numerous distance-bounding protocols, each improving upon previous work regarding efficiency or security. These protocols have been analysed in the computational model by computing the probability of success of an adversary against each type of attack [96], and in the symbolic model with the help of semi-automated tools [50, 95].

Distance-bounding protocols achieve authentication by using public key infrastructures or pre-shared secret keys. This is not possible for some applications, such as peer-to-peer communications between low-powered Internet of Things (IoT) devices. Protocols aimed for this setting, such as the authentication protocol from [135] or the key-exchange protocol from [123] must achieve security through other means. In particular, they assume that agents running the protocol are close to each other, and that no attacker is near them.

A typical example is the protocol from Singelee [123], shown in Figure 1.1. It follows the traditional Diffie-Hellman protocol for computing a shared secret key. It is well-known that the original protocol is vulnerable to man-in-the-middle attackers. To solve this issue, in the DB-Based-Diffie-Hellman each party executes RTT measurements to verify the partner of communication is nearby. This, together with the assumption that attackers are far away, rules out the aforementioned vulnerability.

Protocols which assume attackers are far away are similar to distance-bounding protocols in the sense that both use RTT measurements to estimate distance, but at the same time different

[1] `https://www.top500.org/lists/top500/2022/11/`

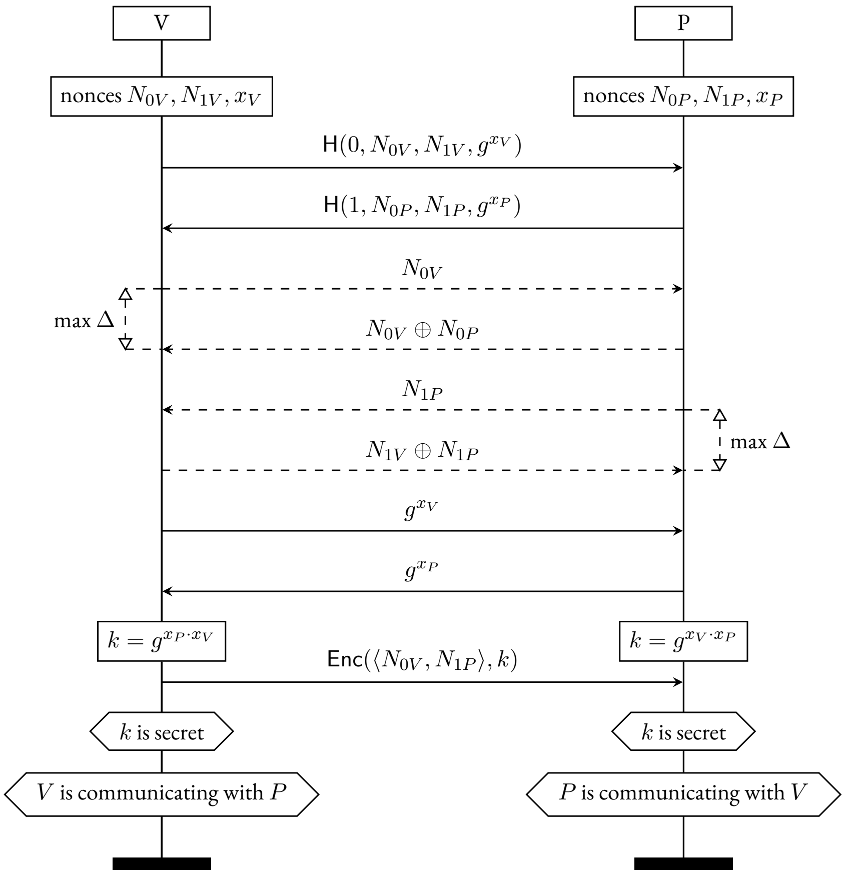

V
P
nonces N_{0V}, N_{1V}, x_V
nonces N_{0P}, N_{1P}, x_P
$\mathsf{H}(0, N_{0V}, N_{1V}, g^{x_V})$
$\mathsf{H}(1, N_{0P}, N_{1P}, g^{x_P})$
N_{0V}
max Δ
$N_{0V} \oplus N_{0P}$
N_{1P}
max Δ
$N_{1V} \oplus N_{1P}$
g^{x_V}
g^{x_P}
$k = g^{x_P \cdot x_V}$
$k = g^{x_V \cdot x_P}$
$\mathsf{Enc}(\langle N_{0V}, N_{1P} \rangle, k)$
k is secret
k is secret
V is communicating with P
P is communicating with V

Figure 1.1: The DB-Based-Diffie-Hellman key-exchange protocol

because the security properties sought for are distinct. While in the former the verifier needs to check if the prover is close-by, these protocols aim to achieve a standard security property under the assumption that attackers are far away. Therefore, verifying these protocols requires a different attacker model, and is currently an open problem.

Remote memory-erasure protocols guarantee that devices are malware free, i.e. not controlled by an adversary. These are two party protocols where the verifier wants to check if the prover has erased its memory while running the protocol. Being remote makes them suitable for applications where physical access is hard or not desirable. These protocols are hard to analyse because the attacker model considers the device being erased to be fully controlled by the adversary. They are usually classified into hardware based, hybrid or software-based. While the hardware based ones require secure hardware such as a Trusted Platform Module (TPM), software-based protocols can be used in any computing device but offer lower security guarantees, and hybrid protocols sit in the middle mixing techniques of the previous two. The presence of secure hardware greatly simplifies the analysis as it weakens the adversary's control over the device. In the case of low-powered and legacy IoT devices, only software-based protocols are feasible, given that adding secure hardware would greatly increase their overall cost and is not desirable for already deployed devices. The main shortcoming of existing software-based memory-erasure protocols is the isolation requirement: during the execution of the protocol the device cannot not receive any external help, which implies it must make every computation locally. This requirement is hard to guarantee in applications where devices are connected to the Internet, as is usual in the IoT.

1.2 Formal Analysis

Although most protocols are designed with security in mind, many protocols which remained secure for a long time have been broken afterwards. A famous example is the authentication protocol proposed by Needham and Schroeder [105]. 20 years later an attack was found by Lowe [90]. The attack consisted of a clever selection of the order of messages, without breaking any of the primitives, and was found using computer programs. This was not a new idea, but one which had not showed its importance until that moment.

In order to prove that a protocol is "secure", researchers have traditionally used formal methods [98]. These require mathematically sound (and preferably realistic) models for how the network works, what actions can an attacker do, who are the entities participating in the protocol and what property it is supposed to possess. Coming up with these models is no trivial task. After decades of research, still today, sometimes a protocol is proved secure (correctly) under one model, but then becomes insecure as soon as the model is extended to account for more realistic behaviours. Recent examples of protocols that were once proven secure are the WPA2 authentication protocol in Wi-Fi, which was broken in [134] and the SSH protocol, broken in [1].

As mentioned before, this thesis focuses on the analysis of security protocols with proximity measurements through RTT. In the field of protocol analysis, researchers usually use either the symbolic or the computational approach, both of which are supported by mathematical models of adversarial behaviour, network communication, cryptographic primitives and protocol's execution. Here we use both, by selecting the most appropriate approach to solve each problem we target.

1.2.1 The symbolic approach

In the last several decades, a large amount of researchers has aimed to create provably secure cryptographic primitives. Today, we are quite certain that the primitives we use for encryption, pseudorandom generators, hash functions, signatures, and many others are secure in the sense that in order to break them any attacker would need to do an unfeasible amount of computation. Still, having secure primitives might not be enough for achieving security, as the interactions of these primitives might result in unforeseeable outcomes. The nature of these interactions made proving their security a highly non-trivial task, which, if properly done by hand, would take too much time to prove and almost as much time to verify. The complexity of the task was not that much on its difficulty, but in the length of the proofs itself. Therefore, the natural idea of using computers to do the hard work came up, thereby creating the symbolic method.

On the highest level of abstraction, in the symbolic model attackers are modelled as algorithms which are restricted to use messages that can be represented as elements from a term algebra. The most common attacker model is the Dolev-Yao model, which assumes the attacker is in full control of the network [55]. For certain applications the Dolev-Yao model is too general and more fine-grained models are required. This is, for example, the case for post-compromise security [41], where the adversary may obtain private key material that was used at some point during the protocol. In some cases, it is still possible to achieve some level of security in future sessions, or even fully recover from the leakage.

In the symbolic model security primitives are considered perfectly secure, which implies that no attack on the primitives itself is possible. Nevertheless, this assumption makes the problem of finding attacks (and security proofs) on larger and complex protocols more tractable. The execution of protocols is usually represented by traces of events, where events might be sending or receiving messages, and security claims. Most well known security properties can be classified as reachability properties or equivalence properties. While the former include secrecy and authentication, the latter comprise anonymity and unlinkability. The security of a protocol is determined by checking properties of the set of all possible traces. For example, in the case of reachability properties, a protocol is secure if and only if a certain event never appears in the set of traces. The concepts mentioned before are the main ingredients of the symbolic method [105]. Using this approach, it is frequently possible to automatize the search for security proofs, which makes it feasible to analyse larger protocols.

Our first objective in this thesis is to formally model and verify security properties in protocols which include proximity measures. To this end, we formalize a new attacker model, the distant-attacker model, where attackers are still in full control of the network, but must act from a distance. The notion of distant-attacker was recently introduced in [129] within the context of memory-erasure protocols. Their security model, however, is neither amenable for computer-aided verification nor extendable to other security properties.

There exist various symbolic models to analyse security protocols that depend on time and location, although most of them target distance-bounding protocols [21, 95, 99]. Up to our knowledge, no symbolic model with tool support has been proposed to analyse protocols which aim to achieve standard properties such as secrecy and authentication under the distant-attacker assumption. Our objective is to verify security properties by framing them as reachability properties under a symbolic model. This leads us to our first research question:

Research question 1. Can we use automated tools to rigorously analyse the security of protocols which assume the distant-attacker model?

1.2.2 The computational approach

For some protocols such as memory-erasure, security depends on the computational characteristics of the participants, such as their memory size. Therefore, these protocols are more naturally analysed using the computational approach. Within the formal methods, this is the lowest level of abstraction, where attackers are modelled as algorithms (which can be represented by Turing Machines) and the security bounds are probabilistic. The computational approach [72, 73, 137] has been popular among cryptographers since the 80s. In this approach, proofs are classical mathematical proofs with no computer support; the so-called pen-and-paper proofs.

The first protocol for secure memory-erasure based on cryptographic techniques was introduced by Perito and Tsudik [111]. They proposed a protocol based on computing a (time-optimal) message authentication code (MAC) over randomly chosen data that occupies the memory of the device being erased. This protocol uses a high bandwidth to transmit the random data and cannot prevent the computation to be delegated to an external attacker without isolating the device. Later works proposed graph-based protocols to reduce the communication through the network [58, 81]. These protocols, however, still need to assume that the protocol run in an isolated environment to prove their security. Hence, our second objective is to drop this assumption.

The main challenge in this setting is to create an erasure algorithm that remains secure even if the prover receives external help during the protocol execution. To prove this type of assertion, we need to model the space on the prover explicitly. Therefore, we attack the problem using the computational approach. To this end, we model the attacker as two algorithms: one with unbounded computational resources, in particular memory, that acts from a distance, e.g. the Internet, and the other restricted to use the computational resources available in the device being erased, acting as prover in the same local network as the verifier. This leads us to our second research question:

Research question 2. Can we design software-based remote memory-erasure protocols that drop the device isolation assumption and yet achieve good security bounds?

The main challenge in proving the security bound in memory-erasure is how to reason about the compression power of the adversary, i.e. how much the attacker can compress data to leave space available for malware. A powerful proof technique often used in the literature is to translate the adversary behaviour into a combinatorial game, called the pebbling game. The security of the graph based memory-erasure protocols mentioned above depends on these games and their relation with the computation of labelling functions. Pebbling games are single player combinatorial games played on graphs, which are at the core of many time-space tradeoff results [57]. During each step of the game, a pebble is removed from a node (if it contains a pebble) or added to a node (if the predecessors are already pebbled). The objective of the game is to put pebbles in certain nodes called outputs. The maximum amount of pebbles in the graph at any point during a game is its pebbling complexity. A labelling function assigns a label to each node of the graph, which is equal to the hash of the concatenation of the labels of its predecessors. Several bounds have been proven [57, 59] that relate the pebbling complexity of a graph and the amount of memory required by any algorithm computing a labelling function on the same graph. The tightness of these bounds is essential to create memory-erasure protocols that can erase the full memory of the device. This leads us to our third research question:

Research question 3. Are there tight time-space bounds translating pebbling complexity results to computational adversaries?

In this thesis we go beyond security analyses, as we also aim to understand whether software-based memory-erasure protocols are feasible in practice. Although several protocols for memory-erasure have been proposed in the literature [79, 81, 111], they have never been implemented nor compared in the same practical setting. Having such a comparison is essential for practitioners willing to deploy these protocols, as they usually offer non-trivial trade-offs between speed and security. This leads us to our fourth research question:

Research question 4. Are software-based remote memory-erasure protocols practical?

Security-memory trade-offs problems, mentioned before for memory-erasure, are also present in the class of distance-bounding protocols, the most popular class of protocols with proximity measures. These protocols have also been widely studied using the computational method. Usually, the analysis is done with respect to particular attacks that depend on the attacker's location, such as mafia fraud and distance fraud. As mentioned before, the former is a man-in-the-middle attack aimed at convincing the verifier that an honest far-away prover is close; the latter is an attack executed by a corrupt prover to pass the protocol from far-away. For many protocols, researchers using computational analysis have obtained upper bounds to the probability of success of these types of attack.

For the class of lookup-based protocols [96], exact probabilistic analysis is possible. These protocols have in common that during each round of the fast phase the prover makes a simple lookup operation to compute the response to the challenge. Therefore, it is possible to represent the protocol using automata theory, where each execution of the protocol corresponds to a path in an automaton.

The analysis of lookup-based protocols has mainly focused on the two threats mentioned above [15, 16]. So far, optimal resistance against mafia fraud has only been achieved at the expense of using a look-up table of exponential size [20]. Furthermore, it was proven in [96] that the optimal resistance against mafia fraud is $\frac{1}{2^n}\left(\frac{n}{2}+1\right)$, where n is the number of round-trip-time measurements, and in [128] that the optimal resistance against distance fraud is $\frac{1}{2^n}$. It remains an open question, however, whether there exists a lookup-based protocol achieving both optimal security bounds. Our final research question aims to solve this issue:

Research question 5. Is there a distance-bounding protocol that is optimal with respect to mafia fraud and distance fraud attackers?

1.3 Overview and contributions

We briefly sketch the organization of the thesis, and summarize the contributions made in each chapter:

1. In Chapter 2 we review previous works in the main three problems studied in this thesis: protocols with proximity requirements, memory-erasure and distance-bounding.

2. In Chapter 3 we introduce the symbolic approach and the necessary background knowledge. In Chapter 4 we propose abstract models (with and without time) to enable the formal verification of protocols with proximity measurements that claim security against distant attackers. Time constraints are abstracted as precedence relations in an ordered trace in the timeless model, which makes it possible to verify security properties using tools such as Tamarin. Furthermore, we show the equivalence on the existence of attacks between both abstract models, with or without time, for a class of reachability properties. We then prove that this class includes secrecy, authentication, memory-erasure and distance hijacking. Furthermore, we test our model through case studies, obtaining new results for several protocols. These results answer Research question 1, and are derived from work published in [67].

3. In the subsequent chapters, we employ a computational approach. We begin by explaining its basic theoretical results in Chapter 5 such as the random oracle model (ROM), necessary

to understand posterior chapters. In Chapter 6 we propose a computational model covering memory-erasure protocols and the distant-attacker assumption. Furthermore, we design a simple software-based memory-erasure protocol and prove it secure under our model. This protocol suffers from high communication cost.

4. In Chapter 7 we introduce the theory of pebble games. Furthermore, we revise the original bound using the ex post facto argument, a known result related to the translation of combinatorial bounds into computational bounds. By finding a counter example, we address Research question 3.

5. The previous results are necessary background for our graph-based memory-erasure proposal from Chapter 8. The aim of this protocol is to reduce the communication complexity while remaining secure. It is based on the theory pebbling games, and the communication cost during the initial phase of the protocol is constant. Furthermore, we propose a lightweight version of the protocol, trading security for efficiency, which might be more practical in some applications. These results address Research question 2, and are based on the paper [34].

6. In Chapter 9 we implement previous memory-erasure protocols, as well as the new ones proposed in this thesis, and compared them within the same practical setting. We conduct experiments using off-the-shelf low powered IoT devices from Texas Instruments. Furthermore, we highlight the security guarantees offered by each protocol, its theoretical efficiency and its practical performance. We have open-sourced our experimental artefacts and results in[2]. These results address Research question 4.

7. In Chapter 10 we investigate the existence of a lookup-based distance-bounding protocol optimal against mafia fraud and distant fraud attackers. First we prove that optimality against both kinds of attackers cannot be achieved at the same time. Then we show that assuming optimality against distance fraud, there exists a conditionally optimal value for the resistance against mafia fraud. Finally, we design a new protocol and prove it has the required optimality characteristics. These results address Research question 5 and are derived from work published in [68].

[2]`https://gitlab.uni.lu/regil/memory-erasure-experiments`

2 Related work

In this chapter we describe previous works related to the main problems in this thesis: protocols with proximity requirements, memory-erasure and distance-bounding.

2.1 Protocols with proximity requirements

In the last decades several protocols with proximity requirements have been published [123, 135]. They assume agents executing the protocol are nearby while attackers control the network from a distance. We refer to this as the distant-attacker assumption. To enforce that communication partners are nearby, the aforementioned protocols utilize round-trip-time measurements to estimate distance. Therefore, modelling such protocols requires the notions of time and agent's location. In this section we review symbolic models with these characteristics, and discuss their applicability for the class of protocols with proximity requirements.

There exist various symbolic models to analyse security protocols that depend on time and location, most of them specifically targeting distance-bounding protocols. The first one [99] was proposed in 2007. This model allowed for the analysis of distance-bounding protocols while faithfully representing time and location, but lacked support for computer-aided verification. Basin et al. [21] later addressed that limitation, introducing a security model for distance-bounding protocols with tool support. They made explicit the dependence between the arrival time of a message and the locations of sender and receiver, and the maximum propagation speed of the communication channel. This allows their model to formulate distance-bounding security as a statement on whether a round-trip-time measurement is lower than or equal to twice the distance to the communicating partner divided by the propagation speed. Our model in Chapter 4 is inspired by these modelling choices. To assist formal verification, Basin et al. encoded their model in the theorem proving assistant Isabelle/HOL. Their approach is not fully automated, though, requiring end-users to define several protocol-dependent lemmas.

The problem of realizing a fully automated verification framework for distance-bounding protocols was solved independently in 2018 by Mauw et al. [95], Chothia et al. [38], and Debant et al. [50, 51]. In [49] a procedure to analyse these protocols is presented and integrated in Akiss[1]. Our work intersects with all those seminal works in different ways. Like in [95], we frame time-based protocol requirements as causal relations of protocol events, and show they can be automatically verified in Tamarin. To prove equivalence between a time-based model and a causality-based model, we use proving techniques similar to the one used in [38, 50, 51] to analyse distance-hijacking resistance. We generalize these approaches under the assumption of a distant attacker by supporting other security properties, such as secrecy, agreement and memory-erasure.

[1] `http://people.irisa.fr/Alexandre.Debant/akiss-db.html`

Improvements and extensions of the above verification frameworks have followed. For example, Boureanu et al. [31] introduce a model that supports moving agents and use it to analyse complex payment protocols. Alturki et al. [2] propose a timed multiset rewriting model with memory bounding and timeouts features. Their analyses are of theoretical interest, but their model lacks tool support.

Outside the domain of distance-bounding protocols, there exist time-aware models [43, 78, 125] that support the verification of standard authentication properties, as we do in Chapter 4. Their specification language is richer than ours, allowing for statements on timeouts and the ordering of protocol events based on timestamps. Their network model, however, neither captures the location of agents nor the propagation time of a message via the network in terms of the distance between sender and receiver. Instead, they model the network communication delay as a range within a discrete space, making these models unsuitable to formalize the notion of distant-attacker.

Lastly, the notion of distant-attacker was introduced in [129], from which we borrow the terminology, within the context of memory-erasure protocols. Their security model, however, is neither amenable for computer-aided verification nor extendable to other security properties.

Up to our knowledge, no symbolic model with tool support has been proposed to analyse protocols in which standard properties such as secrecy and authentication can be analysed under the distant-attacker assumption. We introduce the first such model in this thesis, and use it to verify key-exchange and memory-erasure protocols.

2.2 Memory-erasure

In this section we review previous work on software-based memory-erasure (PoSE) protocols. We also review memory attestation protocols, where the verifier aims to check if the contents of the prover's memory equal a predetermined value. The techniques used to solve both problems have multiple similarities, therefore we will review works attempting to solve each problem. In particular, we review memory-erasure and attestation protocols for resource-constrained devices that do not use secure hardware. See [87] for a review of protocols that use it. We will see that the main limitation of current software-based protocols, as highlighted in a recent survey [10], is the assumption that there is a secure communication channel between verifier and prover, called the *device isolation assumption*.

Memory isolation. Several memory attestation/erasure protocols use software-based memory isolation techniques that continuously monitors the programs running on the device. In this class, SPEED [9] and SIMPLE [8] perform memory-erasure and attestation, respectively. These protocols could, in theory, achieve our security goals. In practice, they might add a significant overhead to any program running on the platform, as the software memory protection continuously monitors the platform to ensure memory isolation at all times. Since memory-erasure can start from any state of the device, we do not need to impose constraints on programs running on it. SPEED uses distance-bounding to ensure that only a nearby verifier can start the memory-erasure procedure. In our memory-erasure protocols from Chapters 6 and 8 we use distance-bounding in the opposite direction, where the verifier is the challenger aiming to ensure the prover is near.

Time-optimal routine. SWATT is an attestation protocol [121] based on computing a checksum function over the memory of the prover within a prescribed amount of time. The idea is that, had the prover's memory been compromised, the prover would take noticeably longer to generate the correct attestation proof. Several attacks on SWATT were shown in [36], the main problem being that the checksum function and the timing constraints are rather ad hoc. This problem was later treated in [12], which proposes a formal model of software-based attestation and a generic protocol similar to SWATT, but, this time with a formal security proof. The problem here is that, if the local device is not isolated, the attacker could compute the response on a separate device and return it in time, so that this is not noticed by the verifier. Another problem is that the global running time of the erasure procedure is too high in order to detect small changes to the memory. The attacker could use the available time to optimize procedures and recompute the data that is missing. That is why [12] suggests the idea of a multi-round protocol, requiring shorter computation time in each round. This is an idea we develop further in Chapter 6 as part of a distance-bounding mechanism.

The first protocol for secure memory-erasure based on cryptographic techniques is introduced in [111]. The paper proposes a protocol based on computing a hash-based message authentication code (HMAC) over randomly generated data that fills the memory of the prover, along with an informal adversarial model and argument of security. This protocol uses a high bandwidth to transmit the random data and cannot prevent the computation to be delegated to an external attacker without isolating the device, like in [12, 121].

Distance-bounding techniques. A PoSE protocol that relies on distance-bounding was proposed in [129]. It uses a cyclic tree automaton that occupies the full prover's memory. To obtain the erasure proof, the verifier asks the prover to transition into a new state of the automaton, based on random input chosen by the verifier, and to transmit the label of the new state. This round of exchanges is timed by the verifier to bound its distance to the prover. However, we observe that the fact that the response comes fast in each round is not sufficient to counter pre-computation attacks or an adaptive attacker. For example, depending on previous challenges, the adversary can get rid of states in the automaton that are unreachable by future verifier's challenges. This attack was overlooked because the protocol was analysed within a symbolic model, rather than a computational model as we do in Chapters 6 and 8.

As summarized in Table 2.1, there does not exist in the literature a software-based memory-erasure/attestation protocol whose security can be formally proven without relying on the device isolation assumption. Section 6.4 and Section 8.2 in this thesis introduce the first two protocols of this type. We are also not aware of any freely available implementations of these protocols. Some works include performance results, such as [111] and [79], but they are hard to generalize to other environments without further experiments. In Chapter 9 we try to solve these issues by providing open-source implementations for several previous memory-erasure protocols, and also for the new ones proposed in this thesis.

Formal security definitions. We briefly review existing security definitions related to the memory-erasure problem to motivate the need for a new definition in the context of the distant-attacker. The definition of memory attestation in [12] considers an experiment where a computationally unbounded adversary $\mathcal{A}$ produces a computationally bounded prover $\mathcal{P}$ which represents the state of the device to be attested. The protocol is run between a verifier $\mathcal{V}$ and $\mathcal{P}$, where $\mathcal{V}$ aims to ensure that $\mathcal{P}$ is in a desired state; it is deemed secure if a cheating prover would be caught with high probability. This model has built-in the device isolation requirement, since the local

Protocol	No Isol.	No Hw.	Comm.
[121]	✗	✓	$\mathcal{O}(1)$
[83]	✓	✗	$\mathcal{O}(1)$
[111]	✗	✓	$\mathcal{O}(n)$
[109]	✓	✗	$\mathcal{O}(1)$
[13]	✗	✓	$\mathcal{O}(1)$
[81]	✗	✓	$\mathcal{O}(1)$
[79]	✗	✓	$\mathcal{O}(n)$
6.4	✓	✓	$\mathcal{O}(n) + \mathcal{O}(r)$
8.2	✓	✓	$\mathcal{O}(1) + \mathcal{O}(r)$

Table 2.1: Comparison of erasure and attestation protocols in terms of their communication complexity (Comm) and their capacity to *avoid* the use of the device isolation assumption (No Isol.) and specialized hardware (No Hw.). The value of n refers to the memory size and r is a security parameter that refers to the number of rounds during the fast phase (see Section 6.2).

prover $\mathcal{P}$ does not interact with $\mathcal{A}$ throughout the protocol. Since we are interested in memory-erasure and not attestation, in addition to strengthening the attacker model, we may also relax the requirement on the state of the prover, requiring only that it should utilize enough memory at some point in the protocol. The definition of secure memory-erasure in [81] makes this relaxation on the prover state. However, as in [12], the prover $\mathcal{P}$ is assumed computationally bounded, which amounts to the device isolation assumption. In order to lift this assumption, it is not sufficient to simply make $\mathcal{P}$ computationally unbounded, but we need to consider two adversaries $\mathcal{A}$ and $\mathcal{P}$ that can communicate throughout the protocol, and make $\mathcal{A}$'s memory unbounded.

A symbolic Dolev-Yao style model for secure erasure is proposed in [129]. Their main contribution is a formal proof of the necessity of the distant-attacker assumption in some situations and a construction of a protocol that can be proven secure symbolically. However, their symbolic model cannot be used to quantify the size of the adversarial state, or the adversary's probability of success. Hence, it cannot be used to faithfully analyse existing PoSE protocols, which use information theoretical constructions rather than symbolic ones.

Graph based protocols

In a Proof of Space (PoS) protocol, a prover aims to convince a verifier that it has reserved a non-trivial amount of memory space, making it a popular alternative to proof of work. This is similar to memory-erasure, but with the caveat that in PoS the prover may have more memory available, which is clearly undesirable in memory-erasure. It is, nonetheless, useful to understand how PoS is formalized given its similarity with memory-erasure. A PoS, as defined in [58], has two phases: one where the verifier interacts with an attacker $\mathcal{A}_0$, which is supposed to compute and store a state σ_0 of a certain size; another one where a second attacker $\mathcal{A}_1$ takes σ_0 as input and answers the verifier challenges. The verifier then accepts or rejects the proof based on the responses received from $\mathcal{A}_1$. Let σ_1 be the state of maximal size used by $\mathcal{A}_1$. There are two notions of proof of space security defined in [58, 114], one lower-bounding the size of σ_0 and the other lower-bounding the size of σ_1. The definition in [13] only lower-bounds the size of σ_1. These notions are close to what we need, except that they isolate $\mathcal{A}_1$ from $\mathcal{A}_0$ in the second phase, which amounts to the device isolation assumption.

Leaving aside the isolation issue in the security model, a proof of space [13, 58, 114] could in theory be applied to erase memory, since the stored space is typically a random looking sequence of data. However, all current constructions are targeted for a different application scenario, where the prover is a powerful device (e.g. a cloud server) that has more resources at its disposal in order to prove that it has stored the claimed amount of memory. They require $\mathcal{O}(m \log(m))$ memory to prove memory storage of size m. For a proof of erasure, this gap is too high, as it means a large proportion of the memory is not guaranteed to be erased. Memory-hardness results [7] cannot be applied in our case for similar reasons.

A recent series of papers use graph pebbling techniques to achieve time-memory trade-offs in proofs of secure erasure [60, 81]. The first offers a relatively simple proof of security but has a quadratic runtime, while the second one is only able to provably erase $\frac{1}{32}$ of the prover's memory. We should mention, in addition, that [81] also proposes an erasure protocol that is not based on graphs, but on hard-to-invert hash functions. In addition to assuming device isolation, the security of this protocol is proved assuming that, in the memory challenge phase, the adversary cannot query a hash function. We are not aware how this could be enforced in practice.

2.3 Distance-bounding

The earliest distance-bounding protocol was proposed by Brands and Chaum [33]. This protocol achieves optimal security against mafia fraud and distance fraud by cryptographically signing the protocol transcripts after the time-measurement phase. Yet, it is vulnerable to a distance-hijacking attack [45]. With the years it has been improved [15], extending its security goals to the resistance of terrorist fraud [24]. One such improvement is the Swiss-knife protocol [85], whose main features are reaching optimal security against mafia fraud and resisting terrorist fraud to some extent. Terrorist fraud is an exotic type of fraud whereby a dishonest prover helps an adversary to defeat a distance-bounding protocol without allowing the adversary to impersonate him in future sessions of the protocol. The Swiss-knife protocol, while computationally more costly, does not offer better resistance to distance fraud than previous works.

In 2005, Hancke and Kuhn [76] pointed out that using cryptographic signatures, or other expensive primitives, may cause many false failures of the protocol in noisy environments. Furthermore, it could hinder the deployment of distance-bounding on resource-constrained devices, such as RFID tags. They advocated for simplicity and computational efficiency when designing distance-bounding protocols and proposed the well-known Hancke and Kuhn's protocol. Notably, the use of lookup operations during the time-measurement phase improves the precision of the distance estimation and prevents attacks in which the adversary overclocks the prover. Following [96], we refer to protocols adhering to the design principles outlined by Hancke and Kuhn as lookup-based protocols, which are the focus of Chapter 10. For a comprehensive overview of terrorist fraud and other classes of distance-bounding protocols, we refer the interested reader to [15].

Despite the performance virtues of lookup-based protocols, they have struggled to offer strong security against mafia and distance fraud attacks. This is a major drawback, as several scenarios exist where those attacks are of practical significance. For example, Kfir and Wool documented how relay attacks can be executed with NFC devices [82], which was later put into practice in [65]. The abuse of proximity claims through radio channels was first shown by Hancke [75], and later successfully implemented to attack the remote keyless entry system of ten car models from eight different manufacturers [64].

The first lookup-based protocol whose security against mafia fraud is close to the optimum value $\frac{1}{2^n}$, where n is the number of rounds during the fast phase, was introduced by Avoine and Tchamkerten in [20]. The protocol starts with an exchange of nonces between the prover and the verifier. The nonces and a shared secret key are used to secretly agree on the labels of a binary tree of depth n (see Figure 2.1). The labels of such binary tree are encoded within a sequence $T_1T_2 \cdots T_{2^{n+1}-1}$ of size $2^{n+1} - 1$, where T_1 is the label of the root node and, for every $i \in \{1, \ldots, 2^n - 1\}$, T_i's left and right children have labels T_{2i} and T_{2i+1}, respectively. To bound its distance to the prover, the verifier executes n time measurements by sending n binary challenges. Starting from the root node, each binary challenge determines the child node whose label (also binary) is sent as reply to the challenge. The challenge-response game continues recursively until a leaf node is reached. The verifier authenticates the prover if all the responses are correct and arrive on time.

Avoine and Tchamkerten's protocol has a major drawback, though. The size of the tree grows exponentially with the number of RTT measurements. Note that it requires an encoding of size $2^{n+1} - 2$, while Hancke and Kuhn's protocol merely needs a sequence of length $2 \cdot n$. This problem has been treated by subsequent works, notably by [130] and [96], which resorted into different graph structures, rather than trees, to store the outcome of the precomputation phase. Avoine and Tchamkerten themselves offer a trade-off where a linear number trees of depth $d < n$ are used at the cost of downgraded security. Kim and Avoine proposed a different trade-off [84], one which degrades security against distance fraud to improve resistance to mafia fraud. Lastly, the SKI protocol [32] traded resistance to mafia fraud in exchange for some resistance to terrorist fraud. We observe that, while these works managed to find interesting trade-offs between memory size and security, none has solved the problem of finding a lightweight distance-bounding protocol with resistance to mafia fraud and distance fraud equal or close to their optimal values. That is the goal of chapter Chapter 10. We show in Table 2.2 a quick comparison between lookup-based protocols. The first entry for our protocol is marked with an asterisk (*) because its mafia-fraud resistance is optimal, conditional to the optimality of resistance against distance fraud, as will be shown in Sections 10.3 and 10.4.

	Mafia Fraud	Distance Fraud	Space
HK [76]	no	no	linear
Tree Based [20]	yes	no	exponential
Poulidor [130]	no	no	linear
Modular [96]	no	no	linear
Our protocol	yes*	yes	linear

Table 2.2: Comparison of lookup-based protocols with respect to Mafia Fraud resistance optimality, Distance Fraud resistance optimality, and space requirements

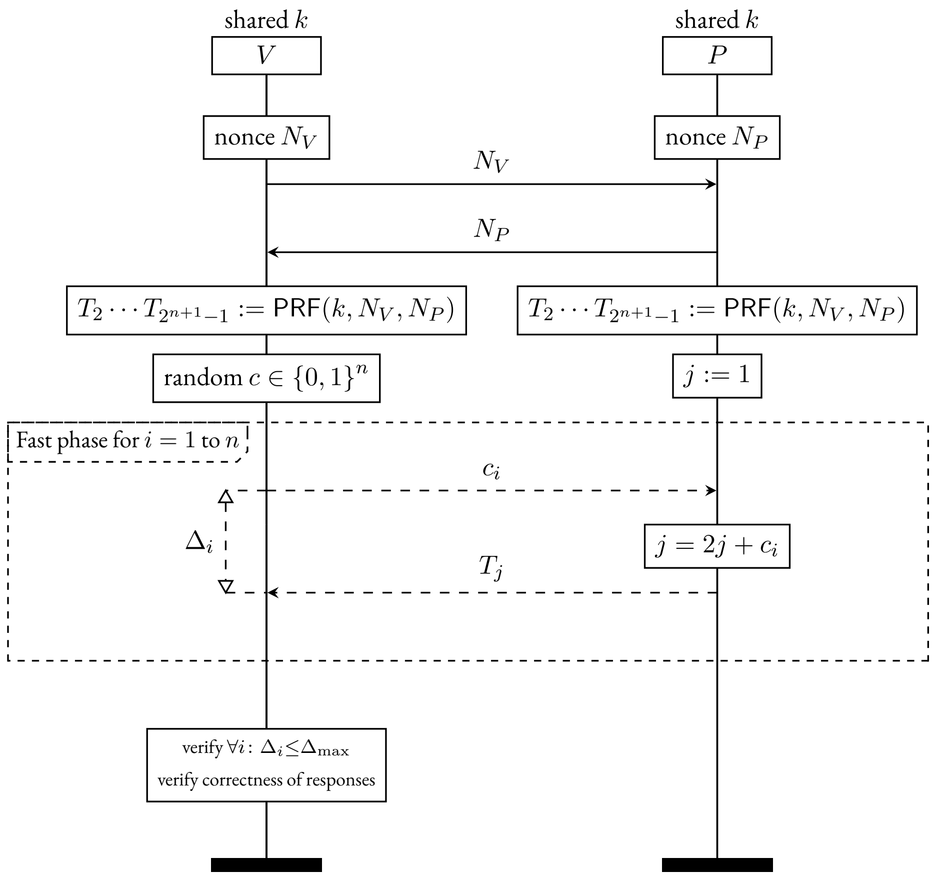

shared k
V
shared k
P
nonce N_V
nonce N_P
N_V
N_P
$T_2 \cdots T_{2^{n+1}-1} := \mathsf{PRF}(k, N_V, N_P)$
$T_2 \cdots T_{2^{n+1}-1} := \mathsf{PRF}(k, N_V, N_P)$
random $c \in \{0,1\}^n$
$j := 1$
Fast phase for $i = 1$ to n
c_i
Δ_i
$j = 2j + c_i$
T_j
verify $\forall i\colon\ \Delta_i \leq \Delta_{\max}$
verify correctness of responses

Figure 2.1: The tree-based protocol by Avoine and Tchamkerten. The tree is encoded in the binary sequence $T_2 \cdots T_{2^{n+1}-1}$.

We conclude by remarking that many security models have been proposed to analyse distance-bounding protocols, such as [32] and [52]. Notably, in [32], Boureanu and Vaudenay study the impossibility of resisting to all types of fraud efficiently. We are interested in a different question, though. Rather than merely resisting an attack, we are interested in *optimally* resisting an attack. For that, we need a dedicated security model particularly tailored to lookup-based protocols, which we introduce in Chapter 10.

3 Background on symbolic analysis

Symbolic models are abstract models that make it easier to build automatic verification tools for the analysis of security protocols [28]. These models idealize the components of the protocol being verified, such as the network, the algorithms executed by the attackers and the protocol participants, and the messages sent. These abstractions have proven successful in practice, as have been shown by attacks found on several popular protocols in the literature. Nowadays, symbolic verification is part of the process of designing new protocols, as it is considered a necessary step in order to achieve strong security guarantees.

In symbolic models, messages are usually modelled as terms from a term algebra. This entails the assumption that cryptographic primitives are perfectly secure, and is one of the main aspects in which these models differ from computational models. By removing the possibility of weak primitives, symbolic models can focus on the interaction between agents executing the protocol. This assumption has proved to be of huge practical importance, because it enabled the automatic formal verification of security protocols.

Several tools have been created to automatically analyse security protocols. The most popular among them, and in chronological order, are FDR [91], Proverif [27], AVISPA [11], Scyther [47], Tamarin [100]. As it happens in many other research fields most tools are short-lived in the sense that they stop being used. Although the reasons vary, in most cases researchers lack the time to maintain the software, or new results/tools appear which make the older ones obsolete. In this sense, from the list mentioned above only two remain highly popular today: Proverif and Tamarin. The first one is already more than two decades old, but has been greatly improved throughout the years. It supports the modelling of protocols using a variant of the applied pi-calculus, and translates the problem of proving security properties to Boolean satisfiability problem (SAT) which then is solved by using custom solutions for the problem. This is a great feature as it allows to eventually improve the performance of the tool by incorporating advances from general SAT solvers. The Tamarin tool has been the result of the collaboration between several groups of researchers during the last ten years. One characteristic that has made Tamarin special is its ability to interactively "guide" the proof. This is needed for the more complicated protocols, and does not affect its fully automated capabilities. Protocols are modelled using a term rewriting formalism, which makes the tool very flexible, although it might feel unnatural for newcomers. It supports several equational theories such as Diffie-Hellman, symmetric and asymmetric signatures, hashing, XOR, and very recently, natural numbers.

A symbolic security model

In this section we present a simple symbolic model based on [45]. We will use this model to showcase the components of standard symbolic model and use it to analyse the famous Needham-

Schroeder protocol and the attack found by Lowe in [90]. The original model includes the main concepts in modern symbolic analysis of protocols and was used as the base for the tool Scyther. Furthermore, it served as inspiration for our model in Chapter 4. Therefore, our aim here is to offer a gentle introduction to readers unfamiliar with the topic.

3.1 Security protocol model

Security protocols models encompass several components: message model, protocol specification, agent model, communication model, attacker model, protocol execution and security requirements. Next we explain each of these components.

The **message model** determines which messages can be sent or received during the protocol. Each message is represented by an element of a recursively defined set. Given a set of primitive messages (constants and fresh values), and a finite set of functions, the set of all possible messages is determined the composition of the functions and the primitive messages. Constants include agent names and other publicly known values, while fresh values represent privately generated random values. Two messages might be equal even if they are not syntactically the same. For example if $\mathsf{Enc}, \mathsf{Dec}$ is a pair of encryption and decryption functions, then for every key k and message m we have $\mathsf{Dec}(\mathsf{Enc}(m, k), k) = m$. These "equational properties" allow expressing algebraic properties of many functions used in common cryptographic protocols.

The **protocol specification** describes how each agent executing the protocol should behave, by defining a finite number of roles the agents execute. Roles are defined through a specification language, which describes which actions are taken by each role in the protocol by a sequence of events, such as sending or receiving a message, or making a security claim. The protocol specification also includes the initial knowledge of each agent, and the declaration of functions, global constants, variables, and fresh values.

The **agent model** describes what agents can do while executing the roles of the protocol. Under the closed world assumption, honest agents strictly follow the behaviour described in the protocol specification. Nevertheless, an agent may execute any number of sessions of the protocol in parallel, acting as potentially different roles. In every session, the agent executes the events described in a role sequentially.

The **communication model** and the **attacker model** are defined by the well-known Dolev-Yao model [55]. The attacker has complete control over the network, where messages are exchanged asynchronously. Messages can be intercepted, modified or constructed by the adversary. These actions are invisible to the agents executing the protocol. When an agent executes a receive event, exactly one message from the network is selected and received by the agent. Similarly, when an agent executes a send event, the associated message becomes accessible in the network, and can be received any number of times by other agents (or attackers).

The **execution** of a protocol is defined by traces of events. Each event must have been executed by an agent following a specific role. A trace implicitly creates a total order of events, and provides a global view of a given protocol execution.

The **security requirements** of a protocol state the purpose of a security protocol. For simplicity, here we consider to reachability properties such as authentication and secrecy. These type of properties can be verified by checking that no trace of the protocol can "reach" (i.e. contain) certain events. More formally, reachability properties can be defined by predicates on the traces of the protocol. A protocol satisfies a security property if the corresponding predicate is valid for every trace of the protocol.

3.2 Needham-Schroeder protocol

We will use the well known Needham-Schroeder protocol to illustrate the components of the symbolic model more formally. Two parties participate in this protocol: the initiator and the responder. The objective of the protocol is to achieve mutual authentication. The message sequence chart (MSC) of the protocol is shown in Figure 3.1.

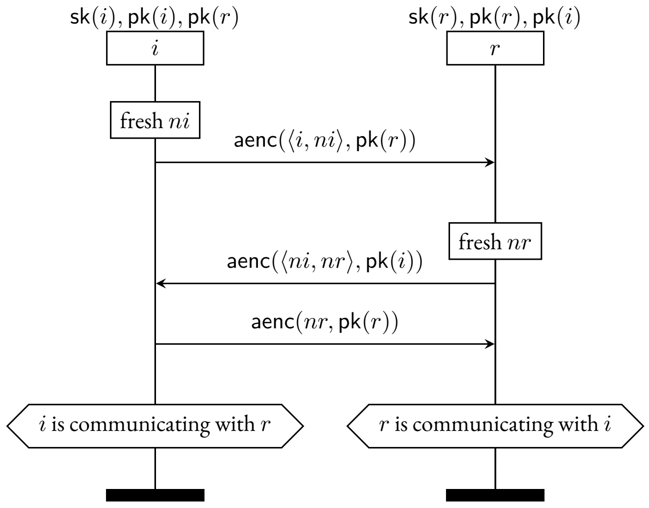


Figure 3.1: Needham-Schroeder public-key authentication protocol

For this example, we consider agent names and fresh values as our set of primitive messages. Recall that agent names are constants that are globally known, and fresh values are initially secret from everyone except the agent that instantiated them. The set of functions includes aenc (asymmetric encryption), $\langle , \rangle$ (pair), pk (public key) and sk (secret key). Valid terms are:

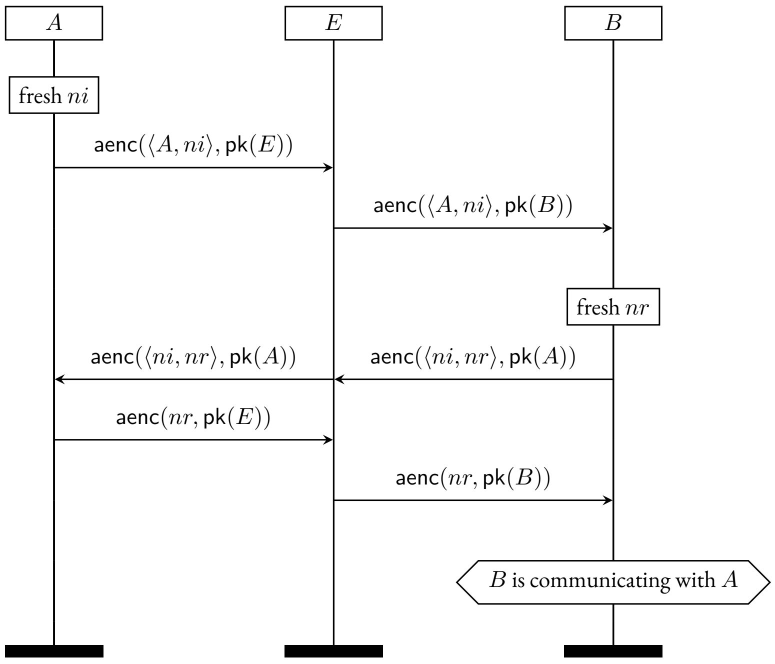
- Agent names: A, B
- Fresh values, which include nonces and keys: ni, nr, k
- Public keys: $\mathsf{pk}(A), \mathsf{pk}(B)$
- Secret keys: $\mathsf{sk}(A), \mathsf{sk}(B)$
- Messages: $\langle A, B\rangle, \mathsf{aenc}(A, \mathsf{pk}(B))$

As mentioned before, the protocol specification defines each role as a sequence of events. The roles for the Needham-Schroeder protocol follow:

$$\begin{aligned}\mathcal{P}_{NS} = \{\text{initiator} =& \text{send}_i(\mathsf{aenc}(\langle i, ni\rangle, \mathsf{pk}(r))) \cdot \text{recv}_i(\mathsf{aenc}(\langle ni, nr\rangle, \mathsf{pk}(i)))\cdot\\ & \text{send}_i(\mathsf{aenc}(nr, \mathsf{pk}(r))) \cdot \text{claim}_i(\text{auth}, r, \mathsf{aenc}(\langle ni, nr\rangle, \mathsf{pk}(i))),\\ \text{responder} =& \text{recv}_r(\mathsf{aenc}(\langle i, ni\rangle, \mathsf{pk}(r))) \cdot \text{send}_r(\mathsf{aenc}(\langle ni, nr\rangle, \mathsf{pk}(i)))\cdot\\ & \text{recv}_r(\mathsf{aenc}(nr, \mathsf{pk}(r))) \cdot \text{claim}_r(\text{auth}, i, \mathsf{aenc}(nr, \mathsf{pk}(r)))\}\end{aligned}$$

There are three types of events, $\text{send}()$, $\text{recv}()$ and $\text{claim}()$. The first two events state which agent made the event in the subindex, and which message (represented by a term) was sent or received. The claim events state which agent made the claim, the type of claim (in this case authentication), and a term related to the validity of the claim. In this protocol, the claim event $\text{claim}_r(\text{auth}, i, \mathsf{aenc}(nr, \mathsf{pk}(r)))$ states that the responder r thinks it has run the protocol with the initiator i. The claim is correct if and only if i sent the message $\mathsf{aenc}(nr, \mathsf{pk}(r))$.

3.3 Attack by Lowe



A
E
B
fresh ni
$\mathsf{aenc}(\langle A, ni\rangle, \mathsf{pk}(E))$
$\mathsf{aenc}(\langle A, ni\rangle, \mathsf{pk}(B))$
fresh nr
$\mathsf{aenc}(\langle ni, nr\rangle, \mathsf{pk}(A))$
$\mathsf{aenc}(\langle ni, nr\rangle, \mathsf{pk}(A))$
$\mathsf{aenc}(nr, \mathsf{pk}(E))$
$\mathsf{aenc}(nr, \mathsf{pk}(B))$
B is communicating with A

Figure 3.2: An attack on the Needham-Schroeder public-key authentication protocol

Several years after the Needham-Schroeder protocol was published, an attack was found by Lowe [90]. The adversarial model included the possibility of an attacker executing the role of initiator, which was not part of the original model. The attack is shown in Figure 3.2 and formally represented by the following trace:

$$\begin{aligned}
&\mathrm{send}_A(\mathsf{aenc}(\langle A, ni\rangle, \mathsf{pk}(E))) \cdot \mathrm{recv}_B(\mathsf{aenc}(\langle A, ni\rangle, \mathsf{pk}(B)))\cdot\\
&\mathrm{send}_B(\mathsf{aenc}(\langle ni, nr\rangle, \mathsf{pk}(A))) \cdot \mathrm{recv}_A(\mathsf{aenc}(\langle ni, nr\rangle, \mathsf{pk}(A)))\cdot\\
&\mathrm{send}_A(\mathsf{aenc}(nr, \mathsf{pk}(E))) \cdot \mathrm{recv}_r(\mathsf{aenc}(nr, \mathsf{pk}(B)))\cdot\\
&\mathrm{claim}_B(\mathrm{auth}, A, \mathsf{aenc}(nr, \mathsf{pk}(B)))
\end{aligned}$$

In the same article, Lowe proposed a fix, shown in Figure 3.3. Notice that it was as simple as including the identity of the responder as part of the second message.

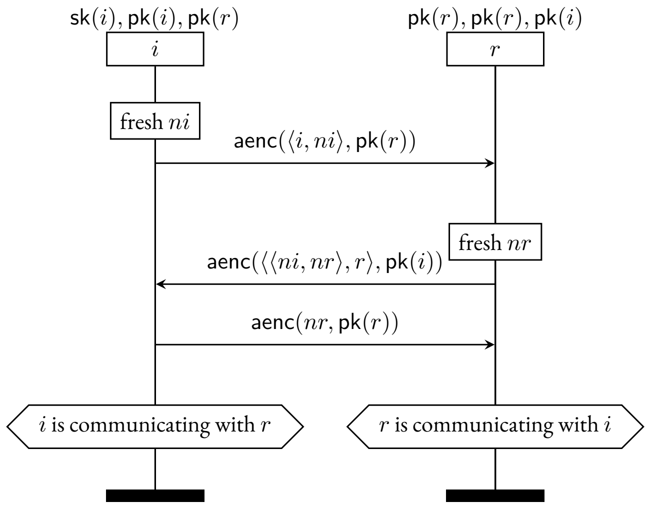

sk(i), pk(i), pk(r)
i
pk(r), pk(r), pk(i)
r
fresh ni
aenc($\langle i, ni\rangle$, pk(r))
fresh nr
aenc($\langle\langle ni, nr\rangle, r\rangle$, pk($i$))
aenc(nr, pk(r))
i is communicating with r
r is communicating with i

Figure 3.3: Needham-Schroeder-Lowe public-key authentication protocol

While showing the existence of an attack is equivalent to finding a single trace that invalidates the security property, proving protocols secure requires demonstrating that no such trace exists. This entails overcoming several sources of infinity: the size of messages and the number of protocol sessions. Even if these two quantities are bounded by constants, the problem of proving protocols secure with respect to simple security properties such as secrecy is known to be undecidable [102]. Therefore, tools such as Tamarin try to prove protocols secure by using heuristics that aim to cover most “interesting” and practical protocols.

4 Analysing protocols under the distant-attacker assumption

Various modern protocols tailored to emerging wireless networks, such as body area networks, rely on the proximity and honesty of devices within the network to achieve their security goals. However, there does not exist a security framework that supports the formal analysis of such protocols, leaving the door open to unexpected flaws. In this chapter we introduce such a security framework, show how it can be implemented in the protocol verification tool Tamarin[1], and use it to find previously unknown vulnerabilities on two recent key-exchange protocols.

4.1 Introduction

In the past few years, we have seen the emergence of a new class of security protocols that have as a common feature that their security goals are only guaranteed under the assumption that the adversary is not in the proximity of the proper communication partners. Examples of such protocols are key-exchange protocols for Body Area Networks [117], pairing protocols of smart devices [123, 135, 139], or protocols for memory-erasure and memory attestation [9, 111].

Assuming that attackers are far or distant, called the *distant-attacker assumption*, can be motivated in various ways. A local attacker can, for instance, be excluded due to physical protection or human observation of the environment. Alternatively, attacks by local agents may be considered infeasible due to the use of out-of-band channels that open to nearby devices only, such as short-range or low-powered communication. A traceability attack, which occurs when a user can be traced based on the transcripts of the communication protocol, that requires the attacker to be close to the victim is arguably ineffective, as the victim is already being physically monitored. Lastly, memory erasure and attestation protocols have proven unable to resist a standard man-in-the-middle attacker [129], such as the Dolev-Yao attacker. The state-of-the-practice for this type of protocol is to isolate the prover and verifier by radio jamming or hardware manipulation.

Most of the protocols that depend on the distant-attacker assumption have not been formally verified yet and may thus suffer from unexpected vulnerabilities. This lack of verification effort can mainly be explained by the use of informal, physical or out-of-band techniques that are hard to formalize in a symbolic security model. Hence, there is a need for a security model that makes explicit what a distant-attacker can and cannot do, and that is amenable to formal verification.

Because the distant-attacker assumption states that the distance between the adversary and the proper communication partners is (much) larger than the mutual distance between the communication partners, this notion appears to have a strong link to the notion of distance-bounding

[1]`https://tamarin-prover.github.io/`

protocols. However, while the goal of distance-bounding protocols is to ensure that the communication partners are close, protocols from the above-mentioned class aim to ensure some classical security property, like secrecy or authentication, under the assumption of proximity of the communication partners. Such focus on distance-bounding is reflected on the verification frameworks [21, 31, 38, 50, 51, 95] developed for the verification of distance-bounding protocols, which ignore classical security properties.

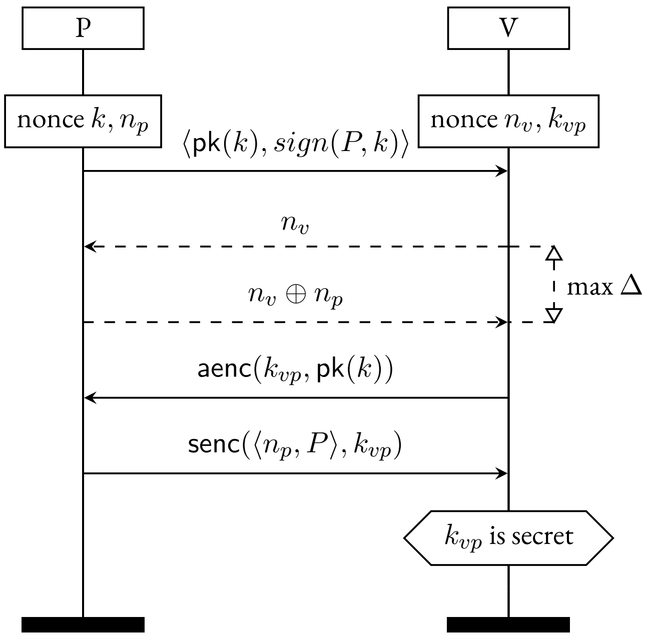

P
V
nonce k, n_p
nonce n_v, k_{vp}
$\langle \mathsf{pk}(k), sign(P,k)\rangle$
n_v
max Δ
$n_v \oplus n_p$
$\mathsf{aenc}(k_{vp}, \mathsf{pk}(k))$
$\mathsf{senc}(\langle n_p, P\rangle, k_{vp})$
k_{vp} is secret

Figure 4.1: An insecure pairing protocol $\mathcal{P}_{ex}$: a running example

Based on our study of the literature, the most common security argument used to analyse this type of protocol is to assume that some messages are unavailable to the attackers, because they are far. Case in point, using the example protocol shown in Figure 4.1: if the messages used to measure RTT can only be received by honest and nearby agents, then the last message $\mathsf{senc}(\langle n_p, P, \rangle, k_{vp})$ should have been generated by an honest agent, supposedly P. By using this security argument one may conclude that this protocol is secure. Yet it is not. The protocol suffers from an attack known as *distance fraud* in the distance-bounding literature [15], because it relies on the ability of an attacker to inject messages from far away.

The attack works as follows. The attacker E executes the protocol with V by sending their own public key. E does not wait for the second message. Instead, E sends a random value m soon enough to be received by V right after V sends n_v. The same antenna that E uses to inject messages is used to eventually receive the verifier's challenge n_v. This allows E to compute $n_p = n_v \oplus m$ and correctly finish the protocol with V. There is another attack on this protocol, that does not require the attacker to interfere with the RTT measurement. It consists in hijacking a session between an honest prover P and V similarly to distance-hijacking attacks on distance-bounding protocols [46]. Without providing further details on the second attack, we argue that the intuitive use of a secure or protected channel between nearby devices to formally model the physical assumptions underlying the protocol, risks missing attacks. Hence, such a modelling could lead to the conclusion that the protocol is safe, while it is not. Consequently, there is a need for a verification methodology that augments standard symbolic security protocol verification with a distant-attacker model and incorporates techniques used for the verification of protocols with physical properties, such as distance-bounding protocols [95]. This chapter introduces such a methodology.

Our methodology starts from the time-based security model which allows for the analysis of standard security properties described in Section 4.2. Next, we add the distant-attacker assumption and round-trip-time restrictions (Section 4.3). In order to prepare for efficient verification with an analysis tool like Tamarin [100], this time-based model is then reduced to a causality-based model (Section 4.4). The input to our methodology consists of the formalized description of a security protocol, in which we modelled the distant-attacker assumption as a time-bound challenge-response loop in the protocol. Using this approach, we formally verified seven protocols for which we found a number of novel attacks (Section 4.5).

4.2 A security model for timed security properties

This section introduces a security model for protocols with RTT restrictions. Like previous models for distance-bounding protocols [21, 49], our model uses a timed communication channel where protocol participants are provided with a spatial location, and the arrival time of messages is consistent with the propagation speed of the communication channel and the distance between sender and receiver. Our model, however, is used to analyse general properties of security protocols. Hence, we provide the model with a simple protocol specification language (à la Cremers and Mauw [45]) and operational semantics that allows us to prove general properties of protocols with RTT measurements in relation to their security goals. We note that although security models including a notion of time do exist, they are either too expressive (e.g. [21]), making it a laborious task to prove general properties of protocols, or too specific (e.g. [95]) and constrained to the analysis of distance-bounding protocols.

4.2.1 Messages

A security protocol defines the way various protocol participants, called *agents*, exchange cryptographic messages. To model them, we use an order-sorted term algebra $(\mathcal{S}, \leq, \mathcal{T}_\Sigma(\mathcal{V}, \mathcal{C}))$ where Σ is a signature, $\mathcal{V}$ a set of variables, $\mathcal{C}$ a set of constants, and $(\mathcal{S}, \leq)$ a partially ordered set (poset) of sorts [71]. We will often use $\mathcal{T}_\Sigma(\mathcal{V}, \mathcal{C})$ to refer to our sorted term algebra when the sorts are clear from context. All terms have a sort. In particular, the sorts agent and nonce are reserved for agent names and nonces. We also define the sort msg (short for message) to be the supersort or the greatest element of the poset $\mathcal{S}$, i.e. $\mathsf{s} \leq \mathsf{msg}$ for all $\mathsf{s} \in S$. We use $t\colon \mathsf{s}$ to denote that term t is of sort s.

Let $\mathsf{Agent} = \{a \in \mathcal{C} \mid a\colon \mathsf{agent}\}$ be the set of agent names and $\mathsf{Nonce} = \{n \in \mathcal{C} \mid n\colon \mathsf{nonce}\}$ be the set of nonces. Given an agent a, we use Nonce_a to denote the set of nonces that agent a can produce. We restrict agents to produce unique nonces, hence we require $\forall a, b \in \mathsf{Agent}\colon a \neq b \implies \mathsf{Nonce}_a \cap \mathsf{Nonce}_b = \emptyset$. The set Agent is partitioned into Honest (honest agents) and $\mathsf{Dishonest}$ (dishonest agents). Finally, we assume that the signature Σ contains the following function symbols:

- $\mathsf{pair}(m, m')$ denoting the pairing of two terms m and m'. We will often use $\langle m, m' \rangle$ as shorthand notation, and write $\langle m_1, \dots, m_n \rangle$ instead of $\langle m_1, \langle m_2, \dots, m_n \rangle \rangle$. The functions fst and snd allow us to recover the first and second element of a pair, respectively.
- $\mathsf{xor}(m, m')$, often written as $m \oplus m'$, denoting the *exclusive or* of the terms m and m'. The constant zero represents the null element with respect to xor.
- $\mathsf{k}(a, b)$ denoting the long-term symmetric key shared by agents a and b.
- $\mathsf{sk}(a)$ denoting the long-term secret key of an agent a.
- $\mathsf{pk}(m)$ denoting the public key associated to a term m. If it does not lead to confusion, we shall write $\mathsf{pk}(a)$ to denote $\mathsf{pk}(\mathsf{sk}(a))$ when $a \in \mathsf{Agent}$.
- $\mathsf{aenc}(m_1, \mathsf{pk}(k))$ and $\mathsf{adec}(m_2, k)$ denoting, respectively, the asymmetric encryption of m_1 with the public key $\mathsf{pk}(k)$, and the asymmetric decryption of m_2 with the secret key k.
- $\mathsf{senc}(m_1, k_1)$ and $\mathsf{sdec}(m_2, k_2)$ denoting, respectively, the symmetric encryption of m_1 with the key k_1, and the symmetric decryption of m_2 with the key k_2.
- $\mathsf{sign}(m, k')$ denoting the signature of m with the secret key k'. The function verify and the constant true are used to verify whether a signature is correct.
- $\mathsf{h}(m)$ denoting the hash of the term m.
- $x \cdot y$ and x^y denote, respectively, the multiplication and exponentiation of x and y, in a Diffie-Hellman group. The function inv and the constant 1 denote, respectively, the inverse function with respect to multiplication and the unit element.

We assume that the sort of all composed terms is the super sort msg.

The semantics of the function symbols above is formalized by an equational theory E that models perfect cryptography, such as the one supported by TAMARIN and ProVerif. We use the symbol $=_E$ to denote equality of two terms modulo E.

Terms with variables will be used in our model to specify the behaviour of protocol participants (roles), in such a way that their behaviour can be *instantiated* multiple times by means of variable substitution. Formally, let $vars\colon \mathcal{T}_\Sigma(\mathcal{V}, \mathcal{C}) \to \mathbb{P}(\mathcal{V})$, where $\mathbb{P}(.)$ denotes the power set, be an auxiliary function that, given a term t, gives all variables occurring in t. A term $t \in \mathcal{T}_\Sigma(\mathcal{V}, \mathcal{C})$ is called *ground* if and only if $vars(t) = \emptyset$. We use $\mathcal{T}_\Sigma(\mathcal{C})$ to denote the set of ground terms over the term algebra. A *substitution* is a function $\sigma\colon \mathcal{V} \to \mathcal{T}_\Sigma(\mathcal{V}, \mathcal{C})$ from variables to terms such that $\sigma(v) \neq v$ for finitely many variables. An *instantiation* of a term t via a substitution σ, denoted $t\sigma$, is inductively defined by

$$t\sigma = \begin{cases} t & \text{if } t \in \mathcal{C} \\ \sigma(t) & \text{if } t \in \mathcal{V} \\ f(\sigma(t_1), \dots, \sigma(t_n)) & \text{if } t = f(t_1, \dots, t_n) \end{cases}$$

We say that a substitution σ is *type-preserving* if for every variable $v \in \mathcal{V}$ it holds that $v\colon \mathsf{s} \wedge \sigma(v)\colon \mathsf{s}' \implies \mathsf{s}' \leq \mathsf{s}$. This means, for example, that a variable of type msg can be substituted by a term of type nonce, but not the other way around. In our model, we consider type-preserving substitutions only, and use Γ to denote the universe of such substitutions.

4.2.2 Protocol specification

We partition a protocol into roles. A role is composed of events it uses to communicate with other roles, security claims, time measurements, etc. An *event* is a term of the form $E_a(t)$, where E is a symbol from an unsorted signature $\mathcal{E}$, and t and a are terms in $\mathcal{T}_\Sigma(\mathcal{V}, \mathcal{C})$ with $a\colon \mathsf{agent}$. The application of a substitution σ to an event $E_a(t)$, denoted $E_a(t)\sigma$, results in the event $E_{a\sigma}(t\sigma)$. The set of all events is denoted Ev and the set $\mathsf{Ev_g} \subseteq \mathsf{Ev}$ denotes the set of *ground events*, which are events with only ground terms as arguments. The function $actor(\cdot)$, defined by $actor(E_a(t)) = a$, provides the actor executing an event.

We reserve the event symbols send, recv, claim, clock, and equal. The events $\mathrm{send}_a(m)$ and $\mathrm{recv}_a(m)$ denote the sending and reception, respectively, of a message m. For the remaining reserved events we impose the following syntactical restrictions: clock events have the form $\mathrm{clock}_a(i, j)$, where i and j are integers representing the start and end of a timer. This timer starts with the execution of the ith event of the role, and stops at the execution of its jth event. Claim events have the form $\mathrm{claim}_a(\psi, t)$, where ψ is a constant denoting a security property name and t is an argument of the property, such as an agent's name or a nonce; equality events have the form $\mathrm{equal}_a(\langle m_1, m_2\rangle)$ denoting the expectation that $m_1 =_E m_2$. The impact of these event types in the behaviour of a protocol will be made precise soon.

As in [45], we consider a role specification R to be a sequence of events $r_1 \cdots r_n$ establishing a total order on the execution of the role events. We require every role specification with sequence of events $r_1 \cdots r_n$ to satisfy that $actor(r_1) = \cdots = actor(r_n)$, i.e. events within a role specification are executed by the same agent. We also require $r_i = \mathrm{clock}_a(x, y) \implies x \leq y < i$ for every $i \in \{1, \ldots, n\}$; i.e. a time measurement is built upon preceding events only, and the event at which the clock stops does not precede the event at which the clock starts.

A role is a mapping from role names, such as *server* and *client*, to role specifications. We use $R = r_1 \cdots r_n$ to denote the role with name R and specification $r_1 \cdots r_n$, and we use R to denote the universe of roles.

Definition 1 (Protocol specification)**.** A protocol $\mathcal{P}$ consists of a set of roles, such that no two roles share the same role name, built over an order-sorted term algebra $(\mathcal{S}, \leq, \mathcal{T}_\Sigma(\mathcal{V}, \mathcal{C}))$.

At the semantic level, we will treat all variables within a role as local variables. This ensures that agents can only communicate by messaging each other. Given a role $R = r_1 \cdots r_n$, we use $rolevars(R)$ to obtain all role variables in R, which is defined as follows.

$$\begin{aligned} rolevars(R) = \{v \in \mathcal{V} \mid \exists i \in \{1, \ldots, n\}\colon r_i = \mathrm{send}_a(m) \wedge \\ v \in vars(m) \wedge \forall j \in \{1, \ldots, i-1\}\colon v \notin vars(r_j)\} \end{aligned}$$

As naming convention for variables, we will use upper case letters when a variable is intended to be instantiated within a receive event, such as K and N_p in the example below, lower case letters otherwise, such as n_v and k_{vp}.

Example 1. To illustrate our security model, we formalize a simple protocol with proximity requirements. The pairing protocol from the previous section (Figure 4.1), inspired by Move2Auth [139], is an example of the type of protocols we are referring to: it aims at secure key exchange, relies on the distant-attacker assumption, and uses the physical properties of the communication channel to check proximity. The goal is for an agent V to create a shared key k_{vp} with another agent P. V does not have any previous cryptographic secret shared with P, but it is confident that all agents in its vicinity are honest. The prover first generates a public/private key pair, denoted $\mathsf{pk}(k)$ and k, respectively. The public key, together with the signature of the prover's identity P with k, is sent to V. Upon reception, V generates a fresh symmetric key k_{vp}, executes a round-trip-time (RTT) measurement with P, and sends k_{vp} encrypted with $\mathsf{pk}(k)$. The RTT measurement, illustrated by dashed arrows, is based on the messages n_v and $n_v \oplus n_p$. Once P decrypts $\mathsf{senc}(k_{vp}, \mathsf{pk}(k))$, it confirms reception of the key by encrypting the nonce n_p and its own identity P with k_{vp}. If the RTT is lower than a time threshold Δ, and $\langle n_p, P\rangle$ is correctly encrypted with k_{vp}, then V concludes that P is nearby and, therefore, honest (based on the distant-attacker assumption). This allows V to claim that k_{vp} is secret, i.e. unknown to attackers.

The specification within our security model of the prover and verifier roles, denoted P and V, respectively, is given below. It assumes that n_v, n_p, k_{vp}, and k are variables of type nonce, while the variables V and P are of type agent. All the other variables, namely N_v, N_p, K_{vp}, S, PK, are of type msg.

$$\begin{aligned}
\mathcal{P}_{\text{ex}} = \{ & \\
& V = \mathrm{recv}_{\mathsf{V}}(\langle PK, S\rangle) \cdot \mathrm{send}_{\mathsf{V}}(n_v) \cdot \mathrm{recv}_{\mathsf{V}}(n_v \oplus N_p) \cdot \\
& \quad \mathrm{clock}_{\mathsf{V}}(2,3) \cdot \mathrm{send}_{\mathsf{V}}(\mathsf{senc}(k_{vp}, PK)) \cdot \mathrm{recv}_{\mathsf{V}}(\mathsf{senc}(\langle N_p, P\rangle, k_{vp})) \cdot \\
& \quad \mathrm{equal}_{\mathsf{V}}(\langle \mathsf{verify}(S, P, PK), \mathsf{true}\rangle) \cdot \mathrm{claim}_{\mathsf{V}}(sec, \langle P, k_{vp}\rangle), \\
& P = \mathrm{send}_{\mathsf{P}}(\langle \mathsf{pk}(k), \mathsf{sign}(P, k)\rangle) \cdot \mathrm{recv}_{\mathsf{P}}(N_v), \mathrm{send}_{\mathsf{P}}(N_v \oplus n_p) \cdot \\
& \quad \mathrm{recv}_{\mathsf{P}}(\mathsf{senc}(K_{vp}, \mathsf{pk}(k))) \cdot \mathrm{send}_{\mathsf{P}}(\mathsf{senc}(\langle n_p, P\rangle, K_{vp})) \\
& \}
\end{aligned}$$

The Δ symbol labelling a dashed arrow that connects the 2nd and 3rd protocol message in Figure 4.1 represents the round-trip-time measurement of the verifier, and is translated into the event $\mathrm{clock}_{\mathsf{V}}(2,3)$ in the role specification.

4.2.3 Role instantiation.

Protocols are executed by instantiating their roles. Syntactically, the instantiation of a role results in a sequence of ground events that respect the order of the events established by the role specification. Given a role $R = r_1 \cdots r_n$, we define all its instantiations, denoted $insts(R)$, as follows.

$$insts(R) = \{e_1 \cdots e_i \in \mathsf{Ev_g}^* \mid i \leq n \wedge \exists \sigma \in \Gamma \colon e_1 = r_1\sigma, \ldots, e_i = r_i\sigma\}$$

Note that the empty sequence of events ϵ is a valid instantiation of all role specifications.

In the operational semantics provided further below, we restrict role variables of type nonce to be assigned a *fresh* value, i.e. a term that has not been used in other role instantiations during

$$\frac{a \vdash_s m \quad m =_E m'}{a \vdash_s m'}\ \texttt{I0} \qquad \frac{m \in \mathsf{Agent} \cup \mathsf{Nonce}_a}{a \vdash_s m}\ \texttt{I1}$$

$$\frac{b \in \mathsf{Agent}}{a \vdash_s \langle \mathsf{k}(a,b), \mathsf{k}(b,a), \mathsf{sk}(a), \mathsf{pk}(\mathsf{sk}(b))\rangle}\ \texttt{I2}$$

$$\frac{a \vdash_s m_1 \quad \ldots \quad a \vdash_s m_n \quad f \in \Sigma \setminus \{\mathsf{sk}, \mathsf{k}\}}{a \vdash_s f(m_1, \ldots, m_n)}\ \texttt{I3}$$

$$\frac{(t, \mathrm{recv}_a(m)) \in s}{a \vdash_s m}\ \texttt{I4}$$

Figure 4.2: Inference rules

the protocol execution. Therefore, we introduce an auxiliary function to extract the set of nonces used in an instantiation, for every $e_1 \cdots e_i \in insts(R)$,

$$\begin{aligned} nonces(e_1 \cdots e_i, R) = \{t \in \mathsf{Nonce} \mid \exists v \in rolevars(R), \sigma \in \Gamma \colon \\ \sigma(v) = t \wedge v \colon \mathsf{nonce} \wedge e_1 = r_1\sigma, \ldots, e_i = r_i\sigma\} \end{aligned}$$

4.2.4 Inference

Before establishing how agents exchange messages, we need to define how agents obtain and create knowledge. We model this by means of an inference relation $\vdash \subseteq \mathsf{Agent} \times \mathbb{P}(\mathsf{Ev}) \times \mathsf{Msg}$. We use the shorthand notation $a \vdash_s m$ to denote $(a, s, m) \in \vdash$, indicating that agent a can infer message m from a set of events s. The relation $\vdash$ is defined as the least set that is closed under the inference rules in Figure 4.2. These rules express the following:

(I0) if an agent a can infer a term m, then a can infer all terms within the equivalence class of m defined by $=_E$

(I1) an agent a can infer agent names and its own set of nonces

(I2) agents can infer their shared secret keys with other agents, long term public keys of any agent and its own long term secret key

(I3) all function symbols in Σ, except the reserved symbols for secret keys k and sk, can be used to infer arbitrary terms constructed over already inferable terms

(I4) a receive event $\mathrm{recv}_a(m)$ allows agent a to infer m

4.2.5 Protocol semantics

We model the execution of a protocol $\mathcal{P}$ as a Labelled Transition System (LTS) $(Q, \Lambda, \rightarrow, s_0)$, where Q is a set of states, Λ is a set of labels used to annotate the transition of the system from one

state to another, $\rightarrow : Q \times \Lambda \times Q$ is a transition relation, and $s_0 \in Q$ is the initial state. We will often use $s \xrightarrow{\ell} s'$ as a shorthand notation for $(s, \ell, s') \in \rightarrow$.
States. A state in the system is composed of a set of *runs*, where a run is an instantiation of a role by an agent. A run contains a possibly partial execution of a role and a run identifier. The latter allows agents to play multiple roles in parallel. A run also includes the time-stamps at which the events are executed. This allows us to reason about time properties, such as consistency between the time-of-flight of a transmitted message with respect to the speed of the communication channel and the location of sender and receiver.

Definition 2 (Run)**.** Let Id be a countably infinite set of run identifiers. A *run* is any tuple $(id, (t_1, e_1) \cdots (t_n, e_n), R, a) \in \mathsf{Id} \times (\mathbb{R}^+ \times \mathsf{Ev_g})^* \times \mathsf{R} \times \mathsf{Agent}$ satisfying that $e_1 \cdots e_n \in insts(R)$ is an instantiation of the role R, a is the actor executing the events $e_1, \ldots, e_n$, i.e. $a = actor(e_1) = \cdots = actor(e_n)$, and $t_1 \leq t_2 \leq \cdots \leq t_n$.

A run is an execution of a role by an actor, containing instantiations of events in the order imposed by the role specification, and timestamps indicating the time at which each event was instantiated. The order of the timestamps $\leq$ is consistent with the order of the events in the instantiated role. An *empty run* contains no events.

A *state* in our LTS is a set of runs. The *initial state* s_0 is the empty set. Given a state s, the functions $labels(s)$ and $nonces(s)$ return, respectively, all timed events and fresh values occurring in s.

$$\begin{aligned} labels(s) = &\{(t, e) \mid \exists (id, (t_1, e_1) \cdots (t_n, e_n), R, a) \in s : \\ &(t, e) = (t_i, e_i) \text{ for } i \in \{1, \ldots, n\}\} \end{aligned}$$

$$nonces(s) = \bigcup_{(id, (t_1, e_1) \cdots (t_n, e_n), R, a) \in s} nonces(e_1 \cdots e_n, R)$$

Given a state s, a role R, and a substitution σ mapping the role R to its instantiation $e_1 \cdots e_n$, we use $e_1 \cdots e_n \in_s insts(R)$ to denote that the instantiation $e_1 \cdots e_n$ of R satisfies:

1. $nonces(e_1 \cdots e_i, R) \cap nonces(s) = \emptyset$
2. for every $x, y \in rolevars(R)$ of type nonce, $x \neq y$ implies $\sigma(x) \neq \sigma(y)$

That is, no instantiation in s exists sharing fresh values with the instantiation $e_1 \cdots e_n$ of R, and the role variables are instantiated with pairwise distinct nonces.
Labels and execution traces. A transition in our LTS will either add a new empty run to the current state or add a timed event to an existing run. Each transition is labelled with a timestamp, a description of the state update and the id of the run modified in that transition (this will become clear later). We denote the creation of a new run by $\mathrm{create}_a(R)$, where $a \in \mathsf{Agent}$ is an agent and $R \in \mathsf{R}$ is a role. The addition of a protocol event will be labelled by using the events themselves as labels. Therefore, an *execution* of the protocol is an interleaved sequence of states and labels of the type $s_0 \xrightarrow{(t_1, l_1)^{id_1}} s_1 \cdots s_{n-1} \xrightarrow{(t_n, l_n)^{id_n}} s_n$, where $s_0, \ldots, s_n$ are states, $t_1, \ldots, t_n$ timestamps,

and l_i is either a protocol event or a label of the type $\text{create}_a(R)$, for $i \in \{1, \ldots, n\}$. A *trace* is the resulting sequence of LTS labels $\tau = (t_1, \ell_1)^{id_1} \cdots (t_n, \ell_n)^{id_n}$. In this case, we say that τ has cardinality n, denoted $|\tau|$, and we use τ_i to denote the ith element of τ, i.e. $\tau_i = (t_i, \ell_i)^{id_i}$. When $n = 0$, we use ϵ to denote the empty trace, while the initial state s_0 alone represents the empty execution. Lastly, we will omit the run ids from traces when they are not necessary.

We write $a \vdash_\tau m$ to denote $a \vdash_s m$ where s is the set of events occurring in τ.

Transition relation. The transition relation of the protocol LTS is defined by a set of derivation rules, similar to the inference rules above. The premise of a rule is a formula with no free variables. Its conclusion is a transition of the form $s \xrightarrow{(t,\ell)^{id}} s'$. Unless otherwise specified, variables in our derivation rules are universally quantified. For run identifiers, we use $id \in_s \mathsf{Id}$ to denote that id is chosen from the set Id in such a way that id has not been used in the state s. The function $\text{max\_time}\colon Q \to \mathbb{R}^+$ gives the maximum timestamp used by an event in a given state. If $\mathit{labels}(s) = \emptyset$, then $\text{max\_time}(s) = 0$, otherwise $\text{max\_time}(s) = \max\{t \mid (t, e) \in \mathit{labels}(s)\}$.

Our first rule adds an empty run to a state by instantiating either a protocol role or an adversarial role.

$$\frac{\begin{array}{c} s \in Q, t \geq \text{max\_time}(s), R \in \mathsf{R}, a \in \mathsf{Agent}, \\ id \in_s \mathsf{Id}, a \in \mathsf{Honest} \implies R \in \mathit{roles}(\mathcal{P}) \end{array}}{s \xrightarrow{(t, \text{create}_a(\mathcal{P}(R)))^{id}} s \cup \{(id, \epsilon, R, a)\}} \ \mathsf{Create}_{\mathcal{P}}$$

Here $\mathcal{P}$ denotes the protocol specification whose behaviour is being modelled. This rule ensures that honest agents instantiate a protocol role, i.e. that honest agents do not deviate from the protocol specification. Dishonest agents, on the other hand, can influence the protocol execution by instantiating arbitrary role specifications, which we refer to as an adversarial role.

The remaining transition rules express how a run of the system state can make progress by executing a single event. There are three of these rules, which can all be defined as an extension of the following rule template:

$$\frac{\begin{array}{c} s \in Q, t \geq \text{max\_time}(s), (id, \tau, R, a) \in s, \\ \tau = (t_1, e_1) \cdots (t_i, e_i), \\ e_1 \cdots e_i e_{i+1} \in_{s \setminus (id, \tau, R, a)} \mathit{insts}(R), \\ [\,] \end{array}}{s \xrightarrow{(t, e_{i+1})^{id}} (s \setminus \{(id, \tau, R, a)\}) \cup \{(id, \tau \cdot (t, e_{i+1}), R, a)\}},$$

where the bracketed part [ ] is a placeholder to add premises. This rule template triggers the execution of an event e_{i+1} at time t if t is greater than or equal to the largest timestamp in s, and there exists a run $(id, (t_1, e_1) \cdots (t_i, e_i), R, a)$ in the state satisfying that $e_1 \cdots e_{i+1}$ is an instantiation of R. The condition $e_1 \cdots e_i e_{i+1} \in_{s \setminus (id, \tau, R, a)} \mathit{insts}(R)$ ensures that nonces assigned to role variables in other runs, i.e. in s minus (id, τ, R, a), are not used in the current run.

We use $\rightarrow [p_1, \ldots, p_n]$ to denote the rule obtained from the above rule template by replacing the placeholder [ ] by the set of premises $\{p_1, \ldots, p_n\}$. Using this notation, we define the remaining rules for our LTS in the following.

The Send rule:

$$\to \left[e_{i+1} = \mathrm{send}_a(m), a \vdash_{\{e_1,\ldots,e_i\}} m\right] \qquad \mathsf{Send}$$

allows agents to send messages to the network. The Send rule restricts both honest and dishonest agents to send messages whose content is inferable from their initial knowledge, constants, and the sequence of events already executed in the run. This is expressed by the premise $a \vdash_{\{e_1,\ldots,e_i\}} m$, and means that our model does not consider an omnipresent adversary overseeing all events sent to the network. Instead, our model forces dishonest agents to collaborate by messaging each other.

The $\mathsf{Recv}_{\mathrm{d}}$ rule:

$$\to \begin{bmatrix} e_{i+1} = \mathrm{recv}_a(m), (t', \mathrm{send}_b(m')) \in \mathit{labels}(s), \\ \mathrm{d}(a,b) \leq \mathsf{c}(t-t'), m =_E m' \end{bmatrix} \qquad \mathsf{Recv}_{\mathrm{d}}$$

models how agents receive messages from other agents, enforcing that the time of flight of a message exchange is consistent with the distance between sender and receiver. The rule is parametrized on the function $\mathrm{d} \colon \mathsf{Agent} \times \mathsf{Agent} \to \mathbb{R}^+$ which defines a metric space $(\mathsf{Agent}, \mathrm{d})$, and assumes a constant propagation speed c of the communication channel.

The rule $\mathsf{Recv}_{\mathrm{d}}$ triggers the execution of a receive event $\mathrm{recv}_a(m)$ at time *t* if $e_{i+1} = \mathrm{recv}_a(m)$ and there exists a timed send event $(t', \mathrm{send}_b(m'))$, where $m =_E m'$, in some run in the state such that the distance between the sender and receiver is smaller than or equal to $\mathsf{c}(t-t')$.

It is worth pointing out that the $\mathsf{Recv}_{\mathrm{d}}$ rule does not consider that messages may fade away as they travel, implying that secrets revealed to nearby agents leak to the entire network. This is a deliberate choice made with the goal of making no assumptions about signal strength, nor about the distance at which a message can be eavesdropped. For example, RFID eavesdropping on messages at a range of 20 m or more has proven feasible, depending on the power of the devices [113].

The Equal rule

$$\to \begin{bmatrix} e_{i+1} = \mathrm{equal}_a(\langle m_1, m_2\rangle), m_1 =_E m_2, \\ a \vdash_{\{e_1,\ldots,e_i\}} m_1, a \vdash_{\{e_1,\ldots,e_i\}} m_2 \end{bmatrix} \qquad \mathsf{Equal}$$

states that an event $\mathrm{equal}_a(\langle m_1, m_2\rangle)$ only executes when m_1 and m_2 are equal modulo the equational theory E. This type of event is used, e.g., to model the verification of signatures.

The Signal rule

$$\to [e_{i+1} \in \mathsf{SignalEvent}] \qquad \mathsf{Signal}$$

models the execution of *signal* events. Signal events are useful for instrumenting security properties, which often rely on expectations announced by agents by means of signal events, such as claim events. Formally, the set of signal events is defined by $\mathsf{SignalEvent} = \{e \in \mathsf{Ev_g} \mid \nexists a, m \colon e \in \{\mathrm{send}_a(m), \mathrm{recv}_a(m), \mathrm{equal}_a(m)\}\}$.

Definition 3 (Protocol semantics)**.** The semantics of a protocol $\mathcal{P}$ with respect to a distance function d is the LTS $(Q, \Lambda, \to, s_0)$ where,

- $Q = \mathbb{P}(\mathsf{Id} \times (\mathbb{R}^{+} \times \mathsf{Ev_g})^{*} \times \mathsf{R} \times \mathsf{Agent})$
- $\Lambda = \mathbb{R}^{+} \times (\mathsf{Ev_g} \cup \{\mathrm{create}_a(R) \mid a \in \mathsf{Agent}, R \in \mathsf{R}\}) \times \mathsf{Id}$
- $\rightarrow = \mathsf{Create}_{\mathcal{P}} \cup \mathsf{Send} \cup \mathsf{Recv_d} \cup \mathsf{Equal} \cup \mathsf{Signal}$.
- $s_0 = \emptyset$

We use $[\![\mathcal{P}]\!]_{\mathrm{d}}$ to denote the set of traces obtained from $\mathcal{P}$'s semantics with respect to the distance function d.

Example 2. Let E be an adversarial role specification with all role variables of type nonce:

$$\begin{aligned} E = {} & \mathrm{send}_e(\langle \mathsf{pk}(k_e), \mathsf{sign}(P, k_e)\rangle) \cdot \mathrm{recv}_e(N_v) \cdot \\ & \mathrm{send}_e(N_v \oplus n_e) \cdot \mathrm{recv}_e(\mathsf{aenc}(K_{vp}, \mathsf{pk}(k_e))) \cdot \mathrm{send}_e(\mathsf{senc}(\langle n_e, P\rangle, K_{vp})) \end{aligned}$$

Assuming that the pairwise distance between the agents a, b and c is $100 \cdot \mathsf{c}$ (a hundred times the speed of light), and that $s = \mathsf{sign}(b, k')$, an execution trace of our running example protocol $\mathcal{P}_{ex}$ is the following:

$$\begin{aligned} & s_0 \xrightarrow{(0,\mathrm{create}_a(id1,\mathcal{P}_{\mathrm{ex}}(V)))} s_1 \xrightarrow{(0,\mathrm{create}_b(id2,\mathcal{P}_{\mathrm{ex}}(P)))} s_2 \\ & \xrightarrow{(0,\mathrm{create}_e(id3,E))} s_3 \xrightarrow{(0,\mathrm{send}_e(\langle \mathsf{pk}(k'),s\rangle))} s_4 \\ & \xrightarrow{(100,\mathrm{recv}_a(\langle \mathsf{pk}(k'),s\rangle))} s_5 \xrightarrow{(100,\mathrm{send}_a(n'))} s_6 \xrightarrow{(200,\mathrm{recv}_e(n'))} s_7 \\ & \xrightarrow{(200,\mathrm{send}_e(n' \oplus n))} s_8 \xrightarrow{(300,\mathrm{recv}_a(n' \oplus n))} s_9 \xrightarrow{(300,\mathrm{clock}_a(2,3))} s_{10} \\ & \xrightarrow{(300,\mathrm{send}_a(\mathsf{aenc}(k'',\mathsf{pk}(k'))))} s_{11} \xrightarrow{(400,\mathrm{recv}_e(\mathsf{aenc}(k'',\mathsf{pk}(k'))))} s_{12} \\ & \xrightarrow{(400,\mathrm{send}_e(\mathsf{senc}(\langle n,b\rangle,k'')))} s_{13} \xrightarrow{(500,\mathrm{recv}_a(\mathsf{senc}(\langle n,b\rangle,k'')))} s_{14} \\ & \xrightarrow{(500,\mathrm{equal}_a(\langle \mathsf{verify}(s,b,\mathsf{pk}(k')),\mathsf{true}\rangle))} s_{15} \xrightarrow{(500,\mathrm{claim}_a(sec,\langle b,k''\rangle))} s_{16} \end{aligned}$$

4.2.6 Security properties, claims, and protocol correctness

We define a security property ψ as a predicate on traces and integers such that $\psi(\tau, i)$ means that ψ is satisfied at step i of the trace τ. For illustration purposes, we define next the secrecy property used in our running example, whereby an agent expects the adversary to not know a given term.

$$\begin{aligned} & sec((t_1, e_1), \ldots, (t_n, e_n), i) \iff \\ & \Big(\quad e_i = \mathrm{claim}_a(sec, \langle b, m\rangle) \wedge b \in \mathsf{Honest} \implies \nexists c \in \mathsf{Dishonest}\colon c \vdash_{\{e_1,\ldots,e_n\}} m \Big) \end{aligned}$$

As in this example, we consider properties instrumented by claim events. A security claim denotes a belief about the protocol execution that led to the claim, e.g. $\mathrm{claim}_a(sec, \langle b, m\rangle)$ denotes a's belief that as long as its communicating partner is honest, no adversary knows the secret term m. Note the slight abuse of notation in the definition of secrecy: sec is used as a claim identifier and a predicate symbol. We shall keep this convention from now on to make the connection between claims and their intended meaning explicit. That is, for every claim event $\mathrm{claim}_a(\psi, m)$, we consider ψ to be the predicate giving ψ its meaning. Hence, the following definition of protocol correctness with respect to a security claim follows.

Definition 4 (Claim correctness)**.** A claim event $\mathrm{claim}_a(\psi, m)$ is said to be correct in a protocol $\mathcal{P}$, denoted $\mathcal{P} \triangleright \psi$, if for every distance function d, trace $\tau \in [\![\mathcal{P}]\!]_{\mathrm{d}}$, and index $i \in \{1, \ldots, |\tau|\}$, $\psi(\tau, i)$ holds.

In the next section we extend the definition of claim correctness with time restrictions and assumptions about the honesty of agents in relation to their distance to another agent, making it possible for $\mathcal{P}_{ex}$, and several other modern protocols, to formalize their security goals.

4.3 Modelling security requirements based on the distant-attacker assumption

The formal model introduced in the previous section can be used to specify the structure and behaviour of a security protocol with time measurements. This section introduces a new class of security requirements that captures the intended security goals of a relatively recent wave of protocols based on proximity [9, 35, 117, 124, 129, 135, 139]. Such protocols aim at classical security properties, such as secrecy and authentication, but rely on a particular trust assumption that has not been captured within a formal security model. We are referring to trust assumptions that are based on the honesty of agents in the neighbourhood of another agent, usually a verifier. We call this assumption the *distant-attacker assumption*, as it considers the neighbourhood of a verifier to be free of attackers.

The goal of this section is to formalize the distant-attacker assumption and instrument it within classical security properties. The result is a class of security requirements expressed as a premise-conclusion formula, where the premise is a proximity check in conjunction with a distant-attacker claim, and the conclusion is a standard trace-based property. In the subsequent sections, we show such a class of security requirements to be sufficient for the verification of many security protocols based on proximity which currently lack formal correctness proofs.

4.3.1 The distant-attacker assumption

The security goals of many modern protocols are contingent upon the assumption that no attacker is in the vicinity of the verifier. When formalizing this assumption, we shall allow the verifier's communicating partner (the prover) to be both malicious and close, as it allows us to model memory attestation and erasure properties [129] for free. Hence, our task next is to formalize the following statement: *agents making a security claim are aware of what attackers (if any) are in their vicinity.*

We define the vicinity of an agent as the locus of a circle with radius δ centred in the agent's location. We also use the auxiliary function $\mathsf{dishonest\_agents}(\mathrm{claim}_a(\psi, m))$ which gives the set of dishonest agents that a allows to be close when claiming the property ψ. For the case of secrecy, for example, it follows that $\mathsf{dishonest\_agents}(\mathrm{claim}_a(sec, \langle b, m\rangle)) = \emptyset$ for every b and m. This is, indeed, the case for authentication properties, as the verifier does not expect to authenticate a corrupt prover. In remote memory-erasure and attestation protocols, however, the prover is considered dishonest, implying that the function $\mathsf{dishonest\_agents}(\cdot)$ should return the prover agent for erasure and attestation claims. This leads to the following formalization of the distant-attacker assumption as a predicate with domain $\mathcal{T} \times \mathbb{Z}^+$, where $\mathcal{T}$ is the universe of traces. For every $\tau = (t_1, e_1) \cdots (t_n, e_n)$, distance threshold δ and index i,

$$\begin{aligned}&\mathsf{dist\_attacker}_\delta(\tau, i) \iff e_i = \text{claim}_a(\psi, m) \wedge \\ &\quad \forall c \in (\mathit{actors}(\tau) \cap \mathsf{Dishonest}) \colon (\text{d}(a, c) > \delta \vee c \in \mathsf{dishonest\_agents}(e_i))\end{aligned}$$

The distant-attacker assumption holds for a claim $e_i = \text{claim}_a(\psi, m)$ in a trace

$$\tau = (t_1, e_1) \cdots (t_n, e_n)$$

if all dishonest agents in the trace are either far from a, i.e. at a distance larger than δ, or are part of a list of expected dishonest agents $\mathsf{dishonest\_agents}(e_i)$.

4.3.2 Round-trip-time restrictions

Intuitively, for a protocol to use the distant-attacker assumption effectively, it needs to provide agents with the ability to measure distance to other agents. Our security model allows protocols to accomplish this by means of clock events, syntactically establishing the calculation of the time difference between two events. Semantically, time measurements are local to protocol runs. That is, any two events involved in the calculation of a time measurement should be part of the same protocol session or run. We thus need a mechanism to extract runs from traces, which we obtain by exploiting the run identifiers present in protocol traces. Given a trace $\tau = (t_1, e_1)^{id_1} \cdots (t_n, e_n)^{id_n}$ and run identifier id, we use $run(\tau, id)$ to denote the subtrace $(t_{i_1}, e_{i_1})^{id} \cdots (t_{i_k}, e_{i_k})^{id}$ of maximum cardinality, i.e. $run(\tau, id)$ denotes the full run in τ with run identifier id.

Here we formalize an interpretation of time measurements in relation to a security claim and the distant-attacker assumption, leading to a definition of correctness that we show is applicable to a large class of protocols. Clock events are used to measure the round-trip-time of a message exchange with a communicating partner, with the expectation that the communicating partner is within a δ radius, for some distance parameter δ. If a time measurement is below $\frac{2\delta}{\mathsf{c}}$ where c is the speed of the communication medium, then we say that such measurement is correct with respect to the distance bound δ. If all clock events in a given protocol run are correct, then we say that such a run has correct time measurements. Formally, the correctness of the time measurements of the run $run(\tau, id_i) = (t_{i_1}, e_{i_1}) \cdots (t_{i_k}, e_{i_k})$ in the trace $\tau = (t_1, e_1)^{id_1} \cdots (t_n, e_n)^{id_n}$, where e_i is a claim event by agent a is defined by:

$$\begin{aligned}&\mathsf{correct\_time}_\delta(\tau, i) \iff \forall j \in \{i_1, \ldots, i_k\} \colon \\ &\quad (e_j = \text{clock}_a(x, y) \wedge j < i) \implies t_{i_y} - t_{i_x} < \frac{2\delta}{\mathsf{c}}\end{aligned}$$

Note that in the definition of $\mathsf{correct\_time}_\delta(\tau, i)$ we only consider clock events that precede the event e_i. The predicate ensures that for every possible run of the protocol resulting in the trace

τ, the time measurements performed by the agent that produced the event e_i are all below the threshold $\frac{2\delta}{\mathsf{c}}$.

Now we are ready to define a class of security requirements that extend classical security properties by making them conditional to round-trip-time restrictions and the distant-attacker assumption.

Definition 5 (Claim correctness under the distant-attacker assumption)**.** Let $\mathcal{P}$ be a protocol and δ a distance value. A claim event $\mathrm{claim}_a(\psi, m)$ is said to be correct in $\mathcal{P}$ under the distant-attacker assumption, denoted $\mathcal{P} \triangleright_\delta \psi$, if for every distance function d, trace $\tau \in [\![\mathcal{P}]\!]_{\mathrm{d}}$, and index $i \in \{1, \dots, |\tau|\}$,

$$\mathsf{correct\_time}_\delta(\tau, i) \wedge \mathsf{dist\_attacker}_\delta(\tau, i) \implies \psi(\tau, i)$$

As a security requirement, correctness with respect to $\triangleright_\delta$ is weaker than correctness with respect to $\triangleright$ (see Definition 4), i.e. for every protocol $\mathcal{P}$, every claim $\mathrm{claim}_a(\psi, m)$, and every $\delta > 0$, it follows that $\mathcal{P} \triangleright \psi \implies \mathcal{P} \triangleright_\delta \psi$. That said, we argue that the relation $\triangleright_\delta$ better captures the intended goal of many other modern protocols, including the running example used in this section. Therefore, we dedicate the remainder of this section to introducing a verification framework for proving the correctness of protocols with respect to $\triangleright_\delta$.

4.4 A causality-based interpretation of locality

A security model that explicitly carries the notion of time and location, as introduced in the previous chapter, is useful to reach consensus on the formal definition of the distant-attacker assumption and how to instrument it as a security requirement. However, it defies computer-aided reasoning since modern protocol verification tools, such as Tamarin and ProVerif, do not cope well with temporal and spatial information.

Our goal in this section is to provide a timeless protocol semantics, denoted $[\![\cdot]\!]_\pi$, and a correctness relation of protocols with respect to a security property ψ, denoted $[\![\mathcal{P}]\!]_\pi \triangleright_\sim \psi$ where $\sim$ is an equivalence relation on the set of agents that encodes proximity, such that $\forall\, \mathrm{d} \colon [\![\mathcal{P}]\!]_{\mathrm{d}} \triangleright_\delta \psi \iff \forall \sim \colon [\![\mathcal{P}]\!]_\pi \triangleright_\sim \psi$ is valid. This will allow us to analyse the property ψ on the timeless semantics by considering $\triangleright_\sim$ to be the security goal, rather than $\triangleright_\delta$.

4.4.1 Properties of the timed semantics

We start by proving properties of the time-based semantics introduced in Section 4.2.

The relation defined in the following definition is essential for determining when a trace can be generated by our timed semantics, as it represents a precedence relation between events in the trace.

Definition 6. Given a trace $\tau = (t_1, e_1) \cdots (t_n, e_n) \in [\![\mathcal{P}]\!]_{\mathrm{d}}$, let $\rightsquigarrow_\tau$ be the relation defined as follows:

$(t_i, e_i) \rightsquigarrow_\tau (t_j, e_j)$ if and only if $i < j$ and either:

- $actor(e_i) = actor(e_j)$ or

- $e_i = \mathrm{send}_a(m_i) \land e_j = \mathrm{recv}_b(m_j) \land a \neq b \land m_i =_E m_j \land \mathrm{d}(a,b) \leq (t_j - t_i) \cdot \mathsf{c} \land \neg(\exists k \in \{1,\dots,n\}, \exists m_k \colon k < i \land e_k = \mathrm{send}_c(m_k) \land m_i =_E m_k \land \mathrm{d}(c,b) \leq (t_j - t_k) \cdot \mathsf{c})$.

We denote the transitive closure of $\rightsquigarrow_\tau$ by $\rightsquigarrow_\tau^*$.

When timed events are related by $\rightsquigarrow_\tau^*$, there are constraints on the times at which they can occur, as shown by the next two lemmas.

Lemma 1. Let $\tau \in [\![\mathcal{P}]\!]_\mathrm{d}$ be a trace. Then for each pair of timed events such that $(t_i, e_i) \rightsquigarrow_\tau^* (t_j, e_j)$ we have:

$$t_j - t_i \geq \frac{\mathrm{d}(actor(e_i), actor(e_j))}{\mathsf{c}}$$

Proof. From the definition of $\rightsquigarrow_\tau$ we deduce that if $(t_x, e_x) \rightsquigarrow_\tau (t_y, e_y)$ we have:

$$t_y - t_x \geq \frac{\mathrm{d}(actor(e_x), actor(e_y))}{\mathsf{c}}$$

Let us assume $(t_i, e_i) = (t_{i_1}, e_{i_1}) \rightsquigarrow_\tau \dots \rightsquigarrow_\tau (t_{i_k}, e_{i_k}) = (t_j, e_j)$. The property above and the triangle inequality lead to the required result:

$$\begin{aligned} t_j - t_i = t_{i_k} - t_{i_1} &= \sum_{j=1}^{k-1} t_{i_{j+1}} - t_{i_j} \geq \sum_{j=1}^{k-1} \frac{\mathrm{d}(actor(e_{i_j}), actor(e_{i_{j+1}}))}{\mathsf{c}} \\ &\geq \frac{\mathrm{d}(actor(e_{i_1}), actor(e_{i_k}))}{\mathsf{c}} = \frac{\mathrm{d}(actor(e_i), actor(e_j))}{\mathsf{c}} \end{aligned}$$

□

Lemma 2. Let $\tau = (t_1, e_1) \cdots (t_n, e_n) \in [\![\mathcal{P}]\!]_\mathrm{d}$ be a trace, $i, j \in \{1, \dots, n\}$ such that $actor(e_i) = actor(e_j) = a$, and $\delta \in \mathbb{R}_+$ a constant. If $t_j - t_i \leq \frac{2\delta}{\mathsf{c}}$, then for all $k \in \{i, \dots, j\}$ such that $(t_i, e_i) \rightsquigarrow_\tau^* (t_k, e_k) \rightsquigarrow_\tau^* (t_j, e_j)$ we have $\mathrm{d}(a, actor(e_k)) \leq \delta$.

Proof. By the previous lemma, we have that:

$$\begin{aligned} &\frac{2\delta}{\mathsf{c}} \geq t_j - t_i = (t_j - t_k) + (t_k - t_i) \geq \\ &\frac{\mathrm{d}(a, actor(e_k)) + \mathrm{d}(actor(e_k), a)}{\mathsf{c}} = \frac{2 \cdot \mathrm{d}(a, actor(e_k))}{\mathsf{c}} \\ &\implies \mathrm{d}(a, actor(e_k)) \leq \delta \end{aligned}$$

□

The following definition states the minimal necessary condition that needs to be satisfied by a sequence of timed events in order for it to be a valid trace of some protocol: for each recv event there is a corresponding send event.

Definition 7. A sequence of timed events $(t_1, e_1) \cdots (t_n, e_n)$ is time valid if $t_1 \leq \cdots \leq t_n$ and:

$$\begin{aligned} &\forall i \in \{1, \dots n\}, \forall m \colon e_i = \mathrm{recv}_a(m) \\ &\implies \exists j \in \{1, \dots n\}, \exists m' \colon (t_j, e_j) \rightsquigarrow_\tau (t_i, e_i) \wedge \\ &\qquad e_j = \mathrm{send}_b(m') \wedge m =_E m'. \end{aligned}$$

The following definition introduces a notation for the local view of a trace by an agent.

Definition 8. Given a trace $\tau = (t_1, e_1) \cdots (t_n, e_n) \in [\![\mathcal{P}]\!]_{\mathrm{d}}$, and an agent $a \in \mathsf{Agent}$, we define τ_a to be the sequence of timed events in τ executed by a. The *timeless projection* of τ, denoted by $\pi(\tau)$ is the list $e_1 \cdots e_n$.

At this point, we state a general result that relates two traces given by the semantics of a protocol.

Lemma 3. Let $\tau = (t_1, e_1) \cdots (t_n, e_n)$ and $\tau' = (t'_1, e'_1) \cdots (t'_n, e'_n)$ be two time valid sequences of events, and $\mathcal{P}$ a protocol. If $\forall a \in \mathsf{Agent} \colon \pi(\tau_a) = \pi(\tau'_a)$ then $\tau \in [\![\mathcal{P}]\!]_{\mathrm{d}} \iff \tau' \in [\![\mathcal{P}]\!]_{\mathrm{d}}$.

Proof. By symmetry, it is only necessary to prove that $\tau \in [\![\mathcal{P}]\!]_{\mathrm{d}} \implies \tau' \in [\![\mathcal{P}]\!]_{\mathrm{d}}$.

Notice that as $\tau \in [\![\mathcal{P}]\!]_{\mathrm{d}}$, then τ can be generated inductively by applying the rules defined in the semantics in $[\![\mathcal{P}]\!]_{\mathrm{d}}$. Furthermore, in this semantics, all time constraints in τ are generated by the application of the $\mathsf{Recv}_{\mathrm{d}}$ rule.

We prove that each prefix of τ' can be generated by applying the rules in the semantics. The proof follows by induction in the size of the prefixes. The base case is the empty trace, which by definition is a valid trace of any protocol. For the induction step, assume for some i that $(t'_1, e'_1) \cdots (t'_i, e'_i) \in [\![\mathcal{P}]\!]_{\mathrm{d}}$ and that $a = \mathit{actor}(e'_{i+1})$. We prove that $(t'_1, e'_1) \cdots (t'_{i+1}, e'_{i+1}) \in [\![\mathcal{P}]\!]_{\mathrm{d}}$ by case analysis:

- e'_{i+1} is a receive event: as τ' is a time valid sequence of timed events, it can be generated by applying the $\mathsf{Recv}_{\mathrm{d}}$ rule at time t'_{i+1}.
- e'_{i+1} is not a receive event: notice that

$$\begin{aligned} &\pi(\tau_a) = \pi(\tau'_a) \implies \exists j \in \{1, \dots, n\} \colon \\ &\pi\left(\left((t'_1, e'_1) \cdots (t'_{i+1}, e'_{i+1})\right)_a\right) = \pi\left(\left((t_1, e_1) \cdots (t_j, e_j)\right)_a\right) \\ &\wedge e'_{i+1} = e_j \end{aligned}$$

  The same rule that was used to generate (t_j, e_j) can also be used to generate (t'_{i+1}, e'_{i+1}), given that this rule only depends on the sequence of previous events by agent a.

We conclude all events in τ' can be generated inductively by the rules, as was needed. □

The next lemma proves the existence of a trace in a protocol which will be useful for the main result in this section. Given a trace, for each pair of events by the same actor, it is possible to construct another trace for which all the events in between the pair are executed by close agents. In the new trace some events are postponed and others are anticipated with respect to the pair. This result relies heavily on Lemmas 2 and 3.

Lemma 4. Let $\mathcal{P}$ be a protocol and $a \in \mathsf{Agent}$. Let $\tau = (t_1, e_1) \cdots (t_n, e_n) \in [\![\mathcal{P}]\!]_{\mathrm{d}}$ be a trace such that there exist two timed events $(t_u, e_u), (t_v, e_v) \in \tau$ with $u < v$, $t_v - t_u \leq \frac{2 \cdot \delta}{\mathsf{c}}$ and $actor(e_u) = actor(e_v) = a$. Then there exists a trace $\tau' = (t'_1, e'_1) \cdots (t'_n, e'_n) \in [\![\mathcal{P}]\!]_{\mathrm{d}}$ and a bijection f from $\{1, \dots, n\}$ to $\{1, \dots, n\}$ such that:

- $\forall i \in \{1, \dots, n\} \colon e'_{f(i)} = e_i$; the events of τ' are a permutation of the events in τ
- $\forall a \in \mathsf{Agent} \colon \pi(\tau_a) = \pi(\tau'_a)$; the permutation preserves the local order of events for each agent
- $t'_{f(v)} - t'_{f(u)} \leq \frac{2 \cdot \delta}{\mathsf{c}}$ and $f(u) < f(v)$; the time restriction in τ translates to a corresponding time restriction in τ'
- $\forall k \in \{f(u), \dots, f(v)\} \colon \mathrm{d}(a, actor(e'_k)) \leq \delta$; agents executing events between $f(u)$ and $f(v)$ have a bounded distance to a

Proof. For any trace $\tau^* = (t^*_1, e^*_1) \cdots (t^*_n, e^*_n)$, let $K_{\tau^*} \subseteq \{1, \dots, n\} \colon k \in K_{\tau^*} \iff \mathrm{d}(a, actor(e^*_k)) > \delta \wedge u < k < v$. The set K_{τ^*} thus contains the indices of all timed events between u and v that are executed by an agent outside the vicinity of a.

Let $\tau^0 = (t^0_1, e^0_1) \cdots (t^0_n, e^0_n) = \tau$ and $r = |K_\tau|$. If $r = 0$ then all the necessary conditions are fulfilled for $\tau' = \tau$ and f the identity. Otherwise, there exist $\tau^1, \dots, \tau^r$, where $\forall i \in \{1, \dots, r\} \colon \tau^i = (t^i_1, e^i_1) \cdots (t^i_n, e^i_n)$, and a sequence of bijections $f_1, \dots, f_r$ such that:

- $\forall i \in \{1, \dots, r\} \colon f_i$ is a bijection between $\{1, \dots, n\}$ and $\{1, \dots, n\}$ such that $\forall j \in \{1, \dots, n\} \colon e^{i-1}_j = e^i_{f_i(j)}$
- $\forall i \in \{1, \dots, r\}, \forall a \in \mathsf{Agent} \colon \pi(\tau^i_a) = \pi(\tau^{i-1}_a)$
- $\forall i \in \{1, \dots, r\}, \exists k \in K_{\tau^{i-1}} \colon K_{\tau^i} = K_{\tau^{i-1}} \setminus \{k\}$
- $t^1_{f_1(u)} = t^0_u = t_u \wedge t^1_{f_1(v)} = t^0_v = t_v$
- $\forall i \in \{2, \dots, r\} \colon t^i_{f_1 \circ \dots \circ f_i(u)} = t^{i-1}_{f_1 \circ \dots \circ f_{i-1}(u)} \wedge t^i_{f_1 \circ \dots \circ f_i(v)} = t^{i-1}_{f_1 \circ \dots \circ f_{i-1}(v)}$
- $\forall i \in \{1, \dots, r\} \colon \tau^i$ is a time valid sequence.

Notice $\tau' = \tau^r$ and $f = f_1 \circ \dots \circ f_r$ fulfil the required conditions in the lemma:

- $\forall a \in \mathsf{Agent} \colon \pi(\tau_a) = \pi(\tau'_a)$ and $\forall i \in \{1, \dots, n\} \colon e'_{f(i)} = e_i$ are true by definition of f, given it is the composition of functions that possess the same properties.
- $K_{\tau'} = \emptyset \implies \forall k \in \{f(u), \dots, f(v)\} \colon \mathrm{d}(a, actor(e'_k)) \leq \delta$
- $t'_{f(u)} = t_u \wedge t'_{f(v)} = t_v$, as $t'_{f(u)} = t^r_{f_1 \circ \dots \circ f_r(u)} = t^{r-1}_{f_1 \circ \dots \circ f_{r-1}(u)} = \dots = t^1_{f_1(u)} = t^0_u = t_u$ and the same can be deduced for v. Then $t'_{f(v)} - t'_{f(u)} = t_v - t_u \leq \frac{2 \cdot \delta}{\mathsf{c}}$

$\tau' \in [\![\mathcal{P}]\!]_{\mathrm{d}}$ follows from Lemma 3, as we have $\tau' = \tau^r$ is a time valid sequence by definition, and that $\forall a \in \mathsf{Agent}\colon \pi(\tau_a) = \pi(\tau'_a)$ as mentioned above. We complete the proof by showing how to construct the trace τ^i and the bijection f_i from τ^{i-1}, for any index i.

To simplify notation, in what follows we will refer to traces τ, τ' (instead of τ^i, τ^{i+1}), bijection f (instead of f_i), integers x and y (instead of $f_1 \circ \ldots \circ f_{i-1}(u)$ and $f_1 \circ \ldots \circ f_{i-1}(v)$).

Let $k \in K_\tau$, then by Lemma 2 $(t_x, e_x) \rightsquigarrow^*_\tau (t_k, e_k) \wedge (t_k, e_k) \rightsquigarrow^*_\tau (t_y, e_y)$ is false. We analyse two cases:

- $(t_k, e_k) \rightsquigarrow^*_\tau (t_y, e_y)$ is false
- $(t_x, e_x) \rightsquigarrow^*_\tau (t_k, e_k)$ is false

We will focus on the first case, the other one can be analysed in an analogous way. Let ω be a constant greater than $t_y - t_k$ and $\tau'' = (t''_1, e''_1) \cdots (t''_n, e''_n)$ a sequence of timed events defined as follows:

1. for each timed event $(t_i, e_i) \in \tau$ such that $i > y$, let $(t''_i, e''_i) = (t_i + \omega, e_i)$
2. for each timed event $(t_i, e_i) \in \tau$ such that $i < y$ and $(t_k, e_k) \rightsquigarrow^*_\tau (t_i, e_i)$, let $(t''_i, e''_i) = (t_i + \omega, e_i)$. In this case clearly $i \geq k$
3. for every other timed event $(t_i, e_i) \in \tau$, let $(t''_i, e''_i) = (t_i, e_i)$

Next, we define a bijection f from $\{1, \ldots, n\}$ to $\{1, \ldots, n\}$ according to the following property:

- $\forall i, j \in \{1, \ldots, n\}\colon (t''_i < t''_j) \vee (t''_i = t''_j \wedge i < j) \iff f(i) < f(j)$

Notice that f exists and is unique as it defines a stable order for the values of $t''_1, \ldots, t''_n$.

At this point we are ready to define the trace $\tau' = (t'_1, e'_1) \cdots (t'_n, e'_n)$. For all $i \in \{1, \ldots, n\}$ let $(t'_i, e'_i) = (t''_{f^{-1}(i)}, e''_{f^{-1}(i)})$. Then we deduce $\forall i \in \{1, \ldots, n\}\colon e'_{f(i)} = e_i \wedge (t'_{f(i)} = t_i \vee t'_{f(i)} = t_i + \omega)$

Notice this f and τ' fulfil the required conditions:

- By construction $\forall i \in \{1, \ldots, n\}\colon e_i = e'_{f(i)}$
- Assume $i, j \in \{1, \ldots, n\}$ such that $i < j \wedge \mathit{actor}(e_i) = \mathit{actor}(e_j)$. Then $(t_i, e_i) \rightsquigarrow^*_\tau (t_j, e_j)$ is true, and $t'_{f(j)} = t_j + \omega \implies t'_{f(i)} = t_i + \omega$. We conclude that $\forall a \in \mathsf{Agent}\colon \pi(\tau_a) = \pi(\tau'_a)$
- By construction $t'_{f(x)} = t_x \wedge t'_{f(y)} = t_y$
- $t'_{f(k)} = t_k + \omega > t_y = t'_{f(y)} \implies f(k) > f(y) \implies \neg(f(x) < f(k) < f(y))$
- τ' is a time valid sequence given that by construction:

$$\forall i, j \in \{1, \ldots, n\}\colon (t_i, e_i) \rightsquigarrow^*_\tau (t_j, e_j) \iff (t'_{f(i)}, e'_{f(i)}) \rightsquigarrow^*_\tau (t'_{f(j)}, e'_{f(j)})$$

□

The following example shows how the transformation of the previous lemma works for a simple trace.

Example 3. Let τ be the following execution of $\mathcal{P}_{ex}$:

$$s_0 \xrightarrow{(0,\mathrm{create}_a(id1,\mathcal{P}_{ex}(V)))} s_1 \xrightarrow{(0,\mathrm{create}_b(id2,\mathcal{P}_{ex}(P)))} s_2 \xrightarrow{(0,\mathrm{create}_e(id3,E))} s_3$$
$$\xrightarrow{(0,\mathrm{send}_b(\langle \mathsf{pk}(k),\mathsf{sign}(n,k)\rangle))} s_4 \xrightarrow{(100,\mathrm{recv}_a(\langle \mathsf{pk}(k),\mathsf{sign}(n,k)\rangle))} s_5 \xrightarrow{(100,\mathrm{send}_a(n'))} s_6$$
$$\xrightarrow{(200,\mathrm{recv}_b(n'))} s_7 \xrightarrow{(200,\mathrm{send}_b(n'\oplus n))} s_8 \xrightarrow{(250,\mathrm{recv}_e(n'))} s_9 \xrightarrow{(250,\mathrm{send}_e(n'\oplus m'))} s_{10}$$
$$\xrightarrow{(300,\mathrm{recv}_a(n'\oplus n))} s_{11} \xrightarrow{(300,\mathrm{clock}_a(2,3))} s_{12} \xrightarrow{(300,\mathrm{send}_a(\mathsf{aenc}(k',\mathsf{pk}(k))))} s_{13}$$
$$\xrightarrow{(400,\mathrm{recv}_b(\mathsf{aenc}(k',\mathsf{pk}(k))))} s_{14} \xrightarrow{(400,\mathrm{send}_b(\mathsf{senc}(\langle n,b\rangle,k')))} s_{15}$$
$$\xrightarrow{(500,\mathrm{recv}_a(\mathsf{senc}(\langle n,b\rangle,k')))} s_{16} \xrightarrow{(500,\mathrm{equal}_a(\langle \mathsf{verify}(\mathsf{sign}(n,k),b,\mathsf{pk}(k)),\mathsf{true}\rangle))} s_{17}$$
$$\xrightarrow{(500,\mathrm{claim}_a(sec,\langle b,k'\rangle))} s_{18}$$

Then according to Lemma 4 with $\delta = \frac{\mathsf{c}\cdot 200}{2}$, $\mathrm{d}(a,b) = 0$, $\mathrm{d}(a,e) > \delta$, $u = 6$, $v = 11$, one possible τ' is:

$$s_0 \xrightarrow{(0,\mathrm{create}_a(id1,\mathcal{P}_{ex}(V)))} s_1 \xrightarrow{(0,\mathrm{create}_b(id2,\mathcal{P}_{ex}(P)))} s_2$$
$$\xrightarrow{(0,\mathrm{create}_e(id3,E))} s_3 \xrightarrow{(0,\mathrm{send}_b(\langle \mathsf{pk}(k),\mathsf{sign}(n,k)\rangle))} s_4$$
$$\xrightarrow{(100,\mathrm{recv}_a(\langle \mathsf{pk}(k),\mathsf{sign}(n,k)\rangle))} s_5 \xrightarrow{(100,\mathrm{send}_a(n'))} s_6 \xrightarrow{(200,\mathrm{recv}_b(n'))} s_7$$
$$\xrightarrow{(200,\mathrm{send}_b(n'\oplus n))} s_8 \xrightarrow{(300,\mathrm{recv}_a(n'\oplus n))} s_9 \xrightarrow{(300,\mathrm{clock}_a(2,3))} s_{10}$$
$$\xrightarrow{(300,\mathrm{send}_a(\mathsf{aenc}(k',\mathsf{pk}(k))))} s_{11} \xrightarrow{(301,\mathrm{recv}_e(n'))} s_{12} \xrightarrow{(301,\mathrm{send}_e(n'\oplus m))} s_{13}$$
$$\xrightarrow{(400,\mathrm{recv}_b(\mathsf{aenc}(k',\mathsf{pk}(k))))} s_{14} \xrightarrow{(400,\mathrm{send}_b(\mathsf{senc}(\langle n,b\rangle,k')))} s_{15}$$
$$\xrightarrow{(500,\mathrm{recv}_a(\mathsf{senc}(\langle n,b\rangle,k')))} s_{16} \xrightarrow{(500,\mathrm{equal}_a(\langle \mathsf{verify}(\mathsf{sign}(n,k),b,\mathsf{pk}(k)),\mathsf{true}\rangle))} s_{17}$$
$$\xrightarrow{(500,\mathrm{claim}_a(sec,\langle b,k'\rangle))} s_{18}$$

4.4.2 Security properties

The following corollary applies Lemma 4 to a protocol trace and to the events defined inside a clock event of interest.

Corollary 1. Let $\tau \in [\![\mathcal{P}]\!]_{\mathrm{d}}$ be an execution trace. If $e_i = \mathrm{claim}_a(\psi, m)$ is a claim in τ and $\mathsf{correct\_time}_\delta(\tau, i)$ is true, then there exists a trace $\tau' \in [\![\mathcal{P}]\!]_{\mathrm{d}}$ such that the following conditions hold:

1. $\forall c \in \mathsf{Agent}\colon \pi(\tau_c) = \pi(\tau'_c)$.

2. $e'_i = e_i$

3. $\{e_1, e_2, \ldots, e_i\} = \{e'_1, e'_2, \ldots, e'_i\}$

4. If $run(\tau', id_i) = (t'_{i_1}, e'_{i_1})^{id_i} \cdots (t'_{i_k}, e'_{i_k})^{id_i}$ then $\forall j \in \{i_1, \ldots, i_k\}$:

$$e'_j = \mathrm{clock}_a(x, y) \implies (\forall z\colon (i_x < z < i_y \implies \mathrm{d}(actor(e'_z), a) \leq \delta))$$

Proof. This is a direct application of the Lemma 4 to the events mentioned in the clock events in the same run as the claim.

Notice that as the predicate $\mathsf{correct\_time}$ is true, for all clock events $\mathrm{clock}_a(x, y)$ in the run corresponding to e_i, we get that $t_{i_y} - t_{i_x} \leq \frac{2 \cdot \delta}{\mathsf{c}}$. As such, we can apply Lemma 4 and obtain a new trace in which all actors executing an event between i_x and i_y are near a. Doing this procedure for each clock event, we get the desired trace. This is possible given that the segments events referenced in clock events do not intersect. □

Before introducing the main result in this section, we need to restrict the security properties in our model, specified in the next definition, so that the previous lemma is applicable.

Definition 9. Let $\mathcal{P}$ a protocol and ψ a security property. We say ψ is a *robust* security property if and only if for all $\tau, \tau' \in [\![\mathcal{P}]\!]_{\mathrm{d}}$ such that $e_i = \mathrm{claim}_a(\psi, m)$, and the first three conditions in Corollary 1 hold, we also have

$$\begin{aligned} &\mathsf{dist\_attacker}_\delta(\tau, i) \wedge \mathsf{correct\_time}_\delta(\tau, i) \implies \psi(\tau, i) \\ &\iff \\ &\mathsf{dist\_attacker}_\delta(\tau', i) \wedge \mathsf{correct\_time}_\delta(\tau', i) \implies \psi(\tau', i) \end{aligned}$$

The next lemma enables the use of any *robust* security property in our main equivalence result.

Lemma 5. Let ψ be a *robust* security property. Then $\forall \tau \in [\![\mathcal{P}]\!]_{\mathrm{d}}$, such that $e_i = \mathrm{claim}_a(\psi, m)$, and τ' is a trace as defined in Corollary 1 with respect to τ, then:

$$\begin{aligned} &\mathsf{dist\_attacker}_\delta(\tau, i) \wedge \mathsf{correct\_time}_\delta(\tau, i) \implies \psi(\tau, i) \\ &\iff \\ &\mathsf{dist\_attacker}_\delta(\tau', i) \wedge \mathsf{correct\_time}_\delta(\tau', i) \implies \psi(\tau', i) \end{aligned}$$

Next, we define formally the main security properties our model supports, which we employ in the case studies later.

Definition 10 (Secrecy)**.**

$$\begin{aligned} &\forall \tau \in [\![\mathcal{P}]\!]_{\mathrm{d}}, \tau_i = (t_i, \mathrm{claim}_a(sec, \langle b, k \rangle))\colon \\ &\mathsf{dist\_attacker}_\delta(\tau, i) \wedge \mathsf{correct\_time}_\delta(\tau, i) \implies \\ &(\nexists c \in \mathsf{Dishonest}\colon c \vdash_\tau k) \vee b \in \mathsf{Dishonest} \end{aligned}$$

For some protocols the communication partner is not known. The next definition covers this case, which we call anonymous secrecy.

Definition 11 (Anonymous Secrecy).

$$\begin{aligned}
&\forall \tau \in [\![\mathcal{P}]\!]_{\mathrm{d}}, \tau_i = (t_i, \mathrm{claim}_a(a\_sec, k))\colon\\
&\mathsf{dist\_attacker}_\delta(\tau, i) \wedge \mathsf{correct\_time}_\delta(\tau, i) \implies\\
&(\nexists c \in \mathsf{Dishonest}\colon c \vdash_\tau k)
\end{aligned}$$

In our case studies we use the non-injective agreement property [89], its anonymous variant and the secure remote erasure [129]. In what follows we define these properties formally.

Definition 12 (Non injective agreement).

$$\begin{aligned}
&\forall \tau \in [\![\mathcal{P}]\!]_{\mathrm{d}} \tau_i = (t_i, \mathrm{claim}_a(non\_in\_agree, \langle b, m\rangle))\colon\\
&\mathsf{dist\_attacker}_\delta(\tau, i) \wedge \mathsf{correct\_time}_\delta(\tau, i)\\
&\ \implies (\exists j < i\colon e_j = \mathrm{running}_b(\langle roleB, a, m\rangle))\\
&\ \vee b \in \mathsf{Dishonest}
\end{aligned}$$

Definition 13 (Anonymous non injective agreement).

$$\begin{aligned}
&\forall \tau \in [\![\mathcal{P}]\!]_{\mathrm{d}}, \tau_i = (t_i, \mathrm{claim}_a(a\_non\_in\_agree, m))\colon\\
&\mathsf{dist\_attacker}_\delta(\tau, i) \wedge \mathsf{correct\_time}_\delta(\tau, i)\\
&\ \implies (\exists b \in \mathsf{Agent}, j < i\colon e_j = \mathrm{running}_b(\langle roleB, m\rangle))
\end{aligned}$$

Definition 14 (Remote Memory-erasure).

$$\begin{aligned}
&\forall \tau \in [\![\mathcal{P}]\!]_{\mathrm{d}}, \tau_i = (t_i, \mathrm{claim}_a(erasure, \langle b, m\rangle)),\colon\\
&\mathsf{dist\_attacker}_\delta(\tau, i) \wedge \mathsf{correct\_time}_\delta(\tau, i) \implies\\
&\exists j < i\colon (t_j, \mathrm{send}_b(m)) \in \tau \vee (t_j, \mathrm{recv}_b(m)) \in \tau
\end{aligned}$$

The next proposition shows that all the security properties defined in this section are *robust*.

Proposition 1. *sec*, *a_sec*, *non_in_agree*, *a_non_in_agree*, and *erasure* are *robust* security properties.

Proof. Let $\tau, \tau' \in [\![\mathcal{P}]\!]_{\mathrm{d}}$ such that the first three conditions in Corollary 1 hold. Then the condition $\forall c \in \mathsf{Agent}\colon \pi(\tau_c) = \pi(\tau'_c)$ implies that in both traces the knowledge deduced by dishonest agents is the same, so *sec* and *a_sec* are *robust*.

The properties *non_in_agree*, *a_non_in_agree* and *erasure* are robust given that all of them depend on the existence of events before e_i, and by definition $e_i = e'_i$ and:

$$\{e_1, e_2, \ldots, e_i\} = \{e'_1, e'_2, \ldots, e'_i\}$$

$\square$

4.4.3 A timeless protocol semantics

Now we simplify the security model by eliminating the notion of time from the original model, and provide a causality-based characterization of the dist_attacker predicate. We do so by removing all occurrences of the time variables in the rules $\mathsf{Create}_{\mathcal{P}}$, Send, $\mathsf{Recv}_{\mathrm{d}}$, Equal and Signal, producing their timeless equivalent $\mathsf{Create}_{\mathcal{P}\pi}$, Send_{π}, Recv_{π}, Equal_{π} and Signal_{π}. For reference, the modified rules follow.

The timeless semantics is obtained by removing the timestamps from the rules used in the time-based model. This means that traces obtained from this semantics are just a sequence of events. The resulting rules are as follows.

$$\frac{\begin{array}{c} s \in Q, R \in \mathsf{R}, a \in \mathsf{Agent}, id \in_s \mathsf{Id}, \\ a \in \mathsf{Honest} \implies R \in roles(\mathcal{P}) \end{array}}{s \xrightarrow{(\mathrm{create}_a(R))^{id}} s \cup \{(id, \epsilon, R, a)\}} \ \mathsf{Create}_{\mathcal{P}\pi},$$

The template rule for the timeless semantics, and its corresponding instantiations follow:

$$\frac{\begin{array}{c} s \in Q, (id, \tau, R, a) \in s, \tau = e_1 \cdots e_i, \\ e_1 \cdots e_i e_{i+1} \in_{s \setminus (id,\tau,R,a)} insts(R), \\ [\,] \end{array}}{s \xrightarrow{(e_{i+1})^{id}} (s \setminus \{(id, \tau, R, a)\}) \cup \{(id, \tau \cdot e_{i+1}, R, a)\}},$$

$$\to \left[e_{i+1} = \mathrm{send}_a(m), a \vdash_{\{e_1,\ldots,e_i\}} m\right] \qquad \mathsf{Send}_{\pi}$$

$$\to \left[\begin{array}{c} e_{i+1} = \mathrm{recv}_a(m), \mathrm{send}_b(m') \in \mathit{labels}(s), \\ m =_E m' \end{array}\right] \qquad \mathsf{Recv}_{\pi}$$

$$\to [e_{i+1} = \mathrm{equal}_a(\langle m_1, m_2 \rangle), m_1 =_E m_2] \qquad \mathsf{Equal}_{\pi}$$

$$\to [e_{i+1} \in \mathsf{SignalEvent}] \qquad \mathsf{Signal}_{\pi}$$

The timeless protocol semantics associated with these rules produces a sequence of events, rather than a sequence of timed events.

Definition 15 (Timeless protocol semantics)**.** The timeless semantics of a protocol $\mathcal{P}$ is the LTS $(Q, \Lambda, \to, s_0)$ where,

- $Q = \mathbb{P}(\mathsf{Id} \times \mathsf{Ev_g}^* \times \mathsf{R} \times \mathsf{Agent})$
- $\Lambda = \mathsf{Ev_g} \times \mathsf{Id} \ \cup \ \{\mathrm{create}_a(id, R)^{id} | a \in \mathsf{Agent}, id \in \mathsf{Id}, R \in \mathsf{R}\}$
- $\to = \mathsf{Create}_{\mathcal{P}\pi} \cup \mathsf{Send}_{\pi} \cup \mathsf{Recv}_{\pi} \cup \mathsf{Equal}_{\pi} \cup \mathsf{Signal}_{\pi}$.

- $s_0 = \emptyset$

We use $[\![\mathcal{P}]\!]_\pi$ to denote the set of traces obtained from $\mathcal{P}$'s timeless semantics.

4.4.4 A causality-based interpretation of the distant-attacker assumption

In the previous section, we introduced the notion of a distant-attacker. This notion depended on a distance parameter δ that defines the vicinity of the actor in question. Here we provide a similar definition adapted to the timeless case. Let $\sim$ be an equivalence relation on the set of actors with two equivalence classes. Intuitively, actors in the same class ($a \sim b$) are near, and those in different classes are far. The $\mathsf{dist\_attacker}_\pi$ predicate is defined by:

$$\begin{aligned}\mathsf{dist\_attacker}_\pi(\tau, i) \iff{}& \forall c \in (actors(\tau) \cap \mathsf{Dishonest})\colon \\ & \neg(a \sim c) \vee c \in \mathsf{dishonest\_agents}(e_i)\end{aligned}$$

Our definition of $\mathsf{correct\_time}$ for the timeless case is inspired by the causal characterization of distance-bounding given in [95]. Our main result later shows how this definition exactly fits our objective. For every trace $\tau = e_1^{id_1} \cdots e_n^{id_n}$ and every index i such that e_i is a claim event by agent a and $run(\tau, id_i) = e_{i_1} \cdots e_{i_k}$ we have:

$$\begin{aligned}\mathsf{correct\_time}_\pi(\tau, i) \iff{}& \forall j \in \{i_1, \ldots, i_k\}\, \forall b \in \mathsf{Agent}\colon \\ & (e_j = \mathrm{clock}_a(x, y) \wedge j < i \wedge \neg(a \sim b)) \\ & \implies \nexists k\colon (i_x \leq k \leq i_y \wedge actor(e_k) = b)\end{aligned}$$

A protocol $\mathcal{P}$ satisfies ψ conditional to the trust assumption $\mathsf{dist\_attacker}_\pi$ and the proximity check $\mathsf{correct\_time}_\pi$ within the timeless semantics if, for every trace $\tau \in [\![\mathcal{P}]\!]_\pi$ and every $i \in \{1, \ldots, |\tau|\}$, where e_i is a claim event with security property ψ, it holds that

$$\mathsf{dist\_attacker}_\pi(\tau, i) \wedge \mathsf{correct\_time}_\pi(\tau, i) \implies \psi(\tau, i)$$

We denote this property by $[\![\mathcal{P}]\!]_\pi \rhd_\sim \psi$.

4.4.5 Equivalence between the timed and the timeless semantics

Finally, we prove that, for every robust security property ψ defined as a first-order statement on sequence of events, ψ is correct in the timeless protocol semantics if and only if it is secure in the timed protocol semantics.

Lemma 6. Let $\tau \in [\![\mathcal{P}]\!]_\mathrm{d}$ and $\pi(\tau)$ be the timeless projection of τ. Then $\pi(\tau) \in [\![\mathcal{P}]\!]_\pi$.

Proof sketch. Notice that if τ is generated according to the rules in the timed semantics of $\mathcal{P}$, then $\pi(\tau)$ can also be generated by the corresponding rules in the timeless semantics, as the latter is less restrictive than the former. $\square$

Now we are ready to formulate the main result.

Theorem 1. Given a protocol $\mathcal{P}$ and a *robust* security property ψ. Then $\forall\, \mathrm{d}\colon [\![\mathcal{P}]\!]_{\mathrm{d}} \rhd_{\delta} \psi \iff \forall \sim\colon [\![\mathcal{P}]\!]_{\pi} \rhd_{\sim} \psi$.

Proof. We prove the left right implication first:

$$\forall\, \mathrm{d}\colon [\![\mathcal{P}]\!]_{\mathrm{d}} \rhd_{\delta} \psi \implies \forall \sim\colon [\![\mathcal{P}]\!]_{\pi} \rhd_{\sim} \psi$$

We will formally prove the contrapositive.
Assuming $\forall \sim\colon [\![\mathcal{P}]\!]_{\pi} \rhd_{\sim} \psi$ is false, we deduce

$$\begin{aligned}&\exists \sim, a \in \mathsf{Agent}, p \in \{1,\dots,n\}, \tau \in [\![\mathcal{P}]\!]_{\pi}\colon\\ &e_p = \mathrm{claim}_a(\psi, m) \wedge \mathsf{dist\_attacker}_{\pi}(\tau, p) \wedge\\ &\mathsf{correct\_time}_{\pi}(\tau, p) \wedge \neg\psi(\tau, p)\end{aligned}$$

We will construct a trace $\tau' = (t'_1, e'_1)\cdots(t'_n, e'_n) \in [\![\mathcal{P}]\!]_{\mathrm{d}}$, for some distance function d, such that $\tau = \pi(\tau')$. To achieve this, we will assign locations to actors such that the feasible values of t'_i exist.

For simplicity, we assume the run corresponding to e_p contains only one clock event. Let $run(\tau, id_p) = (e_{p_1})^{id_p}\cdots(e_{p_k})^{id_p}$, $e_{p_i} = \mathrm{clock}_a(x,y)$, $u = p_x$ and $v = p_y$. We assign locations to actors such that $b \sim c \implies \mathrm{d}(b,c) = 0$ and $\neg(b \sim c) \implies \mathrm{d}(b,c) > \delta$. As $\mathsf{correct\_time}_{\pi}(\tau, p)$ is true, we have $\forall k \in \{u,\dots,v\}\colon \mathrm{d}(actor(e_k), a) = 0$. We also deduce that $\mathsf{dist\_attacker}_{\pi}(\tau, p) \implies \mathsf{dist\_attacker}_{\delta}(\tau', p)$.

Let $\gamma > 0$ be an arbitrarily large constant. Then we are ready to define the values of t'_i:

- $t'_1 = 0$
- $1 < i \leq u \implies t'_i = (i-1)\cdot\gamma$
- $u < i < v \implies t'_i = (i-u)\cdot\frac{2\cdot\epsilon}{\mathsf{c}} + t'_u$
- $t'_v = t'_u + \frac{2\cdot\delta}{\mathsf{c}}$
- $v < i \leq n \implies t'_i = (i-v)\cdot\gamma + t'_v$

Notice for sufficiently large γ and sufficiently small ϵ, all necessary conditions are fulfilled (the time stamps are ordered, no message travels faster than the speed of light, and $t'_v - t'_u \leq \frac{2\cdot\delta}{\mathsf{c}}$ which implies $\mathsf{correct\_time}_{\delta}(\tau', p)$ is true). Then, we deduce τ' can be generated inductively with the rules defined within $[\![\mathcal{P}]\!]_{\mathrm{d}}$. On the other hand, as $\psi(\tau, p)$ doesn't depend on time, then τ' also represents an attack in $[\![\mathcal{P}]\!]_{\mathrm{d}}$, as needed.

In what follows we prove the other implication also using the contrapositive:

$$\forall \sim\colon [\![\mathcal{P}]\!]_{\pi} \rhd_{\sim} \psi \implies \forall\, \mathrm{d}\colon [\![\mathcal{P}]\!]_{\mathrm{d}} \rhd_{\delta} \psi$$

Let $\tau = (t_1, e_1)\cdots(t_n, e_n) \in [\![\mathcal{P}]\!]_{\mathrm{d}}, p \in \{1,\dots,n\}$ such that τ is an attack trace:

$$\begin{aligned}&e_p = \mathrm{claim}_a(\psi, m) \wedge \mathsf{dist\_attacker}_{\delta}(\tau, p) \wedge\\ &\mathsf{correct\_time}_{\delta}(\tau, p) \wedge \neg\psi(\tau, p)\end{aligned}$$

By Corollary 1, a trace $\tau' = (t'_1, e'_1)^{id'_1} \cdots (t'_n, e'_n)^{id'_n} \in [\![\mathcal{P}]\!]_{\mathrm{d}}$ exists such that:

- $\forall c \in \mathsf{Agent}\colon \pi(\tau_c) = \pi(\tau'_c)$.
- $e'_p = e_p$ and $\{e_1, e_2, \ldots, e_p\} = \{e'_1, e'_2, \ldots, e'_p\}$
- $run(\tau', id'_p) = (t'_{p_1}, e'_{p_1})^{id'_p} \cdots (t'_{p_k}, e'_{p_k})^{id'_p} \wedge$
  $\forall j \in \{p_1, \ldots, p_k\}\colon e'_j = \mathrm{clock}_a(x, y) \wedge \forall z\colon p_x < z < p_y$
  $\implies \mathrm{d}(actor(e'_z), a) \leq \delta$

Let $\sim$ be an equivalence relation such that $\mathrm{d}(c, a) \leq \delta \iff c \sim a$. We deduce that

$$\mathsf{dist\_attacker}_\pi(\pi(\tau'), p) \wedge \mathsf{correct\_time}_\pi(\pi(\tau'), p)$$

is true by definition, and that $\neg\psi(\pi(\tau'), p)$ is also true given that ψ is *robust*. So $\pi(\tau') \in [\![\mathcal{P}]\!]_\pi$ is an attack trace, as was needed. $\square$

4.5 Automated verification: Case Studies

This section demonstrates how the theoretical development of previous sections can be used to analyse the security of protocols that rely on the distant-attacker assumption. The section starts by introducing a verification methodology, which we showcase using three recent protocols. Then we report on the security analysis of a number of protocols which, to the best of our knowledge, were lacking a formal analysis. This analysis was aided by the Tamarin prover. The models of each protocol, together with their security lemmas, can be accessed online at[2].

4.5.1 Methodology

Our methodology consists of six steps, whose implementation are showcased by carrying on the analysis of three recent protocols: Move2Auth [139], Amigo [135] and SPEED [9].

1. Description of the original protocol, security requirements and assumptions.
2. Abstraction of the original protocol into a symbolic model.
3. Definition of the security property to be verified as a member of the property class given in Definition 5, i.e. as a timeless security property conditional to the distant-attacker assumption.
4. Prove that the underlying timeless property satisfies Definition 9.
5. Replacement of proximity checks, such as signal strength or visual inspection, by distance-bounding based on round-trip-time measurements. This is a workaround intended to fit into our framework proximity-checking protocols that are not based on round-trip-time.

[2] `https://gitlab.uni.lu/regil/distant-attacker-tamarin`

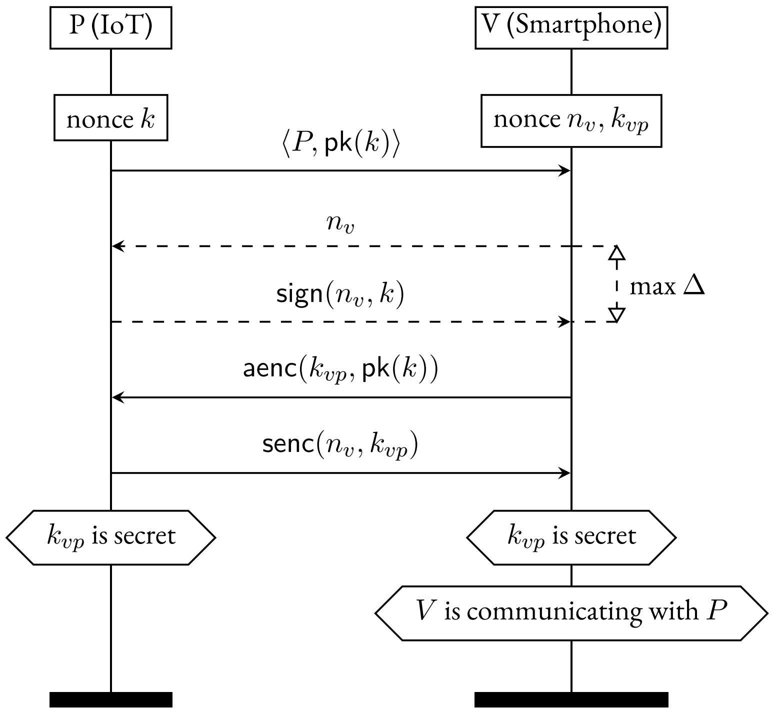

P (IoT)
V (Smartphone)
nonce k
nonce n_v, k_{vp}
$\langle P, \mathsf{pk}(k)\rangle$
n_v
$\mathsf{sign}(n_v, k)$
max Δ
$\mathsf{aenc}(k_{vp}, \mathsf{pk}(k))$
$\mathsf{senc}(n_v, k_{vp})$
k_{vp} is secret
k_{vp} is secret
V is communicating with P

Figure 4.3: A symbolic specification of Move2Auth-RTT

6. Use of a protocol verification tool, such as Tamarin, to implement and verify the resulting protocol within the symbolic model introduced in Section 4.4.

7. If attacks are found, map them back to the original setting of the protocol to ensure they are not a result of the abstraction step.

Steps 2 and 5 above may impose a risk of losing accuracy with respect to the original protocol. The former is standard in symbolic verification, the latter a choice made to analyse several similar protocols within our framework. We acknowledge Step 5 to be a workaround and not a proper solution to the problem of analysing protocols based on signal-strength. The workaround is useful, though, since some attacks found in a protocol modelled with round-trip-time are explainable in a context where signal strength is used (see attacks on Amigo [135] and Move2Auth [139] below). Because radio waves travel very close to the speed of light, it is harder for an attacker to manipulate their distance to honest participants by increasing the propagation speed of the communication channel, compared to manipulating signal strength (which can be amplified). The last step of the methodology is used to tackle the risk of over-abstracting or miss-representing a protocol, by ensuring that attacks found in the abstracted model apply to the original protocol.

4.5.2 Analysing Move2Auth

The protocol Move2Auth [139] aims to provide a secure communication channel between an IoT device and a smartphone. This protocol uses variations in the Received Signal Strength (RSS) perceived by a single antenna to detect proximity. The protocol starts when the smartphone connects to the Wi-Fi network of the device, identified by its SSID. Then the IoT device sends a public key to the smartphone. Immediately after that, the IoT device uses the corresponding private key to encrypt information about its MAC address and its identity. This encrypted data is sent several times to the smartphone, which uses variations on the signal strength of the received packets to determine proximity to the IoT device. The smartphone also verifies that the received packets are correct with respect to the public key received at the start of the protocol. If both signatures and signal strength measurements are correct, the smartphone sends the freshly generated key k_{vp} encrypted with the public key, and concludes that it has correctly exchanged the secret key k_{vp} with the IoT device.

To analyse Move2Auth, we substitute RSS measurements by round-trip-time measurements using a technique established in the literature [15]. This technique consists of a verifier sending a nonce to a prover, expecting to receive as a response a message containing the nonce and a secret key identifying the prover. We added an extra message at the end to let the prover use the exchanged key, as it would in a real scenario. The resulting protocol is displayed in Figure 4.3.

In our abstraction of Move2Auth (Move2Auth-RTT), the device (P) sends its identity (which could represent the SSID of the Wi-Fi network) and a fresh public key $\mathsf{pk}(k)$. Then the smartphone (V) sends a nonce n_v to be used during the fast phase, and expects a message consisting of the signature of n_v with key k, which could only be constructed by the device. Then V generates a fresh key k_{vp} and sends it encrypted to P. Finally, P replies with the nonce n_v encrypted by k_{vp}, showing V that it received the shared key.

We modelled the authentication property from the smartphone's point of view as non-injective agreement on the key k_{vp}. We found this property to be false. The attack consists of an attacker that modifies the identity in the first message. We modelled the secrecy claim by the device as (anonymous) secrecy and found this property to be false, as expected, given that the P does not execute a proximity check on the verifier. The secrecy claim made by the smartphone, on the other hand, is proven correct by Tamarin.

Attacks

The attack on secrecy by P is realized in a trace in which a distant attacker impersonates an honest agent in the verifier role. In the original setting the same attacks applies, as the last message can be sent by the attacker itself. This vulnerability can be potentially exploited because it allows the attacker to control the IoT device by sending encrypted commands, which the IoT device will accept as correct. Regarding the authentication claim by V, the vulnerability derives from the fact that the identity in the first message is not tied to the rest of the messages of the protocol, and it is not protected cryptographically, so the attacker can easily modify it. In the original protocol, this attack would correspond to a situation in which the attacker sets up a fake Wi-Fi network.

4.5.3 Analysing Amigo

Amigo [135] is another protocol for mobile device authentication that depends on proximity. Similar to Move2Auth, each device computes a signature based on the radio environment, and uses it to detect if the other device is near. This is coupled with a key-exchange based on Diffie-Hellman. As in Move2Auth-RTT, we modified the protocol to rely on round trip time rather than signal strength.

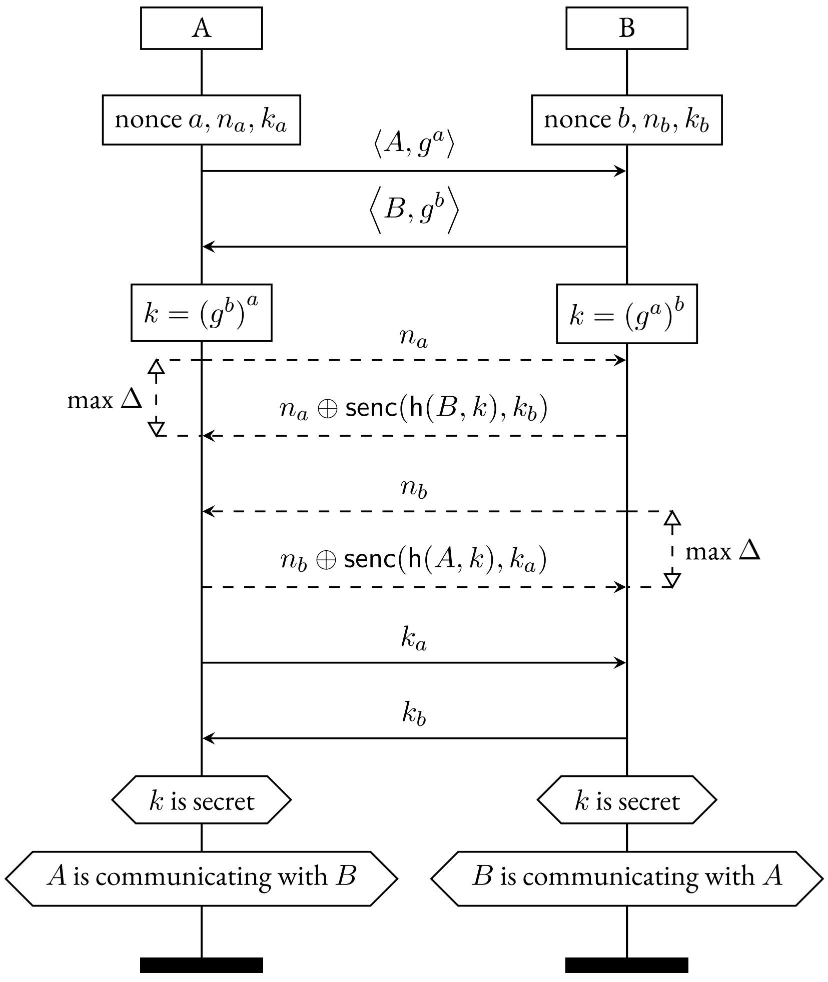


Figure 4.4: A representation of the Amigo-RTT protocol

The modified protocol Amigo-RTT starts by a traditional Diffie-Hellman exchange, then a fast phase is executed by each agent. The rapid phase uses a commitment scheme by encrypting the hash of the agent's identity (A or B) and the common key (k). The encryption key (k_a or k_b) is revealed at the end of the protocol. The abstracted protocol is shown in Figure 4.4.

Attacks

We modelled non-injective agreement and secrecy claims with respect to key k for each role. All claims are invalid. The attack on secrecy for role A works as follows. Two nearby honest agents A and B execute the protocol. The attacker E captures the first two messages, and modifies them so that the new messages are $\langle A, 1\rangle$ and $\langle A, 1\rangle$ where 1 is the unit value in the multiplication group. The protocol continues without any intervention from E, resulting in an exchanged key equal to 1, which is known to E. We note that this attack can be prevented by following a Diffie-Hellman public key validation[3]. Such validation is optional in the specification, but our analysis shows it to be necessary in Amigo-RTT. We found this attack by using the Tamarin extension developed in [44].

The attack on the role A's authentication claim is as follows. Two nearby honest agents X, Y execute the protocol. The attacker E manipulates the messages so that X finishes the execution in role A with Y, while Y was also executing the role A rather than B. Both X and Y send messages $\langle X, g^x\rangle$ and $\langle Y, g^y\rangle$, which the attacker delays such that both are received by the other agent as the second message. When Y starts the proximity check by sending n_y, the attacker responds with a random message m. Notice Y cannot check whether this message is correct until later, so it continues executing the protocol. Then X starts the proximity check by sending n_x, and Y sends the correct response $n_x \oplus \mathsf{senc}(\mathsf{h}(Y, k), k_y)$, even though for Y this is the second proximity check, while for X it is still the first. Y continues the protocol by sending k_y. At this point, E has enough information to complete the protocol with X, by executing the final proximity check in role B.

We finish the analysis of Amigo-RTT by noting that attacks on the claims made by role B are similar to those shown above for role A.

4.5.4 Analysing SPEED

SPEED [9] aims to guarantee that an IoT device has erased its memory. By using a combination of software memory isolation techniques and distance-bounding based on round trip time measurements, the protocol enables a device acting as verifier to erase the memory of another device, of low computational power, acting as prover, only when the verifier is nearby the prover. The protocol starts with a distance-bounding phase based on [33], which allows the prover to check the verifier is nearby. The information exchange in this phase is then used by the prover to compute a fresh key k and a MAC on its memory. The abstraction of the protocol is shown in Figure 4.5. For our security analysis we used the memory erasure property from [129]. The analysis revealed that the security claim made by the verifier is invalid. This is not a surprise given that the verifier does not check its proximity to the prover. Given that prover and verifier do not share cryptographic information before the protocol execution, a trivial impersonation attack can be executed by a distant attacker.

4.5.5 Summary of analysis results

We extended the analysis methodology described above to four more protocols [9, 123, 124, 135] (see Table 4.1). The analysed protocols include (anonymous) secrecy, (anonymous) authentica-

[3]2.1.5. Public Key Validation `https://tools.ietf.org/html/rfc2631`

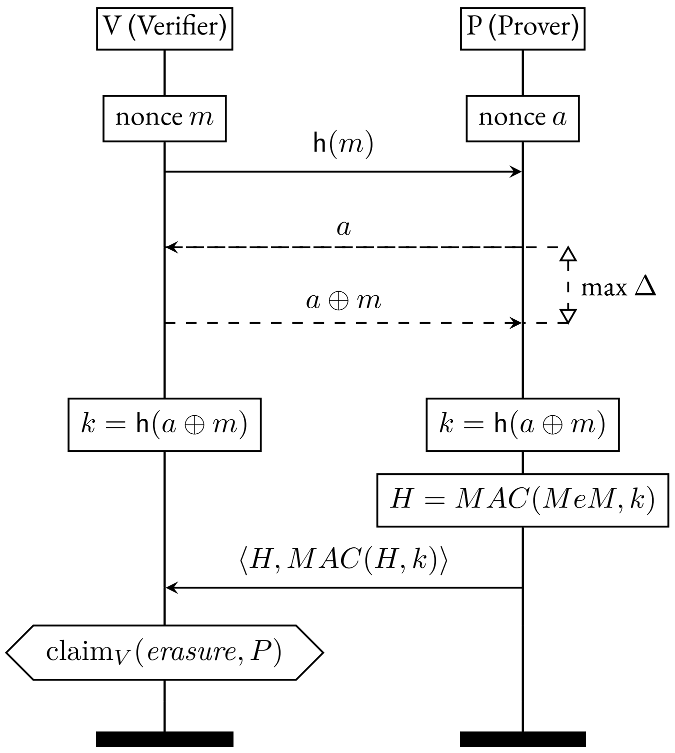

V (Verifier)
P (Prover)
nonce m
nonce a
$\mathsf{h}(m)$
a
max Δ
$a \oplus m$
$k = \mathsf{h}(a \oplus m)$
$k = \mathsf{h}(a \oplus m)$
$H = MAC(MeM, k)$
$\langle H, MAC(H, k)\rangle$
$\text{claim}_V(erasure, P)$

Figure 4.5: A representation of the SPEED protocol

tion and memory-erasure properties. All of these properties can be defined without the notion of time, and, as proved in the previous section, they are compatible with our equivalence results. All of them have the common characteristic of using proximity checks to (hopefully) assure that the communication partner is in the vicinity.

Most protocols that consider the distant-attacker assumption are key-exchange protocols. Two close agents without any previously shared data nor public key infrastructure need to communicate securely using a network that may be controlled by distant attackers. These protocols necessarily use some kind of asymmetric cryptography, given that the adversary will receive (with some delay) all the transmitted messages, and from that it should be impossible to deduce the secret shared keys.

Protocol	**Secrecy**		**Auth.**		**MemE**
	P	V	P	V	
DB-Based-Diffie-Hellman [123]	✓	✓	✓	✓	—
MedicalDB [117]	✓	✓	✓	✓	—
BluetoothJW-RTT [124]	✓	✓	✓	✓	—
Move2Auth-RTT [139]	✗	✓	—	✗	—
Amigo-RTT [135]	✗	✗	✗	✗	—
SPEED [9]	—	—	—	—	✗
DB-Based-Erasure-Protocol [129]	—	—	—	—	✓

Table 4.1: Analysis results

Table 4.1 shows that three out of five of the key exchange protocols analysed are correct. The other two, namely Move2Auth and Amigo, fail to ensure secrecy of the key-exchanged. Our analysis results on memory-erasure protocols coincide with those provided in [129], including the proof of correctness for the memory erasure protocol introduced there. We note, however, that [129] provides manual proofs, while our proofs are computer-generated.

4.5.6 Comments on the Tamarin encoding

Our Tamarin models need to mark adversary actions in the trace, as they are part of our security properties. We do so by creating rules representing a channel that can be used by honest agents and adversaries alike. When modelling all the network interactions using this channel, we faced non-termination issues. For this reason, we decided to restrict this channel usage to the messages directly related to the time measurement. This resulted in an over-approximation, in the sense that with this encoding some false attacks could appear, given that not all actions by adversaries are marked. That said, we manually checked this was not the case for any of the protocols analysed.

The use of the XOR operator made it difficult to analyse the security of several of the protocols, leading to non-termination for several proofs of our Tamarin models. Therefore, we decided to use a simplified user-defined XOR operator without the commutativity property, instead of the built-in one. This is not the first time this modelling decision is taken, as it was also previously used in the Tamarin models for distance-bounding protocols from [95]. The reason why protocols using proximity measurements through RTT containing the XOR operator face this non-termination issue is not known as far as we know. Nevertheless, we believe it could be solved, at least for these very specific cases, by some clever modelling or small improvements in the Tamarin prover, and leave it as future work.

4.6 Conclusions

In this chapter we identified and formalized the distant-attacker assumption, which has so far been used informally to establish the security requirements of various communication protocols for, e.g., IoT devices. We did so by introducing a time-based security model where round-trip-time measurements and the location of agents is used to determine whether the neighbourhood of an agent is free of attackers. To enable computer-aided verification of protocols written in our specification language, we provided a reduction of the time-based model by eliminating the notions of time and location, and defining proximity checks and the distant-attacker assumption as causal relations on the protocol events. We also introduced a class of security requirements that we proved hold in both the time-based and the causality-based model. Because the causality-based model is translatable to Tamarin, we were able to formally verify, for the first time, the security of five key-exchange protocols and two memory-erasure protocols, finding unreported vulnerabilities on three of them.

The results presented in this chapter can be extended in various ways. For example, the protocol specification language we use does not allow for conditionals and non-determinism, and only supports basic round-trip-time calculations. Hence, extending our results to richer specification languages would contribute to capturing a larger class of protocols. It is also worth generalizing our proofs to a causality-based specification model that is a subset of the model supported by a state-of-the-art protocol verification tool, such as Tamarin, since it would reduce the gap between theory and practice. Lastly, like previous verification frameworks for distance-bounding protocols, our methodology assumes that agents do not move. Dropping that assumption is of interest for both classes of protocols.

5 Background on computational analysis

5.1 Computational analysis

The computational analysis (also called computational complexity approach) of security protocols started in the 1980s with the seminal work of Goldwasser and Micali [72] on probabilistic encryption. It was later popularized by Bellare and Rogaway [22], which analysed key establishment protocols and provided the first formal model of the adversary capacities. These works provided strong assurances on the security properties of the protocols by using pen-and-paper proofs. Time has shown that it is often difficult to obtain correct security proofs even for relatively small protocols [37]. Still, for some protocols they have been achieved with great success. When protocols employ new cryptographic primitives, the symbolic analysis cannot, by definition, obtain security guarantees, as soundness proofs (in the computational model) are still needed.

Every major Internet application nowadays is secured by some form of cryptographic protocol. The usual path to the creation of new protocols is as follows. A group of researchers/practitioners creates a protocol aimed at solving a particular problem. For a certain period of time, the protocol is analysed by security researchers and practically implemented as prototypes. If the protocol uses new cryptographic primitives, several years are spent proving these primitives posses the strongest possible security properties in the computational model. In this period, cryptographers look for weaknesses by using cryptanalytic techniques. If the properties are valid and no attacks are found, the protocol gets standardized by international bodies such as Internet Engineering Task Force (IETF). Only at this point the protocol is considered secure and becomes widely available in applications. Unfortunately, most protocols turn out to be insecure, a fact that stresses the importance of provable security in the computational model.

Contrary to the symbolic model where the cryptographic primitives are considered secure, in the computational model their security is a central issue. In this model, messages are finite bitstrings, cryptographic primitives are functions from bitstrings to bitstrings, and attackers are interactive probabilistic Turing Machines. In general, the adversary is only limited by the amount of time and computational power available. This gives the model the highest flexibility from the mathematical standpoint. Still, it does not take into account implementation details (a provably secure protocol might be insecure against side-channel attacks), nor is capable of naturally model physical properties such as time, locations and machine failures.

Since the inception of computers in the last century, there have been thousands of cryptographic primitives proposed in the literature. The most well-known of these functions can be classified as encryption schemes (symmetric and asymmetric), one-way hash functions, digital signatures, mix networks, private information retrieval, commitments, pseudorandom generators, and many others[1].

To prove the primitives secure, researchers use mathematical properties of functions which guarantee the intuitive notion of security sought in applications. The properties are usually defined through the so-called cryptographic games, in which an attacker can interact with the primitive in a formally specified manner, and to win the game it must compute some value with a certain probability (for example discover the secret key). The probability usually depends on the security parameter of the game, which is usually the same as the length of the secret keys involved. A primitive is secure when it is proved that for any attacker the probability of winning the game is very small as a function of the security parameter.

One famous security property of encryption schemes is indistinguishability under adaptive chosen ciphertext attack (IND-CCA2), which is currently considered a basic requirement for this type of primitive. An encryption function takes as input a plaintext and a secret key, and outputs a ciphertext. Computing the plaintext from the ciphertext should not be computationally possible unless the secret key is known. The IND-CCA2 property basically states that an adversary does not distinguish a pair of ciphertexts, even if they can get the ciphertexts of as many plaintexts as they want in an adaptative fashion. There exists several functions which have been proved to possess this property. Unfortunately, this is not the case for other properties such as one-way, needed in the construction of cryptographic hash functions.

One-way functions have been the target of much research, but its existence is still an open conjecture. Intuitively, a one-way function is easy to compute on any input, but hard to invert given the image of a random input [122]. Cryptographic hash functions are usually approximated as one-way functions, as an important step to achieve proofs of higher level primitives which utilize one-way functions as building blocks. This approximation is formalized in the ROM [22]. In this thesis we aim to design a higher level primitive (secure erasure) using hash functions as building blocks. Therefore, we make our proofs in the ROM model which we formally introduce later in the chapter.

In this chapter we introduce the theoretical background needed to understand our new secure erasure protocols. This includes basic notions in the computational model such as security games and negligible functions. Then we introduce the ROM and show how it can be used.

5.2 Notation and basic definitions

We denote by $[n]$ the set $\{1, \ldots, n\}$, by $a \leftarrow\$ A$ the uniformly random sampling of a from the set A and by $o \leftarrow F(x, \ldots)$ the output of an algorithm F running on given inputs. For two bitstrings $a, b \in \{0,1\}^*$, we denote by $a \| b$ their concatenation. An adversary is a probabilistic polynomial time Turing machines, generally denoted by $\mathcal{A}$. Protocols usually posses security parameters, whose size intuitively determine the level of security of the primitives used.

[1] https://en.wikipedia.org/wiki/Category:Cryptographic_primitives

Definition 16 (negligible function)**.** A function $\mu\colon \mathbb{N} \to \mathbb{R}$ is negligible if for every positive polynomial of finite degree P there exists an integer $N_P > 0$ such that:

$$\forall x > N_P\colon |\mu(x)| < \frac{1}{P(x)}$$

We use $\mathsf{negl}(\cdot)$ to denote any single negligible function.

Definition 17 (secure protocol)**.** A protocol with security parameter λ is considered secure if and only if for every attacker $\mathcal{A}$, the probability that $\mathcal{A}$ breaks the protocol is a negligible function in λ.

We formalize security properties as a game against an adversary. During the game, the adversary can interact with the functions defined in the protocol in a predefined way. Each game has a winning condition, such as computing a secret key for secrecy properties. If the attacker is only able to win the game with negligible probability, then the security property is valid.

Next we show how to model the indistinguishability under chosen plaintext attack (CPA) property of an encryption scheme as a game:

Example 4 (Indistinguishability game)**.** Let $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be a symmetric encryption scheme and λ a security parameter. Consider the following experiment $\mathsf{CPA}_{\mathcal{A}}(\lambda)$ against an adversary $\mathcal{A}$:

1. a key $k \leftarrow \mathsf{Gen}(1^\lambda)$ is generated
2. $\mathcal{A}$ chooses a polynomial number of messages $\{m_1, \ldots, m_n\}$ and receives their encryptions $\{\mathsf{Enc}(m_1, k), \ldots, \mathsf{Enc}(m_n, k)\}$
3. $\mathcal{A}$ chooses two challenge messages c_0, c_1 of the same length
4. a bit $b \leftarrow\!\!\$\ \{0, 1\}$ is sampled and a challenge ciphertext $ct \leftarrow \mathsf{Enc}(c_b, k)$ is computed
5. $\mathcal{A}$ is given ct
6. $\mathcal{A}$ chooses a polynomial number of messages $\{m'_1, \ldots, m'_n\}$ and receives their encryptions $\{\mathsf{Enc}(m'_1, k), \ldots, \mathsf{Enc}(m'_n, k)\}$
7. $\mathcal{A}$ outputs a guess b'
8. if $b = b'$ the experiment returns 1, else it returns 0

The experiment above is restricted in the sense that c_0 and c_1 must be distinct and also must not belong to $\{m_1 \ldots, m_n\} \cup \{m'_1 \ldots, m'_n\}$. The encryption scheme is indistinguishable under CPA if for all $\mathcal{A}$:

$$\Pr[\mathsf{CPA}_{\mathcal{A}}(\lambda) = 1] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$$

Introduction to the ROM

The ROM, formalized by Bellare and Rogaway in [22], aimed to solve an issue with many cryptographic schemes that used hash functions as building blocks. For years cryptographers had found proofs of security that depended on the assumptions related to using hash functions as a black-box. This assumption was not generally well justified, and seen as trickery by some researchers. The objective of [22] was to solve this issues and push forward the idea that the ROM provided a bridge between the cryptographic theory and practice. Furthermore, they showed how the paradigm yields more efficient protocols while retaining the advantages of provable security.

Assume that a protocol uses a public hash function h as one of its primitives. Under the ROM, h is modelled as an oracle $\mathcal{O}$, which for each input replies a perfectly random bitstring, and if the same input is asked twice, the answer is the same. This means that the oracle implements a perfectly random function. It also implies that attackers cannot compute any (partial) value of h unless they ask the corresponding value to $\mathcal{O}$. Essentially, the hash function h is modelled as a perfectly random function that is accessible to protocol participants as a black-box.

A simple protocol that can be shown secure in the ROM is the following:

Example 5. Alice wants to show Bob that she knows the solution to a problem, without revealing the solution itself, as it will be officially published at a later time. To achieve this with certainty, they use a cryptographic protocol. The protocol starts when Bob sends Alice a random nonce n. Then Alice replies with $h(n\|s)$ where s is the solution and h is a cryptographic hash function. Later, when the solution s is published, Bob can check that indeed Alice knew the solution by checking that the value sent by Alice was indeed $h(n\|s)$. An attacker $\mathcal{A}$ would like to convince Bob that it also knows the solution s while it does not. The following game describes the intended security property. Let λ be a security parameter. Consider the following experiment $\mathrm{Exp}^{\mathcal{O}}_{\mathcal{A}}(\lambda)$:

1. random bitstrings n, s, n' of length λ are generated
2. $\mathcal{A}$ receives n, n' and $\mathcal{O}(n\|s)$
3. $\mathcal{A}$ interacts with $\mathcal{O}$ a polynomial number of times
4. $\mathcal{A}$ outputs a guess o
5. The experiment returns 1 if $o = \mathcal{O}(n'\|s)$, else returns 0

The protocol is secure if and only if $\forall \mathcal{A}\colon \Pr\left[\mathrm{Exp}^{\mathcal{O}}_{\mathcal{A}}(\lambda) = 1\right] < \mathsf{negl}(\lambda)$. The protocol above can be proved secure in the ROM as follows.

sketch. Assume $P(\lambda)$ is the number of queries that $\mathcal{A}$ makes to the oracle, where P is a polynomial. Consider the case where one of these queries equals $n\|s$, which happens with probability at most $\frac{P(\lambda)}{2^\lambda}$. In this case, the attacker deduces s and computes the correct guess. If none of the queries equals $n\|s$, then the attacker basically has no information about s, in which case the best it can do is to output a random guess o which would lead to a successful attack with probability $\frac{1}{2^\lambda}$. Applying the union bound to the probabilities in each cases above, the probability of success of the attacker is bounded by $\frac{P(\lambda)+1}{2^\lambda}$, which is a negligible function in λ, as was needed. $\square$

6 A provable secure memory-erasure protocol against distant attackers

A proof of secure erasure (PoSE) is a communication protocol where a verifier seeks evidence that a prover has erased its memory within the timeframe of the protocol execution. Designers of PoSE protocols have long been aware that, if a prover can outsource the computation of the memory-erasure proof to another device, then their protocols are trivially defeated. As a result, most software-based PoSE protocols in the literature assume that provers are isolated during the protocol execution, that is, provers cannot receive help from a network adversary. Our main contribution in this chapter is to show that this assumption is not necessary. We introduce formal models for PoSE protocols playing against provers aided by external conspirators and a PoSE protocol that we prove secure in this context. We reduce the requirement of isolation to the more realistic requirement that the external conspirator is further away from the verifier than the prover. Software-based protocols with relaxed isolation assumptions are especially pertinent for low-end devices, where it is too costly to deploy sophisticated protection methods.

6.1 Introduction

Internet of Things (IoT) devices are specially vulnerable to malware infection due to their ubiquity, connectivity and limited computational resources [106]. Once infected, an IoT device becomes both a victim and a useful weapon to launch further attacks on more advanced infrastructure and services. Detecting whether an IoT device is infected with malware is thus essential to maintaining a secure computer network. The challenge for the defender is operating within the resource constraints of IoT devices, which make them ill-suited for active security defences and health monitoring [101].

A pragmatic approach to ensure the absence of malware is secure erasure, consisting of putting a device back into a clean state by wiping out its memory. This approach was first introduced in [111] as a prerequisite for secure software update. Although memory-erasure can be achieved via direct hardware manipulation, here we are interested in Secure Erasure protocols (PoSE), whereby a verifier instructs a resource-constrained device, called the prover, to erase its memory and to prove that it indeed has done so.

A PoSE protocol is not a data erasure tool [74], despite both being aimed at erasing memory. The latter is meant to erase sensitive data from memory in an irreversible manner, i.e., in a way that is unrecoverable by advanced forensics techniques. The former is a lightweight communication protocol whereby a verifier attests whether a (possibly) compromised prover has filled its memory with random data.

Because provers are potentially infected with malware, PoSE protocols in general cannot rely on cryptographic secrets stored in the prover. The exception are protocols that rely on secure hardware [10], such as a Trusted Platform Module. Not all devices are manufactured with secure hardware, though. Examples are those designed for low price and low energy consumption. Hence, in this thesis, we don't assume secure hardware on provers and we seek a software-based solution.

In the absence of cryptographic secrets, software-based PoSE protocols have historically relied on the assumption that provers are isolated during the protocol execution, that is, provers cannot receive external help. The isolation assumption has been deemed necessary so far to prevent a malicious prover from outsourcing the erasure proof to another device. Ensuring isolation is cumbersome, though. It requires closing all communication channels between provers and potential conspirators, for example, by jamming or using a Faraday cage.

One of the goals of this thesis is to reduce the requirement of isolation to the more realistic requirement that the external conspirator (*distant-attacker*) is further away from the verifier than the prover (see Figure 6.1 for an illustration). That is, we do not restrict the communication capabilities of the (possibly corrupt) prover with the external adversary, but their position relative to each other. In practice, this can be accomplished relying on round-trip-time measurements, to ensure that the responses received to a sequence of challenges come from the device whose memory we aim to erase. Such measurements are used for example in distance-bounding protocols, whose goal is to ensure the authenticity of communication with a nearby device [67]. For ensuring our distant-attacker assumption, we can therefore build upon the principles of design in distance-bounding protocols, of which there exist already various proof-of-concepts and real-life implementations. While the messages exchanged in classic distance-bounding protocols typically consist of single bits [116], recent progress in the communication infrastructure allows fast exchange of longer messages. This was proved feasible in [31], assuming for example that attackers are using only commercial off-the shelf hardware. Furthermore, the relay resistant protection mechanism in the EMV protocol [62] and later improvements [115] measure the round-trip-time of 32-bit packets in order to bound distance. In the context of electric vehicle charging systems, [42] also considers a bigger than binary alphabet for their messages. In our proposed protocols, we will exploit this feature in order to challenge random blocks of the device's memory, rather than bits.

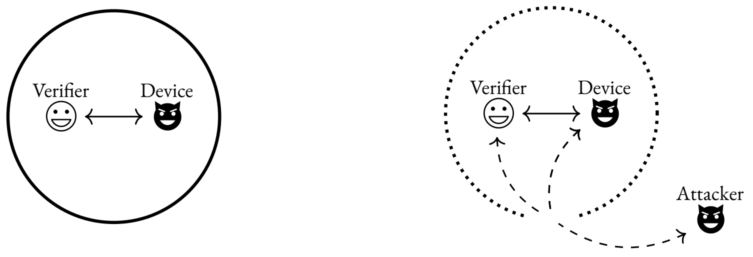


Figure 6.1: The isolation assumption (on the left) assumes no interference in the prover-verifier communication. The distant attacker assumption (on the right) lets the attacker interfere from far away.

To demonstrate that secure memory erasure is possible without assuming isolation, we introduce and formalize a class of PoSE protocols that employs a distance-bounding mechanism, which we call PoSE-DB, and put forward two protocols in this class together with formal security proofs against a corrupt prover that communicates with an external attacker. PoSE-DB protocols will have an interactive phase consisting of several challenge-response rounds. For each round, they will measure the round-trip-time between the challenge and response, and verify that the response is correct, aiming for two security goals. First, the prover cannot relay its communication with the verifier to the distant attacker without failing the distance-bounding check, compelling the prover to compute the responses locally. Second, if the prover locally computes the responses to the verifier's challenges, then it must have erased its memory even if aided by an external attacker. The first contribution of this part is a formalization of those security requirements (Section 6.2). The second one is the development of a PoSE-DB protocol with formal security proofs. The protocol (Section 6.4) is based on a straightforward idea, already used in the first secure erasure protocol proposed by Perito and Tsudik [111]: the verifier sends a random sequence of bits to the prover, who should store it in its memory; then the verifier queries random memory blocks to check that they are stored. We adapt this idea into a PoSE-DB protocol, where we query one random block per round, checking that the reply is correct and is received within the specified time bound. Our main contribution in this case is a formal security proof without resorting to the device isolation assumption. Although the protocol is simple, the proof is not trivial, and we introduce general proof methods in Section 6.3 that help the analysis by allowing to focus on a single challenge-response round.

6.2 A Formal Model for PoSE-DB protocols

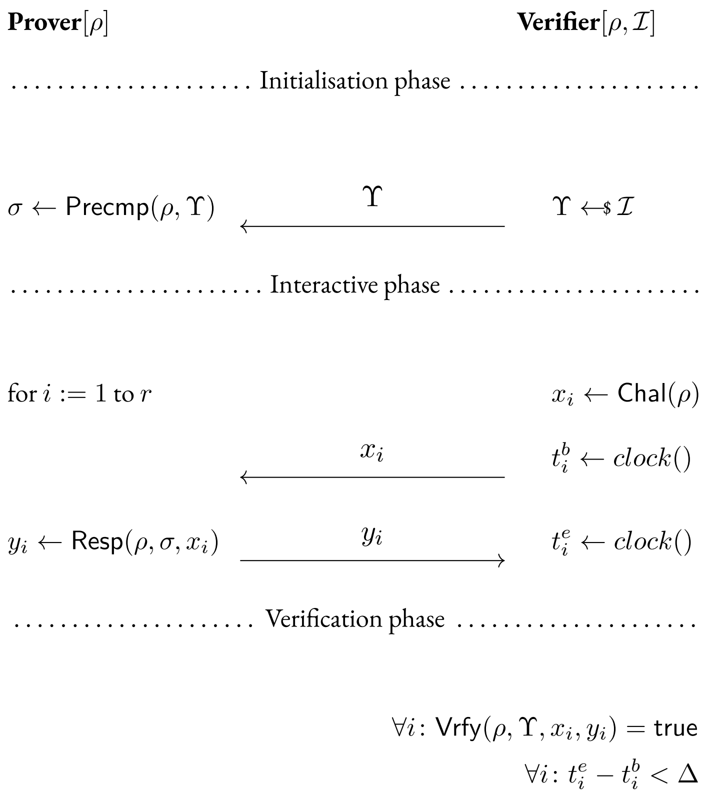
In this section, we introduce a class of memory-erasure protocols that aim to resist collusion between a corrupt prover and a distant-attacker, and a formal model that allows to prove their security. We call this class of protocols *Proofs of Secure Erasure with Distance-Bounding* (PoSE-DB).

6.2.1 Proof of Secure Erasure with Distance-Bounding

The key feature of a PoSE-DB protocol is the use of a distance-bounding mechanism over several challenge-response rounds to prevent the prover from outsourcing the erasure proof to the distant attacker. Figure 6.2 depicts the generic scheme of a PoSE-DB protocol. We consider the following protocol parameters as global constants: block size (w), size of memory in blocks (m), number of rounds (r) and time threshold (Δ). The size of the memory to be erased is therefore $m \cdot w$. The time threshold is chosen at deployment so that the distant-attacker assumption holds within round-trip-time bounded by Δ. In a setup phase which happens once before the protocol sessions, the verifier is instantiated with certain additional parameters necessary to run the protocol: the space used to draw initialization parameters for each session ($\mathcal{I}$, which could be a set of bitstrings or hash functions) and some auxiliary data (ρ) common for all sessions. Each PoSE-DB session then runs in three phases:

- *Initialization phase:* the local device (playing the role of the prover) has to perform a prescribed sequence of computation steps and store its result σ in its internal memory. Such a value is meant to fill the prover's memory, leaving no room for data previously stored in the device.

- *Interactive phase:* verifier and prover interact over a number of challenge/response rounds; the verifier measures the round-trip-time of those exchanges and stores all the challenges and their corresponding responses.
- *Verification phase:* the verifier accepts the proof if all challenge/response pairs from the interactive phase satisfy a prescribed verification test, and if the round-trip-times are below the time threshold Δ.



Prover$[\rho]$
Verifier$[\rho, \mathcal{I}]$
Initialisation phase
$\sigma \leftarrow \mathsf{Precmp}(\rho, \Upsilon)$
Υ
$\Upsilon \leftarrow\!\!\$\ \mathcal{I}$
Interactive phase
for $i := 1$ to r
$x_i \leftarrow \mathsf{Chal}(\rho)$
x_i
$t_i^b \leftarrow clock()$
$y_i \leftarrow \mathsf{Resp}(\rho, \sigma, x_i)$
y_i
$t_i^e \leftarrow clock()$
Verification phase
$\forall i\colon \mathsf{Vrfy}(\rho, \Upsilon, x_i, y_i) = \mathsf{true}$
$\forall i\colon t_i^e - t_i^b < \Delta$

Figure 6.2: PoSE-DB protocol session

Notice that there are no identities exchanged during the protocol, nor pre-shared cryptographic material. Like in existing software-based memory attestation and erasure protocols, we assume the existence of an out-of-the-band authentication channel, such as visual inspection, that allows the verifier to identify the prover. In Definition 18, we formally specify PoSE-DB protocols as a set of algorithms to be executed by the prover and the verifier. We do not specify the way in which messages are exchanged or the time verification step. These are handled by the security definition as described below, considering a Dolev-Yao model with a distant attacker that cannot act within the challenge-response round.

Definition 18. A *proof of secure erasure with distance-bounding* (PoSE-DB) protocol is defined by a tuple of algorithms (Setup, Precmp, Chal, Resp, Vrfy) and parameters (m, w, r, Δ) as illustrated in Figure 6.2. We have:

- $(\rho, \mathcal{I}) \leftarrow \mathsf{Setup}(m, w)$: computes some data ρ necessary to run the protocol and a parameter space $\mathcal{I}$ to be used for instantiating protocol sessions;
- $\Upsilon \leftarrow\$ \mathcal{I}$: sample data uniformly from $\mathcal{I}$ for each protocol session; some protocols may instantiate a hash function at this step;
- $\sigma \leftarrow \mathsf{Precmp}(\rho, \Upsilon)$: computes a value of size $m \cdot w$, to be stored in memory;
- $x \leftarrow \mathsf{Chal}(\rho)$: generates a uniformly random challenge;
- $y \leftarrow \mathsf{Resp}(\rho, \sigma, x)$: computes the response to the challenge. This should be a very-lightweight operation consistent with the design principles of distance-bounding, such as a lookup operation.
- $\mathsf{Vrfy}(\rho, \Upsilon, x, y)$: determines if y is the correct response to challenge x

Consider the simple idea (similar to Perito and Tsudik's protocol [111]) of filling the memory of the device with random data, then challenging it to return randomly chosen blocks of that data during the interactive phase. We call this the *unconditional PoSE-DB protocol*, as its security proof, in Section 6.4, does not rely on cryptographic assumptions. For such a protocol, the parameter space $\mathcal{I}$ is set to $\{0,1\}^{m \cdot w}$. This means that every protocol session will start with a random sequence of length $\{0,1\}^{m \cdot w}$, which is equal to the memory size of the prover. The complete specification of this protocol is as follows.

Definition 19 (The unconditional PoSE-DB protocol)**.**

- $\mathsf{Setup}(m, w)$: return $\rho = \emptyset$, $\mathcal{I} = \{0,1\}^{m \cdot w}$
- $\psi \leftarrow\$ \{0,1\}^{m \cdot w}$
- $\mathsf{Precmp}(\rho, \psi)$: parse ψ as $t_1 \| \dots \| t_m$ with $t_i \in \{0,1\}^w$, return $\sigma = t_1 \| \dots \| t_m$
- $\mathsf{Chal}(\rho)$: return $x \leftarrow\$ [m]$
- $\mathsf{Resp}(\rho, \sigma, x)$: return t_x, which was stored in σ
- $\mathsf{Vrfy}(\rho, \psi, x, y)$: return true if and only if $y = t_x$

Note that the erasure procedure itself running on the local device cannot be overwritten by σ. Hence, in practice, the device should allocate memory to store and execute the erasure procedure, and the goal should be for this procedure to introduce minimal memory overhead. This is a necessary condition in any memory erasure protocol. The erasure procedure for the unconditionally secure protocol consists in simply storing and fetching blocks from the memory. We expect its memory overhead to be minimal.

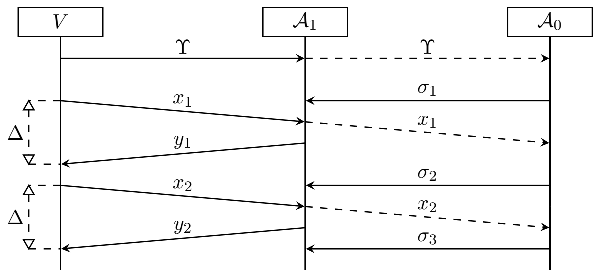


Figure 6.3: The y-axis denotes a timeline. Because A_0 is far, A_1 cannot relay the verifier's challenge to A_0 and wait for a response. A_0 does help A_1 in the other phases of the protocol execution.

6.2.2 Formalizing Secure Erasure Against Distant Attackers

To define secure erasure, we formally split the adversary in two: we use $\mathcal{A}_0$ to denote the distant-attacker, and $\mathcal{A}_1$ to denote the local (possibly corrupt) device. Like in a Dolev-Yao model, our adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ is in full control of the network and can corrupt agents. It can eavesdrop, inject and modify messages sent to the network. One limitation for the pair of attackers is given by the physical constraints of the communication medium, which are leveraged by the distance-bounding mechanism, and the assumption that $\mathcal{A}_0$ is distant, i.e. sufficiently far from the device. In our security definition we abstract away from the distance-bounding check by not letting $\mathcal{A}_0$ act between the sending of the challenge and the receipt of the corresponding response in each round. The intuition of this abstraction is illustrated in Figure 6.3: because $\mathcal{A}_0$ is far and the function Resp should be computed fast, $\mathcal{A}_0$ does not have time to respond to the verifier's challenge in time. We do allow $\mathcal{A}_0$, before each round of the fast phase, to precompute some state σ_i to be used by $\mathcal{A}_1$ during the ith round of the fast phase. Assuming that σ_i is computed by $\mathcal{A}_0$ is without loss of generality, since $\mathcal{A}_0$ is unbounded and has at least as much knowledge as $\mathcal{A}_1$, which forwards all information to $\mathcal{A}_0$. Therefore, we assume that right before the challenge x_i, the attacker's available memory on the device is filled by σ_i and this is the only information that $\mathcal{A}_1$ can use to compute the response in the ith round of the fast phase.

```
$(\rho, \mathcal{I}) \leftarrow \mathsf{Setup}(m, w)$;  $\Upsilon \leftarrow\!\!\$\ \mathcal{I}$
for $i := 1$ to $r$ do :
    $\sigma_i \leftarrow \mathcal{A}_0(1^w, \rho, \Upsilon, \{x_j | j < i\})$
    $x_i \leftarrow \mathsf{Chal}(\rho)$;  $y_i \leftarrow \mathcal{A}_1(1^w, \rho, \sigma_i, x_i)$
return $\forall i$: $\mathsf{Vrfy}(\rho, \Upsilon, x_i, y_i) = \mathsf{true}$
```

Figure 6.4: Security experiment $\mathsf{Exp}^{m,r,w}_{\mathcal{A}_0,\mathcal{A}_1}$.

Figure 6.4 formalizes the environment described above in the form of a security experiment. The attacker's restriction is formalized in Definition 20. The parameter M bounds the size of the memory used by $\mathcal{A}$ on the device throughout the protocol, i.e. the maximum value of σ_i in the security experiment. We consider an attacker successful if it passes the protocol while not erasing a significant proportion of the $m \cdot w$ bits of memory on the device. Assume the portion of memory that is needed by the adversary to store malware or any other information is y. If the adversary needs to use more than $m \cdot w - y$ bits to successfully execute the protocol, then the device becomes "clean", as the memory required by the attacker is erased. Therefore, the goal of the attacker is to use at most $M = m \cdot w - y$ bits of space during the protocol execution.

Definition 20. An adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ against the memory challenge game is *M-bounded* if and only if in any execution of $\mathsf{Exp}^{m,r,w}_{\mathcal{A}_0,\mathcal{A}_1}$ and any round i we have that $|\sigma_i| \leq M$.

To evaluate the security of a given PoSE-DB protocol, we will consider the class of M-bounded adversaries and determine the probability of any adversary from this class to win the security game.

Definition 21 (PoSE-DB security)**.** Assume some fixed parameters (m, r, w) for the experiment from Figure 6.4. An adversary $(\mathcal{A}_0, \mathcal{A}_1)$ *wins the memory-challenge game* with probability ζ if and only if $\Pr\left[\mathsf{Exp}^{m,r,w}_{\mathcal{A}_0,\mathcal{A}_1} = \text{true}\right] = \zeta$, where probability is taken over the randomness used by the experiment.

There are similarities between our security definition presented above and the definition of secure proof of space [58, 114], the main difference being that in proofs of space $\mathcal{A}_1$ is isolated from $\mathcal{A}_0$ in the interactive challenge phase.

6.3 Initial results

Before specifying and analysing our two PoSE-DB protocols, we provide useful initial results on the security of PoSE-DB protocols in general. Concretely, we show that PoSE-DB security increases proportionally to the number of rounds. This will allow us to simplify the security analysis by proving a level of security for the protocol executed in a single round, and then generically deriving the corresponding security guarantees over multiple rounds. Throughout all our proofs we consider deterministic adversaries $\mathcal{A}$. This is without loss of generality: since our security definition upper-bounds the success probability of $\mathcal{A}$, we can always consider that $\mathcal{A}$ is initialized with its best random tape. In this setting, by fixing its best-case random-tape, we can consider $\mathcal{A}$ deterministic. This idea is well known and has been applied elsewhere [133]. Note that we still have randomness left in the security experiment, coming from probabilistic choices in honest algorithms.

Let Υ be an element from the initialization space $\mathcal{I}$ of a PoSE-DB protocol. For an adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$, denote by $\Pr_{\Upsilon \leftarrow\$ \mathcal{I}}[\mathcal{A}^r]$ the probability that $\mathcal{A}$ wins the memory game with r rounds. For a fixed Υ, let the corresponding winning conditional probability be $\Pr[\mathcal{A}^r \mid \Upsilon]$. We say that an adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ playing the memory challenge game with fixed $\Upsilon \in \mathcal{I}$ is *uniform* if, for any sequence of challenges $(x_1, \ldots, x_r)$, $\mathcal{A}_0$ returns the same state σ_i in each round, i.e. $\sigma_1 = \cdots = \sigma_r$. The following lemma allows us to focus on uniform adversaries when proving the security of a PoSE-DB protocol. The main idea of the proof is that, since the challenge in each round is chosen independently, the best that $\mathcal{A}_0$ can do in any round is to choose a state σ_μ that maximizes the success probability of $\mathcal{A}_1$ for a random challenge.

Lemma 7. For any M-bounded adversary $\mathcal{A}$ against the PoSE-DB security experiment, there is a uniform M-bounded adversary $\bar{\mathcal{A}}$ that wins the experiment with at least the same probability: $\Pr_{\Upsilon \leftarrow\$ \mathcal{I}}[\mathcal{A}^r] \leq \Pr_{\Upsilon \leftarrow\$ \mathcal{I}}[\bar{\mathcal{A}}^r]$.

Proof. It is sufficient to prove that, for any fixed $\Upsilon \in \mathcal{I}$, we have $\Pr[\mathcal{A}^r \mid \Upsilon] \leq \Pr\left[\bar{\mathcal{A}}^r \,\middle|\, \Upsilon\right]$. In this case, since $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ is deterministic and Υ is fixed, for every i the state σ_{i+1} returned by $\mathcal{A}_0$ is uniquely determined by the sequence of challenges $x_1, \ldots, x_i$ in the experiment $\mathsf{Exp}^{m,r,w}_{\mathcal{A}_0,\mathcal{A}_1}$. Let Σ be the set of all possible states σ_i that can be output in any round by $\mathcal{A}_0$ over all possible sequences of challenges. We denote by $\Pr[\mathcal{A}_1 \mid \Upsilon, \sigma]$ the probability that $\mathcal{A}_1$ answers a single challenge correctly when given the state σ as input:

$$\Pr[\mathcal{A}_1 \mid \Upsilon, \sigma] = \Pr[\mathsf{Vrfy}(\rho, \Upsilon, x, y) \mid x \leftarrow \mathsf{Chal}(\rho), y \leftarrow \mathcal{A}_1(1^w, \rho, \sigma, x)]$$

where the probability is taken over the randomness used by the challenge generation algorithm. We define σ_μ to be the state in Σ for which the probability of winning given a random challenge is maximal $\sigma_\mu = \arg\max_{\sigma \in \Sigma} \Pr[\mathcal{A}_1 \mid \Upsilon, \sigma]$. Let $\bar{\mathcal{A}} = (\bar{\mathcal{A}}_0, \mathcal{A}_1)$, where $\bar{\mathcal{A}}_0$ always returns σ_μ, independent of the set of challenges it obtains as input. Note that $\bar{\mathcal{A}}$ is (M_0, q)-bounded, since the state returned by $\bar{\mathcal{A}}_0$ is among the possible states returned by $\mathcal{A}_0$, and the adversary $\mathcal{A}_1$ is the same. We will now show that $\bar{\mathcal{A}}$ achieves at least the same probability of success as $\mathcal{A}$. If $\bar{x}_k$ is a sequence of challenges $\{x_1, x_2, \ldots, x_k\}$ we denote by:

- $\Pr\left[\mathcal{A}^k \,\middle|\, \Upsilon, \bar{x}_k\right]$ the probability that the adversary $\mathcal{A}$ gets a correct answer in the first k rounds given that the first k challenges are $\bar{x}_k$. As $\mathcal{A}$ is deterministic, this probability is either 0 or 1.
- $\Pr[\mathcal{A}^t(\bar{x}_k) \mid \Upsilon, \bar{x}_k]$ the probability that the adversary $\mathcal{A}$ gets a correct answer in t successive rounds starting from round $k+1$ given that the challenges in the first k rounds are $\bar{x}_k$.
- $\Pr[\bar{x}_k \mid \Upsilon]$ the probability, taken over the randomness used by the challenge algorithm, that the first k challenges are $\bar{x}_k$.

For a vector of challenges $\bar{x}_k$ we have $\Pr[\mathcal{A}^1(\bar{x}_k) \mid \Upsilon, \bar{x}_k] = \Pr[\mathcal{A}_1 \mid \Upsilon, \sigma, \bar{x}_k]$ where $\sigma \leftarrow \mathcal{A}_0(\bar{x}_k)$ and:

$$\Pr[\mathcal{A}_1 \mid \Upsilon, \sigma, \bar{x}_k] = \Pr[\mathcal{A}_1 \mid \Upsilon, \sigma] \leq \Pr[\mathcal{A}_1 \mid \Upsilon, \sigma_\mu]$$

where the equality comes from the independence between A_1 and $\bar{x}_k$ given Υ, σ; the inequality follows by definition of σ_μ. Using the inequality above, we prove by induction that, for any $k \geq 0$, we have $\Pr\left[\mathcal{A}^{k+1} \,\middle|\, \Upsilon\right] \leq \Pr\left[\bar{\mathcal{A}}^{k+1} \,\middle|\, \Upsilon\right]$. The case $k = 0$ is trivial. For $k > 1$, we apply the induction hypothesis to obtain:

$$\begin{aligned}
&\Pr\left[\mathcal{A}^{k+1} \,\middle|\, \Upsilon\right] \\
&= \sum_{\bar{x}_k} \Pr[\bar{x}_k \mid \Upsilon] \cdot \Pr\left[\mathcal{A}^k \,\middle|\, \Upsilon, \bar{x}_k\right] \cdot \Pr\left[\mathcal{A}^1(\bar{x}_k) \,\middle|\, \Upsilon, \bar{x}_k\right] \\
&\leq \sum_{\bar{x}_k} \Pr[\bar{x}_k \mid \Upsilon] \cdot \Pr\left[\mathcal{A}^k \,\middle|\, \Upsilon, \bar{x}_k\right] \cdot \Pr[\mathcal{A}_1 \mid \Upsilon, \sigma_\mu] \\
&= \Pr\left[\mathcal{A}^k \,\middle|\, \Upsilon\right] \cdot \Pr[\mathcal{A}_1 \mid \Upsilon, \sigma_\mu] \\
&\leq \Pr\left[\bar{\mathcal{A}}^k \,\middle|\, \Upsilon\right] \cdot \Pr[\mathcal{A}_1 \mid \Upsilon, \sigma_\mu] = \Pr\left[\bar{\mathcal{A}}^{k+1} \,\middle|\, \Upsilon\right]
\end{aligned}$$

□

Next, we show that there is a direct relationship between the winning probability of an adversary in one round and the winning probability in multiple rounds. The analysis is simplified by the fact that we can focus on uniform adversaries, according to Lemma 7. If the initialization parameters are fixed, it follows immediately that running the experiment for r rounds exponentially decreases the cheating ability of the adversary.

Lemma 8. For any uniform adversary $\mathcal{A}$ and any $\Upsilon \in \mathcal{I}$, we have $\Pr[\mathcal{A}^r \mid \Upsilon] = \Pr[\mathcal{A}^1 \mid \Upsilon]^r$.

Proof.

$$\Pr[\mathcal{A}^r \mid \Upsilon] = \Pr[\mathcal{A}^r \mid \Upsilon, \sigma \leftarrow \mathcal{A}_0] = \Pr[\mathcal{A} \mid \Upsilon]^r$$

□

When lifting this lemma to uniformly chosen parameters from $\mathcal{I}$, it will be necessary to account for the fact that the (one-round) adversary may be lucky on a small proportion of those random choices. The following definition and proposition tolerate this, by bounding the success probability of the adversary only for a subset of good parameters, and showing the effect after r rounds.

Definition 22. For a set $\mathcal{I}_{\mathsf{good}} \subseteq \mathcal{I}$ and a value $\zeta < 1$, we say that $\mathcal{A}$'s *winning probability is bounded by ζ within $\mathcal{I}_{\mathsf{good}}$* if and only if $\forall \Upsilon \in \mathcal{I}_{\mathsf{good}} : \Pr[\mathcal{A}^1 \mid \Upsilon] \leq \zeta$.

If the proportion of cases $\mathcal{I} \setminus \mathcal{I}_{\mathsf{good}}$ in which the adversary is lucky is negligible, then for a large enough number of rounds the success probability of the adversary is also negligible, as shown by the next proposition.

Proposition 2. Given $\zeta < 1, \mathcal{I}_{\mathsf{good}} \subseteq \mathcal{I}$ and a uniform adversary $\mathcal{A}$ whose winning probability is bounded to ζ within $\mathcal{I}_{\mathsf{good}}$, we have $\Pr_{\Upsilon \leftarrow\$ \mathcal{I}}[\mathcal{A}^r] \leq \zeta^r + \frac{|\mathcal{I} \setminus \mathcal{I}_{\mathsf{good}}|}{|\mathcal{I}|}$.

Proof. From Lemma 8, we have:

$$\Pr_{\Upsilon \leftarrow\$ \mathcal{I}}[\mathcal{A}^r] = \frac{1}{|\mathcal{I}|} \sum_{\Upsilon \in \mathcal{I}} \Pr[\mathcal{A}^r \mid \Upsilon] = \frac{1}{|\mathcal{I}|} \sum_{\Upsilon \in \mathcal{I}} \Pr[\mathcal{A} \mid \Upsilon]^r$$

To reduce notation in what follows, we let $p_\Upsilon = \Pr[\mathcal{A} \mid \Upsilon]$ and omit $\Upsilon \in \mathcal{I}$ when we sum over all Υ in $\mathcal{I}$. We have:

$$\begin{aligned}
\sum_{\Upsilon} \Pr[\mathcal{A} \mid \Upsilon]^r &= \sum_{\Upsilon} p_\Upsilon^r = \sum_{\Upsilon, p_\Upsilon > \zeta} p_\Upsilon^r + \sum_{\Upsilon, p_\Upsilon \leq \zeta} p_\Upsilon^r \\
&\leq \sum_{\Upsilon,\, p_\Upsilon > \zeta} 1 + \sum_{\Upsilon} \zeta^r \leq \sum_{\substack{\Upsilon,\, p_\Upsilon > \zeta \\ \Upsilon \in \mathcal{I}_{\mathsf{good}}}} 1 + \sum_{\substack{\Upsilon,\, p_\Upsilon > \zeta \\ \Upsilon \notin / I_{\mathsf{good}}}} 1 + \sum_{\Upsilon} \zeta^r \\
&\leq 0 + |\mathcal{I} \setminus \mathcal{I}_{\mathsf{good}}| + |\mathcal{I}| \cdot \zeta^r
\end{aligned}$$

where for the last inequality we use that $\mathcal{A}$'s winning probability is bounded to ζ within $\mathcal{I}_{\mathsf{good}}$ to deduce that the first sum is empty and that the second one has at most $|H \setminus H_{\mathsf{good}}|$ terms. Combining the two results above, we deduce $\Pr_{\Upsilon \leftarrow\$ \mathcal{I}}[\mathcal{A}^r] \leq \zeta^r + \frac{|\mathcal{I} \setminus \mathcal{I}_{\mathsf{good}}|}{|\mathcal{I}|}$ as claimed. □

6.4 The unconditional PoSE-DB protocol

This section provides the security analysis for the unconditional PoSE-DB protocol presented in Section 6.2, Definition 19. Notice that the adversary $\mathcal{A}_1$ is memory bounded, but may do any amount of computation. Recall that the value $y = m \cdot w - M$ is the amount of memory that the adversary cannot erase (where it may store malware).

Before proving Theorem 2, we need some preliminary definitions and results. Let

$$S_c = \sum_{j=0}^{c} \binom{m}{j} (2^w - 1)^j$$

for $c \geq 0$. This coincides with the cardinality of a Hamming sphere over an alphabet of size 2^w as defined in [40].

Lemma 9. If $0 \leq c < m$ and $2^w \geq m + 3$ then $m(m+1)\binom{m}{c}(2^w - 1)^c \geq 2^w \cdot S_{c-1}$.

Proof. By induction on c. The base case $c = 0$ follows directly. We need to prove the induction step: $m \cdot (m+1) \cdot \binom{m}{c+1} \cdot (2^w - 1)^{c+1} \geq 2^w \cdot S_c$. First we prove:

$$\begin{aligned}
& m(m+1)\binom{m}{c+1}(2^w - 1)^{c+1} \\
& \geq m(m+1)\binom{m}{c}(2^w - 1)^c + 2^w \binom{m}{c}(2^w - 1)^c \\
\iff & \frac{m \cdot (m+1)}{c+1} \cdot (2^w - 1) \geq \frac{m \cdot (m+1)}{m-c} + \frac{2^w}{m-c} \\
\iff & \frac{m \cdot (m+1) \cdot (m-c)}{c+1} \cdot (2^w - 1) \geq m \cdot (m+1) + 2^w
\end{aligned}$$

which follows from $\frac{m \cdot (m+1) \cdot (m-c)}{c+1} \cdot (2^w - 1) \geq m \cdot (m+1) + 2^w$. We conclude:

$$\begin{aligned}
& m \cdot (m+1) \cdot \binom{m}{c+1} \cdot (2^w - 1)^{c+1} \\
& \geq m \cdot (m+1) \cdot \binom{m}{c} \cdot (2^w - 1)^c + 2^w \cdot \binom{m}{c} \cdot (2^w - 1)^c \\
& \geq 2^w \cdot S_{c-1} \cdot (2^w - 1)^{c-1} + 2^w \cdot \binom{m}{c} \cdot (2^w - 1)^c \\
& = 2^w \cdot S_c
\end{aligned}$$

□

Lemma 10. If $y \geq m + w$ and c is maximal s.t. $S_c \leq 2^y$, then $c \geq \left\lceil \frac{y-m-w+1}{w} \right\rceil$.

Proof. Assume the contrary, $c < \left\lceil \frac{y-m-w+1}{w} \right\rceil \implies (c+1) \cdot w + m \leq y$. Then:

$$
\begin{aligned}
S_{c+1} &= \sum_{j=0}^{c+1} \binom{m}{j} \cdot (2^w - 1)^j \leq \sum_{j=0}^{c+1} \binom{m}{j} \cdot (2^w - 1)^{c+1} \\
&\leq 2^m \cdot (2^w - 1)^{c+1} < 2^{m+w\cdot(c+1)} \leq 2^y
\end{aligned}
$$

which is a contradiction, as c was the largest integer with this condition. □

Denote by $d(\psi, \psi')$ the number of blocks in which two bitstrings of size $m \cdot w$ differ. For a set $R \subseteq \{0,1\}^{n \cdot w}$, let $d_c(R, \psi')$ the number of $\psi \in R$ s.t. $d(\psi, \psi') \geq c$.

Lemma 11. If $c \geq 1, \psi' \in \{0,1\}^{m\cdot w}, R \subseteq \{0,1\}^{m\cdot w}$ then $d_c(R, \psi') \geq |R| - S_{c-1}$.

Proof. If $|R| \leq S_{c-1}$ the claim is trivial. Assume $|R| > S_{c-1}$. Notice that there are exactly $\binom{m}{j}(2^w - 1)^j$ bitstrings r of size $m \cdot w$ such that $d(\psi, \psi') = j$. Then, there are at most $\sum_{j=0}^{c-1} \binom{m}{j}(2^w - 1)^j = S_{c-1}$ bitstrings in R such that they differ from ψ' in less than c blocks. □

Lemma 12. If $c \geq 1, y \geq 0$ are integers, $\cup_{i=1}^{k} R_i$ is a partition of $\{0,1\}^{m\cdot w}$ with $k \leq 2^{m\cdot w - y}$, and $\psi_i' \in \{0,1\}^{m\cdot w}$, then $\sum_{i=0}^{k} d_c(R_i, \psi_i') \geq 2^{m\cdot w - y} \cdot (2^y - S_{c-1})$.

Proof. Without loss of generality assume $k = 2^{m\cdot w - y}$ (we can always add empty sets to the partition). By Lemma 11, $d_c(R_i, \psi_i') \geq |R_i| - S_{c-1}$. Adding for all R_i:

$$
\sum_{i=0}^{k} d_c(R_i, \psi_i') \geq \sum_{i=0}^{k} (|R_i| - S_{c-1}) = 2^{m\cdot w} - k \cdot S_{c-1} = 2^{m\cdot w - y} \cdot (2^y - S_{c-1})
$$

□

Theorem 2. Assume that the unconditional PoSE-DB protocol is instantiated with parameters $(m, 1, w)$. Let $\mathcal{A}$ be any adversary with measure (M, ∞). Then, there exists a set $\mathcal{I}^1_{\mathsf{good}} \subseteq \mathcal{I}$ such that $\mathcal{A}$'s winning probability is bounded to $1 - m^{-1}$ within $\mathcal{I}^1_{\mathsf{good}}$ and $\left|\mathcal{I}^1_{\mathsf{good}}\right| \geq 2^{m\cdot w} \cdot (1 - 2^{-y})$. Furthermore, if $y \geq m + w$, then there exists a set $\mathcal{I}^2_{\mathsf{good}} \subseteq \mathcal{I}$ such that $\mathcal{A}$'s winning probability is bounded to $1 - \left\lceil \frac{y - m - w + 1}{w} \right\rceil m^{-1}$ within $\mathcal{I}^2_{\mathsf{good}}$ and $\left|\mathcal{I}^2_{\mathsf{good}}\right| \geq 2^{m\cdot w} \cdot (1 - (m^2 + m) \cdot 2^{-w})$.

Proof. Let σ be a bitstring of size M. Let $R_\sigma = \{\psi_1, \ldots \psi_k\}$ be the set of all bitstrings such that $\mathcal{A}_0(1^w, \rho, \psi_i) = \sigma$. We will lower-bound the number of queries related to elements in R_σ for which $\mathcal{A}_1$ gives an incorrect answer. To prove the first result, we count for how many bitstrings there is at least one error. Call this set $\mathcal{I}^1_{\mathsf{good}}$. Let ψ' be the concatenation of the blocks in $\{\mathcal{A}_1(1^w, \rho, \sigma, 1), \ldots, \mathcal{A}_1(1^w, \rho, \sigma, m)\}$, i.e. a bitstring of size $m \cdot w$. Since ψ' can be equal to at most one string in R_σ, $\mathcal{A}_1$ is wrong for at least one input $q \in \{1, \ldots, m\}$ for at least $|R_\sigma| - 1$ of the bitstrings in R_σ. Summing up for all possible sets R and for all possible σ, we deduce that

$\mathcal{A}_1$ is wrong for at least one query with respect to at least $2^{m\cdot w} - 2^{m\cdot w-y}$ bitstrings (as there are at most $2^{m\cdot w-y}$ different sets R_σ). We obtain $\left|\mathcal{I}^1_{\mathsf{good}}\right| \geq 2^{m\cdot w}\cdot(1-2^{-y})$ and:

$$\forall r \in \mathcal{I}^1_{\mathsf{good}}\colon \Pr_{q \leftarrow\$[n]}[\mathcal{A}_1(1^w,\rho,\sigma,q) \text{ is correct} \mid \sigma] \leq 1-m^{-1}$$

where $\sigma \leftarrow \mathcal{A}_0(1^w,\rho,\psi)$.

Next we prove the second result. Assume that $y \geq m+w$ and $2^w \geq m+3$. For each R_σ as in the proof for the first result, we will be interested in lower-bounding the number of errors that $\mathcal{A}_1$ makes while answering the queries of a challenger using some $\psi \in R_\sigma$. Notice this is exactly $d(R_\sigma,\psi')$ defined above. Clearly the set of all possible R is a partition of $\{0,1\}^{m\cdot w}$ with size at most $2^{m\cdot w-y}$. Let c be the largest integer such that $S_c \leq 2^y$ holds. By Lemma 10, we have that $c \geq \left\lceil \frac{y-m-w+1}{w}\right\rceil$. Let $\mathcal{I}^2_{\mathsf{good}}$ be the set of all substrings ψ such that $\mathcal{A}_1$ makes at least c errors while answering queries about ψ. From Lemma 12 we deduce that $\left|\mathcal{I}^2_{\mathsf{good}}\right| \geq 2^{m\cdot w-y}\cdot s$, where $s = 2^y - S_{c-1}$. By Lemma 9:

$$\begin{aligned}2^{m\cdot w} &= 2^{m\cdot w-y}(S_{c-1}+s) \leq 2^{m\cdot w-y}\left(m(m+1)\cdot 2^{-w}\binom{m}{c}\cdot(2^w-1)^c+s\right)\\ &\leq 2^{m\cdot w-y}\cdot(m(m+1)\cdot 2^{-w}+1)\cdot s \implies \left|\mathcal{I}^2_{\mathsf{good}}\right| \geq 2^{m\cdot w}(1-m(m+1)2^{-w})\end{aligned}$$

Then $\mathcal{A}_1$ makes $\left\lceil\frac{y-m-w+1}{w}\right\rceil$ errors for $2^{m\cdot w}\cdot(1-(m^2+m)\cdot 2^{-w})$ bitstrings. □

From Proposition 2 and Theorem 2, we obtain:

Corollary 2. If we execute the unconditional PoSE-DB protocol for r rounds in presence of any M-bounded adversary, then $\Pr\left[\mathsf{Exp}^{m,r,w}_{\mathcal{A}_0,\mathcal{A}_1} = \mathsf{true}\right] \leq (1-m^{-1})^r + 2^{M-m\cdot w}$. Furthermore, if $M \leq m\cdot w - m - w$ the bound improves to $\left(1-\left\lceil\frac{m\cdot w-m-w-M+1}{w}\right\rceil\cdot m^{-1}\right)^r + m\cdot(m+1)\cdot 2^{-w}$.

For example, if we would like to erase a malware of size at least 6 KB from a device with total memory 100 KB, then the parameters would be $w = 256, m = 100\cdot 2^{13}/256 = 3200, M = (100-6)\cdot 2^{13} = 770048$. If we wanted to be certain of erasure with probability 10^{-3}, the first bound tells us we need to execute the protocol for 22102 rounds, while the second bound for 121 rounds. In general, the second bound is better, but it can only be applied when the size of the malware is big enough. For example, on this device, we have $m+w = 3456$, which is less than 0.5 KB; so we can apply the bound to erase any malware of greater size.

6.5 Conclusions

In this chapter we proposed a secure memory-erasure protocol based on distance bounding techniques, and proved it secure against distant attackers. Furthermore, we showed how to reduce the security analysis of this type of protocol to the case of a single round in the interactive phase. These results will also be useful in Chapter 8, where we propose another PoSE-DB that reduces the communication complexity during the initialization phase.

7 The ex post facto argument revisited

In this chapter we introduce the pebbling games and the ex post facto argument, which are the core of our new erasure protocol in Chapter 8. First we define pebbling games, a combinatorial problem which is played on graphs and has found numerous applications in computational complexity. Then, we explain the ex post facto argument, a technique has been used to translate lower bounds on pebbling games into more general computational bounds. Finally, we review a well known result in this setting and prove that the originally stated bound is not accurate.

7.1 Pebbling games

Pebbling games are combinatorial games where a single player puts pebbles in the nodes of a directed graph. In this thesis we focus on the original version of these games, the black pebbling game, in which there is just one type of pebbles. Next, we introduce some graph definitions and notations that will be necessary to understand these games.

A graph G contains a set of nodes $V(G)$ (also called vertices) and a set of pair of nodes $E(G)$ (also called edges). If the edges are directed (which means that $(a,b) \neq (b,a)$), then the graph is directed. A cycle in a graph is a sequence of nodes and edges $(v_1, e_1 \ldots, v_k, e_k)$ such that for each $i \in \{1,\ldots,k-1\}$ we have $e_i = (v_i, v_{i+1})$, and $e_k = (v_k, v_1)$. A graph is acyclic if it contains no cycles. The successors of a node a are all vertices b such that the edge (a,b) exists. Similarly, the predecessors of a are all vertices b such that the edge (b,a) exists. Given a directed acyclic graph (DAG) G, the list of successors and predecessors of the node v in G are respectively $\Gamma^+(v)$ and $\Gamma^-(v)$. If $\Gamma^-(v) = \emptyset$ then v is an input node, and $\Gamma^+(v) = \emptyset$ then v is an output node. Next we define the black pebbling game as in [70].

Definition 23 (Black pebbling game)**.** The black pebbling game is played on a degree-bounded directed acyclic graph (DAG) G by placing a number of tokens (a.k.a. black pebbles) on the vertices of G according to the following rules:

- At each time step, a pebble may be added to an unpebbled vertex, or removed from a pebbled vertex.
- A pebble can be placed on a vertex only if all its predecessors are pebbled. Thus, a pebble can be placed in a vertex without incoming edges at any time.
- A pebble can be removed from a vertex at any time.

The game starts with all nodes without pebbles and ends when a given output vertex in G is pebbled. The maximum amount of pebbles used at any point during the game $\mathcal{P}$ is called pebbling complexity and denoted by $\mathsf{peb}(\mathcal{P})$. A game with n steps is represented by a sequence of subsets of nodes in the graph $(\mathbb{P}_0, \cdots, \mathbb{P}_n)$, where the index represents time steps, $\mathbb{P}_0 = \emptyset$ and $\mathbb{P}_n$ contains the output vertex. The pebbling complexity of this game is thus equal to $\max_{0 \leq i \leq n} |\mathbb{P}_i|$ and its length $|\mathcal{P}|$ equals the number of time steps n.

We say that subset of nodes S was pebbled during the game when $\forall v \in S \, \exists i \in [n] \colon v \in \mathbb{P}_i$, i.e. each node in S was pebbled at least once during the game. A conditional pebbling game is a game for which the starting set of pebbled nodes is arbitrary and at the end the output node does not need to be pebbled.

Pebbling games have been widely studied, and many ground-breaking results have been obtained as a result. It is known that the pebbling complexity of any graph with n vertices is at most $\mathcal{O}(\frac{n}{\log(n)})$ [77]. The relation between the pebbling complexity and the length of the game may have sharp changes [119]: there exist graphs and games with minimum length for a given pebbling complexity $\mathcal{P}_0, \mathcal{P}_1$ with $\mathsf{peb}(\mathcal{P}_0) = s$, $\mathsf{peb}(\mathcal{P}_1) = s + 1$, $|\mathcal{P}_1| = \mathsf{poly}(n)$ and $|\mathcal{P}_0| = \mathcal{O}(2^n)$. Computing the pebbling complexity of a graph is known to be a PSPACE-$\mathsf{complete}$ problem [69].

Pebbling results depend on properties of the graphs where the game is played. Superconcentrator graphs are a family of graphs with many applications in this area.

Definition 24. [Superconcentrator [110]] A directed acyclic graph with bounded indegree, n inputs, and n outputs is an n-superconcentrator if for any $1 \leq k \leq n$ and any set $\{s_1, \ldots, s_k\}$ of inputs and $\{t_1, \ldots, t_k\}$ of outputs there are k vertex-disjoint paths connecting the s_i to the t_i.

The superconcentrator property has led to many interesting pebbling results. A few classic pebbling results that can be easily proved for graphs with this property follow:

Lemma 13 ([110]). Suppose that G is an N-superconcentrator, $\mathbb{P}$ is a subset of nodes with pebbles and $|\mathbb{P}| \leq s$, and Z is a set of strictly more than s outputs in G. Then at least $N - s$ inputs in G have completely pebble-free paths to nodes in Z.

Lemma 14 (basic lower bound argument (BLBA) [88]). Suppose that $a \leq b$ are two indices and $\mathcal{P} = (\mathbb{P}_a, \mathbb{P}_{a+1}, \cdots, \mathbb{P}_b)$ is a conditional pebbling game on an N-superconcentrator G such that $|\mathbb{P}_a| \leq s_a$, $|\mathbb{P}_b| \leq s_b$, and $\mathcal{P}$ pebbles at least $s_a + s_b + 1$ output nodes from G. Then $\mathcal{P}$ pebbles and unpebbles at least $N - s_a - s_b$ different input nodes.

For an in-depth survey of pebbling results and applications in Proof Complexity see the excellent survey by Nordström in [107].

7.2 The *ex post facto* argument revisited

In this section we revise a lemma relating pebbling games with computational bounds from [57]. First we show that the result is not valid as originally stated. Then we explore one possible fix for the lemma, which enforces a more realistic adversary. For this case we also find a counter example, but this one required an exponential number of operations. We leave as an open problem if the modified lemma is valid if adversaries are polynomially bounded.

Dwork et al. proposed in [57] a very influential technique that allows to translate computational bounds from a graph-based game to a computational model, referred to as the *ex post facto argument*. The original application was the creation of easy-to-check proofs of computational effort, which is essential to secure decentralizing payment systems and thwart denial of service attacks, such as spamming. Their observation was that functions whose computational time is dominated by the number of memory accesses (memory-bound functions) are not affected by the progress in computational power as much as functions whose computational time is dominated by CPU access (CPU-bound functions). The reason being that CPU speed has grown at a faster rate than access speed to memory.

Memory-bound functions, and their analysis via the *ex post facto* argument, have seen applications in proofs of space [13, 58], proofs of sequential work [92], memory hardness [7] and secure erasure [81]. Notably, these works focus on functions built over a labelled directed graph whose input is the set of vertices with in-degree zero and whose output is the label of the output node (with out-degree zero).

Definition 25 (graph labelling function)**.** Given a hash function h and a DAG G with set of vertices $V(G)$ and set of edges $E(G)$, a labelling function $\ell\colon V(G) \to \{0,1\}^w$ is defined by,

$$\ell(v) := h(v \| l(v_1) \| \dots \| \ell(v_d))$$

where $\Gamma^-(v) = (v_1, \dots, v_d)$. If v has no predecessor, then $\ell(v) = h(v)$. We let $\ell^-(v) := v \| \ell(v_1) \| \dots \| \ell(v_d)$.

To analyse the complexity of an algorithm wishing to compute a labelling function, the *ex post facto* argument considers a computational model where algorithms are memory bounded and have access to a hash function modelled as a random oracle. The latter means that an algorithm cannot compute the output of the hash function, unless it calls the random oracle [22].

The *ex post facto* argument establishes a relation between the time, measured in terms of the number of calls to the random oracle, and memory, required when computing a labelling function and the strategy used to play the black pebbling game on the same graph, which was defined in Definition 23. Next we define the *ex post facto* argument, following [57].

Definition 26 (The *ex post facto* argument)**.** Given the execution of an algorithm A with access to an oracle, the *ex post facto* pebbling game associated with this execution is defined as follows:

- Placing initial pebbles: At the beginning, some nodes in the graph may contain a pebble, as part of the initial configuration. Which are these nodes is determined by the following. If the label of node v is used as part of a call to the oracle to compute another label, but the label of v was not computed before, then there is a pebble in v in the initial configuration.
- Placing a pebble: If A asks the oracle the label of a node, then place a pebble on the node.
- Removing a pebble: A pebble is removed as soon as it is not needed any more. Here we use our clairvoyant capabilities (i.e. the whole execution of the algorithm is known) to minimize the number of pebbles used during the game.

We say that the algorithm asks (the oracle) for the label of node v when it calls the oracle on input $\ell^{-}(v)$. If v has a predecessor v', then we say the label of node v' was used in the call to the oracle (notice that by construction $\ell(v')$ is part of $\ell^{-}(v)$).

In the original article the adversary's architecture consists of a cache with s words of size w and a memory of size z. Our results are independent of such division. Hence, for simplicity, we consider algorithms with $s \cdot w$ bits of memory. The output of the hash function is also of size w.

Lemma 15 (Lemma 1 in [57], simplified)**.** Consider an algorithm that operates for a certain number of steps and uses at most $s \cdot w$ bits of memory. Then, with probability $1 - 2^{-w}$, the maximum number of pebbles at any given point in the corresponding *ex post facto* pebbling game is bounded by s.

A counter example

Remark 1. Lemma 15 does not hold.

Proof. Our counterexample is an algorithm for which with probability 1 the number of pebbles in the *ex post facto* pebbling game is greater than s. Let $k > 1$ be a positive integer such that $s > k \cdot w$. Consider the graph G with n disjoint copies of a directed edge, where $n = s - 1 + k$, and an algorithm that computes the labels of the output nodes of G as follows:

1. Compute the labels of the input nodes in G one by one, but for each label do not store the last bit. This leaves w bits unused out of the $s \cdot w$ bits of memory available, which can be calculated as follows.

$$\begin{aligned} n \cdot (w-1) + w &= (s - 1 + k) \cdot (w - 1) + w \\ &= s \cdot w + k \cdot w - (s + k - 1) \leq s \cdot w - k < s \cdot w \end{aligned}$$

2. Compute the labels of the output nodes in G one by one, where for each label two oracle calls are needed, as we need to try the two possible values because we did not store the last bit of the label of the predecessor. These labels are stored in the w bits that were unused (the computation overwrites the previous value each time a new label is computed).

In the *ex post fact* pebbling game, Step 1 of the algorithm corresponds to putting pebbles in all input nodes, none of which is removed until the start of Step 2. During Step 2, the output nodes are pebbles in order, and pebbles from the output node and its predecessor input node are erased (as they are not needed afterwards). Therefore, the algorithm above satisfies that its *ex post facto* pebbling game uses at least n pebbles right before Step 2. We finish the proof by noticing that $n = s - 1 + k > s$, so the pebbling uses more pebbles that words of memory, as claimed. □

Notice that the above counter example uses the fact that the algorithm, as stated, does not have a target value, i.e. it does not need to compute a specific value of the labelling function correctly. We could try to make the previous lemma valid by restricting the algorithm as follows. Let $\mathcal{A}$ be an algorithm that aims to compute the label of the output node of a graph. The algorithm is successful when it computes the label of the output node correctly, and outputs this value right before finishing its execution.

Remark 2. Lemma 15 is incorrect even if we demand that the algorithm is successful with probability 1, i.e. always computes correctly the label of the output node.

Proof. Assume that $s \geq k \cdot w$ for some $k > 4$. Consider the graph G with n disjoint copies of a directed edge $a_i \to b_i$ for $11 \leq i \leq n$, where $n = s - 4 + k$, and edges $b_i \to b_{i+1}$ for $1 \leq i \leq n + 1$ and an algorithm that pebbles the output node b_{n+2} of G as follows:

1. Compute the label of node b_{n+1} by using 3 words of memory, and store this value.
2. Compute the labels of the input nodes in G one by one, but for each label do not store the last bit. This fits in the remaining memory and leaves $3 \cdot w$ bits unused, because:
$$\begin{aligned} n \cdot (w-1) + w + 3 \cdot w &= (s - 4 + k) \cdot (w - 1) + w \\ &= s \cdot w + k \cdot w - (s + k - 4) < s \cdot w \end{aligned}$$
3. Using the partial labels computed in the previous step, and the $3 \cdot w$ bits of space that remain unused, we will try all possible combinations of the missing bits (exactly 2^n) and compute the label of b_{n+1} according to that combination.
4. Finally, from the stored value of b_{n+1} in step 1, compute the label of b_{n+2}.

For the previous algorithm, the maximum number of pebbles in the *ex post facto pebbling* is at least n, with probability 1, which happens after step 2 is finished. The reason is that all these pebbles will be used later to put other pebbles during step 3 (b_i for $1 \leq i \leq n$), but none of these labels is recomputed, so they cannot be removed by the "removing pebble" rule. We finish the proof by noticing that $n = s - 4 + k > s$, so the pebbling uses more pebbles that words of memory, as claimed. □

The counter example above uses an exponential number of calls to the random oracle. We believe that if the algorithm is restricted to a polynomial number of oracle calls, similar examples may not exist. Still, the specific bound in Lemma 15 may still be inaccurate.

An extension of Lemma 1 in [57] was stated in [60], although the actual proof of this extension appears in Theorem 4.2 in [59]. In particular, this extension also adds a logarithmic loss, which makes sense given that this result is more general than the original one. Still, to the best of our knowledge the following lemma, which is a specialization of Theorem 4.2 in [59], is the best result that has been obtained in line with Lemma 1 in [57]. The interested reader may derive its proof from its generalization mentioned before.

Lemma 16. Consider an algorithm A that makes at most q oracle queries and has $s \cdot w$ bits of memory, and a DAG G. Then the *ex post facto* pebbling of G corresponding to an execution of A uses at most $\frac{s \cdot w + w}{w - \log(q)}$ pebbles with probability $1 - 2^{-w}$ (over the choice of the random oracle).

Similar results in the literature, where a pebbling bound is translated into a computational bound using the *ex post facto* argument, are Proposition 2.31 in [81] and Lemma 1 in [13]. The bounds in these results are tight, and their proofs use similar arguments to Dwork's in [57]. Therefore, although we have not proved them incorrect, we believe they could be, and leave the verification of these results as future work.

7.3 Conclusions

In this chapter we showed a counterexample to a known result on the relation between pebbling games and computational bounds. We offered two possible paths for fixing the issue, although none of them seems conclusive. As far as we are aware, a tight reduction between pebbling complexity and computation complexity is still an open problem. The issue found with the original result does not affect greatly most of its applications, given that a corrected version exists, Lemma 16, and it incurs only in a logarithmic loss. Nevertheless, for applications where tightness is necessary (for example, memory-erasure), the counter-example found does affect its feasibility.

In the next chapter, we propose a new memory-erasure protocol based on graphs. To prove the protocol secure, we show computational bounds of adversaries aiming to compute labelling functions on graphs similar to the ones discussed in this chapter. For the reasons explained above, in particular the lack of tight bounds such as the one in Lemma 15, we used proofs techniques more inline with Lemma 16, even though they incur in a logarithmic loss.

8 Graph based memory-erasure

In this chapter we propose a software-based memory-erasure protocol based on graphs. Similar to the protocol introduced in Chapter 6, it aims to be secure against distant attackers. Its main objective is to reduce the communication complexity by avoiding the need to send a random value of the size of the prover's memory at the beginning of the protocol.

8.1 Introduction

Memory-erasure protocols such as the one by Perito [111] and the one in Chapter 6 require that the verifier sends a sequence of bits as large as the prover's memory. This may be undesirable in some contexts, e.g. when the communication channels have limited capacity. Therefore, several works [60, 81] have tried solving this issue with varying degree of success. In these protocols the verifier sends a small seed to the prover, which uses it to compute the labelling of a suitably chosen graph and respond with the label of an output node. If executing this computation requires the prover to erasure its memory, then the verifier can confirm this by checking if the label received is correct. Unfortunately, these works are either inefficient [60] (require quadratic time) or are not able to erase but a relatively small fraction of the prover's memory [81]. Furthermore, both of them depend on the assumption that the prover is isolated during the execution of the protocol.

In this chapter we aim to solve these issues. To this end, we propose a graph based memory-erasure protocol with the same structure of the protocol in Chapter 6. During the first phase the verifier sends a seed which is used to compute the labelling of a graph by the prover. In this case the graph does not contain a single output node, but as many output nodes as are required so that its labels fill the prover's memory. In the second phase (interactive phase), we run several time-bound rounds to verify that those labels are stored. To prove our protocol secure, we identify a depth-robustness property for the graph that ensures its security: we show that a cheating prover needs to spend significant time to recompute any missing labels, so it will be caught by the verifier during the interactive phase.

We therefore propose a new class of graphs that allows our protocol to proceed in two phases, first compute and store the labels, and then reply to several rounds of challenges, thus proving that labels are stored. Then we prove that the resulting protocol can securely erase all but a small proportion of the prover's memory. Furthermore, our protocol provides its security guarantees in the presence of a distant-attacker, without resorting to device isolation or protected memory assumptions. The graphs we propose (Section 8.3) satisfy a classic property of depth-robustness and can be labelled **in-place**, i.e. by using only a constant memory overhead for the prover. Finally, in Section 8.4 we propose a lightweight variant of our graph, which achieves more performance with slightly less security requirements.

8.2 PoSE based on depth-robust graphs

This section proposes a protocol where the prover computes the labelling of a graph in its memory, starting from a small random seed transmitted by the verifier which is used to instantiate a hash function. We use the random oracle methodology to model and reason about the access of the adversary to this hash function [22].

Graph labelling (Definition 25) is a well established technique used in protocols for proofs of space [13, 58, 114], memory hardness [3, 4, 5, 6, 7] and classic secure erasure [60, 81]. The typical structure of these protocols is as follows:

- the verifier sends a nonce to the prover, which is used as a seed to determine the hash function h;
- the prover uses h to compute the labelling of an agreed-upon graph G, and stores the resulting labels of a subset of nodes, denoted $O(G)$;
- the verifier challenges the prover to reply with the labels of several randomly chosen vertices in the graph, and accepts the proof if these labels are correct.

Let m be the size of $O(G)$. The general goal of this technique is to ensure that any cheating prover that uses less than m words of memory to compute the responses can only do so correctly by paying a noticeable amount of computational time. This has been achieved by using depth-robust graphs [63, 114], which are graphs that contain *at least one long path*, even if a significant proportion of nodes have been removed. Intuitively, in the corresponding protocol, this means that the label for at least one of the verifier challenges will be hard to compute if the prover has cheated.

Let H be the set of all functions from $\{0,1\}^\kappa$ to $\{0,1\}^w$, for a suitably large κ. Elements from the set H are also called random oracles, since we will make the random oracle assumption for h drawn from this set. We use $\mathcal{A}^{\mathcal{O}}$ to denote that the adversary has access to oracle $\mathcal{O}$.

Given a DAG G_m, a hash function $h \in H$ (which we model as a random oracle) and a list of nodes $O(G_m) \subseteq V(G_m)$ of size m, we define our remote memory-erasure protocol as follows:

- $\mathsf{Setup}(m, w)$: return $\rho = (G_m, O(G_m))$, $\mathcal{I} = H$
- $h \leftarrow\!\$\; H$
- $\mathsf{Precmp}(\rho, h)$: compute the labels of the nodes in $O(G_m)$, and output the concatenation of these labels $\sigma = l(o_1)\|\ldots\|l(o_m)$ where $(o_1, \ldots, o_m) = O(G_m)$
- $\mathsf{Chal}(\rho)$: return a random vertex in $O(G_m)$
- $\mathsf{Resp}(\rho, \sigma, x)$: responds with $l(x)$, which was stored in σ
- $\mathsf{Vrfy}(\rho, h, x, y)$: return true if and only if $y = l(x)$

The protocol defined above cannot be secure if the attacker is allowed to do an unbounded number of operations during each round of the interactive phase, as in the simple protocol in Chapter 6. Therefore, taking into account that during each round of the fast phase the attacker can only do a small amount of computation, we will restrict the number of operations done by $\mathcal{A}_1$. In the new security definition, given that $\mathcal{A}_1$ has utilizes a hash function, we will restrict its use through an oracle and count the number of calls $\mathcal{A}_1$ makes to it. This restriction is formalized by the parameter q in Definition 27 that bounds resources for an adversary.

Definition 27. An adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ against the memory challenge game is (M, q)-*bounded* if and only if in any execution of $\mathsf{Exp}^{m,r,w}_{\mathcal{A}_0,\mathcal{A}_1}$ and any round i we have that $|\sigma_i| \leq M$ and $\mathcal{A}_1$ makes at most q queries to $\mathcal{O}$.

We also slightly modify the security game to include the attacker's access to the random oracle, as shown in Figure 8.1. Notice that the results from Section 6.3, which reduce the analysis of PoSE-DB protocols with several rounds in the interactive phase to a single round, can be easily extended to this new setting (only Lemma 7 needs adjusting).

```
$(\rho, \mathcal{I}) \leftarrow \mathsf{Setup}(m, w);\ \ h \leftarrow\!\!\$\ H$
$\mathbf{for}\ i := 1\ \text{to}\ r\ \mathbf{do}:$
    $\sigma_i \leftarrow \mathcal{A}_0(1^w, \rho, h, \{x_j | j < i\})$
    $x_i \leftarrow \mathsf{Chal}(\rho);\ \ y_i \leftarrow \mathcal{A}_1^{\mathcal{O}}(1^w, \rho, \sigma_i, x_i)$
$\mathbf{return}\ \forall i\colon \mathsf{Vrfy}(\rho, h, x_i, y_i) = \mathsf{true}$
```

Figure 8.1: Security experiment $\mathsf{Exp}^{m,r,w}_{\mathcal{A}_0,\mathcal{A}_1}$.

In order to prove our graph-based PoSE-DB secure, there are several issues we need to address. First, as we need fast responses, our verifier can query only one random challenge node per round. This means that it is not sufficient to have a single long path in the graph in order to catch a cheating prover; we will thus strengthen the depth-robustness property to require *at least a certain number of long paths* to be present. Another constraint, particularly relevant to memory-erasure, is that we should be able to compute the graph labelling with minimal memory overhead, so that as much memory as possible can be erased from the device. We call this property **in-place**, since intuitively it means that the set of labels to be stored should be computed in almost the same amount of space as their total size. To our knowledge, no graph in the literature exists that satisfies these two properties. We design one in Section 8.3.

Definition 28. A graph G can be labelled **in-place** with respect to a list of nodes $O(G) \subseteq V(G)$ if and only if there is an algorithm that outputs the list of labels for all nodes in $O(G)$ using at most $|O(G)| \cdot w + \mathcal{O}(w)$ bits of memory.

A second issue that we address lies in the tightness of the security bound, i.e. how big is the gap between the memory erased by a cheating prover and the memory it is supposed to erase. In our case study, this gap could be used to store malware, so it should be as small as possible. We improve upon previous security bounds for protocols based on graph labelling, first by performing a fine-grained and formal security analysis, and second by identifying a restricted class of adversaries, that simplifies the proofs while also further improving the security bound. We relate this class to previous restrictions in this area and argue that it is a strictly more general notion, resulting in weaker restrictions for the adversary and therefore stronger security guarantees.

8.2.1 Depth-robustness is sufficient for security

We show that a variation of the classic graph depth-robustness property [63, 114] is sufficient to prove security for the PoSE scheme from Section 8.2. As explained above, while depth-robustness in all previous works requires the existence of one single long path after some nodes have been removed, we require the existence of several long paths. The computation required by a cheating prover in this context is proportional to the depth of the longest remaining paths. While in most previous works the length is linear in the size of the graph, we will tolerate graphs with slightly shorter paths, i.e. of sublinear size. On the one hand, this is useful in practice, since it will allow us to construct a class of depth-robust graphs that can be labelled **in-place** efficiently. On the other hand, we show that this does not affect security, as long as the computational power of the adversary during the interactive phase is constrained in proportion to the prescribed path length, which can be done by the time measurements we described in Section 6.2.

If G is a graph and v of G, we let $\mathsf{llp}(v, G)$ be the length of the longest path in G that ends in v. If $R \subseteq V(G)$ is a subset of nodes, we denote by $G \setminus R$ the graph obtained after removing the nodes in R and keeping all edges between the remaining nodes.

Definition 29. Let G be a DAG and let $O(G) \subseteq V(G)$ with $|O(G)| = \mu$. We say G is (μ, γ)-**depth-robust** with respect to $O(G)$ if and only if for every set $R \subset V(G)$ s.t. $|R| < \mu$ there exists a subset of nodes $O' \subseteq O(G)$ with $|O'| \geq \mu - |R|$ such that for every node $v \in O'$ there is a path in $G \setminus R$ of length at least γ that ends in v, i.e. $\mathsf{llp}(v, G \setminus R) \geq \gamma$.

The security bounds obtained using a depth-robust graph in the graph-based PoSE scheme are presented in Corollary 3, which can be deduced from a direct application of Theorem 3 and Proposition 2. For this protocol, the oracle $\mathcal{O}$ in Figure 8.1 gives to $\mathcal{A}_1$ oracle access to the hash function h resulting from the initialization phase. The first part of Corollary 3 shows that we obtain better security bounds if we consider a notion of graph-restricted adversaries, which will be discussed in the next subsection. Note that the parameter m for the number of memory blocks in the experiment and the bound $q < \gamma$ on the number of oracle calls by $\mathcal{A}_1$ are related to the pair (m, γ) determined by the depth-robustness of the graph.

Corollary 3. Assume the graph-based PoSE scheme is instantiated with parameters (m, r, w) and an (m, γ)-**depth-robust** graph G. Then, for any (M, q)-bounded adversary $(\mathcal{A}_0, \mathcal{A}_1)$, with $q < \gamma$, we have:

$$\Pr_h[\mathsf{Exp}^{m,r,w}_{\mathcal{A}_0,\mathcal{A}_1} = \mathsf{true}] \leq (M'/m)^r + 2^{-w_0} \text{ where:}$$

- if $\mathcal{A}$ is *graph-restricted*: $w_0 = w$ and $M' = \lceil M/w \rceil$
- else: $w_0 = w - \log(m) - \log(q)$ and $M' = \lceil M/w_0 \rceil$

Notice that this result is not tight for the case of general adversaries. Although not explicit here, the proof of this result follows a similar approach as the *ex post facto* argument from Chapter 7. As discussed there, we believe obtaining a tight bound for this type of result relating combinatorial bounds on pebbling games and computational bounds in the ROM is currently an open problem.

Intuitively, the result from Corollary 3, with respect to graph-restricted adversaries, shows that the attacker's success probability is proportional to M: the smaller the state it uses to reply to our challenges, the higher the probability that it will fail to pass the verification test. This result is not far from optimal, as this bound can actually be achieved by an adversary that stores $\frac{M}{w}$ of the labels in $O(G)$. For example, if we would like to erase a malware of size at least 5 KB from a device with total memory 100 KB, then the parameters would be $w = 256, m = 100 \cdot 2^{13}/256 = 3200, M = (100 - 5) \cdot 2^{13} = 778240$. If we wanted to be certain that any graph-restricted attacker can pass the protocol without erasing malware with probability at most 10^{-3}, then we need to execute the protocol with 112 rounds.

8.2.2 Graph-restricted adversary

Several works studying protocols based on graph labelling related to ours consider restricted classes of adversaries in order to obtain tight security bounds and reductions to graph pebbling [6, 57, 58]. In all these notions, the adversary can only make oracle calls corresponding to valid labels in the graph and, in addition, the adversary is restricted in the type of computation that it can apply to get a state to be stored in memory, e.g. it can only store labels [57, 58] or so-called entangles labels [6]. The notion that we consider in Definition 30 is more general, considering the full class of graph-restricted adversaries that can perform any computation to obtain the state to be stored.

Definition 30. We say that $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ is *graph-restricted to G* if and only if all oracle calls done by $\mathcal{A}_1$ correspond to valid labels (equal to $\ell^{-}(v)$ for some node v), and its responses to the verifier challenges are always correct.

A graph-restricted adversary may compute arbitrary functions (for example compression, cut labels into pieces, entangle them with new algorithms, etc), but it doesn't do any guessing while making oracle queries nor when responding challenges. The assumption is that the adversary knows what it is doing, i.e. knows that an oracle query would be useless or that a particular response to the challenge would be wrong. The notable difference with respect to the previous classes of graph-playing adversaries, e.g. the pebbling adversary from [58] or the entangled pebbling adversary from [6], is that we have no a priori restriction on how the labels of values are processed and stored. Next, we discuss these notions in more detail.

Pebbling adversary: The use of graph-labelling in security protocols can be traced back to the work of Dwork et al. [57], who proposed its use for proofs of work. Proofs of work don't require precomputation and storage of a prescribed state; the adversary in [57] simply has to compute and return the challenged label. Since guessing a previously unseen label can happen only with negligible probability, the security analysis for such an adversary can be reduced to the classic notion of graph-pebbling complexity [110], where the adversary is restricted to playing a game on the graph (applying the hash function corresponds to placing a pebble).

Adversary with pre-stored pebbles: The pebbling technique cannot be applied directly to proofs of space [58], since the adversary is supposed to perform some pre-computation. Then it could use the available space to encode information about the labels of the graph. When challenged, it will attempt to make the minimal number of oracle calls that, combined with information stored in the state, allows it to obtain the needed responses. For their security proof, Dziembowski et al. [58] make the simplifying assumption that the best the adversary can do is to choose a set of labels on the graph, not necessarily corresponding to the challenge nodes, and store them in the memory, i.e. the adversary cannot compress or combine labels. This fixed set of labels is then used to reply to any of the given challenges. Relying on this assumption, the security analysis can then be reduced to a pebbling game on the graph where the power of $\mathcal{A}$ is slightly increased, i.e. it is allowed to pre-store pebbles.

Adversary with algebraically entangled pebbles. Alwen et al. [6] have further relaxed this restriction, allowing the adversary to store several labels in one block of memory. However, these pebbles are not directly accessible to $\mathcal{A}$, but are entangled in an algebraic relation that allows to derive a subset of pebbles once another subset is available. They show that this is possible in practice: the simplest example is storing the exclusive or of several labels, but more complex encodings are possible, e.g. based on polynomials. Interestingly, under some conjectures, they have shown that the general computational adversary can be reduced with minimal security loss to a graph-restricted adversary with entangled pebbles. However, these conjectures have been later disproved in [93].

8.2.3 Security proof: first step

The following theorem will be the main ingredient for proving Corollary 3. We split its proof in two steps. The first step is presented in this subsection and is independent of the adversarial class (graph-restricted or general). The second step is simpler for graph-restricted adversaries. We present the proof for that case and for the case of general adversaries in the next subsection. Our proof strategy builds upon the proofs from [6] and [114]. We combine ideas from both proofs, improve on their security bounds, adapt them to graph-restricted adversaries and to the case where we only ask one challenge from the prover, relying on our new notion of depth-robustness.

Theorem 3. Assume the graph-based PoSE scheme is instantiated with parameters $(m, 1, w)$ and with a (m, γ)-**depth-robust** graph G. Let $\mathcal{A}$ be any (M, q)-*bounded* adversary, with $q < \gamma$. There exists a set of random oracles $H_{\mathsf{good}} \subseteq H$ such that $\mathcal{A}$'s winning probability is bounded by $\frac{M'}{m}$ within H_{good}, where:

- $M' = \left\lceil \frac{M}{w} \right\rceil$ and $|H_{\mathsf{good}}| \geq |H| \cdot (1 - 2^{-w})$ if $\mathcal{A}$ is graph-restricted
- else: $M' = \left\lceil \frac{M}{w_0} \right\rceil$ and $|H_{\mathsf{good}}| \geq |H| \cdot (1 - 2^{-w_0})$, where $w_0 = w - \log(m) - \log(q)$.

Using the reductions from Section 6.2, we can extend this result to any number of rounds, where we decrease the winning probability of a malicious prover by increasing the number of rounds. From Proposition 2 and Theorem 3, we then obtain the main result Corollary 3.

Consider an adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ against the memory challenge game for our PoSE instantiated with a graph G. Let $\sigma = \mathcal{A}_0(1^w, \rho, h)$. Let $O(G) = \{o_1, \ldots, o_m\}$ be the set of all challenge vertices that can be given to $\mathcal{A}_1$ during the experiment. For this protocol the adversary $\mathcal{A}_1$ has access to the hash function h through the random oracle $\mathcal{O}$, which we make clear in the notation onwards by calling the oracle function $\mathcal{O}_h$. A query Q from $\mathcal{A}_1$ to $\mathcal{O}_h$ is good if $\exists v \colon Q = \ell^-(v)$. For every $i \in \{1, \ldots, m\}$, considering the execution of $\mathcal{A}_1^{\mathcal{O}_h}(1^w, \rho, \sigma, o_i)$, we let:

- $Q_{i,j}$ be the input to the j-th oracle call made to $\mathcal{O}_h$ by $\mathcal{A}_1$ in this execution;
- t_i be the total number of oracle calls made by $\mathcal{A}_1$ in this execution;

- Q_{i,t_i+1} be the output of $\mathcal{A}_1$ on this execution.

Definition 31 (Blue node)**.** We say that a node $v \in V(G)$ is blue if and only if there exists $v' \in V(G), i \in \{1,\ldots,m\}$ and $j \geq 1$ such that:

1. $v \in \Gamma^-(v')$ and $Q_{i,j} = \ell^-(v')$
2. $\forall i' \in \{1,\ldots,m\} \forall j' < j. \, Q_{i',j'} \neq \ell^-(v)$

We say that v is a *blue node* from iteration j, and that $Q_{i,j}$ is the query associated to v. Denote by B_j all blue nodes in iteration j, and by B the set of all blue nodes. A bitstring is a *pre-label* if it is equal to $\ell^-(v)$ for some v, and a *possible pre-label* if it is the concatenation of a node v and $w \cdot |\Gamma^-(v)|$ bits.

The first point above implies $\ell^-(v') = v' \| y_1 \| \ldots \| y_m$ and $\ell(v) \in \{y_1,\ldots,y_m\}$. The second point will ensure that we can extract the labels of blue nodes for free, i.e. without querying them to the random oracle, by running $\mathcal{A}_1$ in parallel for all possible challenges on the state σ computed by $\mathcal{A}_0$. Let $T_i = t_i$ if the response to the challenge o_i by $\mathcal{A}_1$ is correct, or infinite otherwise. The strategy for the proof of Theorem 3 will be the following:

- *First step:* prove that the success probability of the adversary is smaller than the fraction between the number of blue nodes, whose label it has stored, and the total number of labels it is supposed to store.
- *Second step:* prove an upper bound on the number of blue nodes; for graph-restricted adversaries, this will be $\left\lceil \frac{M}{w} \right\rceil$, matching the value M' from Theorem 3.

First, we show that to answer a challenge correctly, $\mathcal{A}_1$ needs to recompute the labels of the longest path to any node that is not blue:

Lemma 17. $\forall i \in \{1,\ldots,m\}\colon T_i \geq \mathsf{llp}(o_i, G \setminus B)$, i.e. the time it takes to compute the label of o_i is at least the length of the longest path $\mathsf{llp}(o_i, G \setminus B)$.

Proof. Fix i. If the response to the challenge o_i is not correct, then the claim is trivial (T_i is infinite). The case $o_i \in B$ is also trivial. Assume $o_i \notin B$ and that the answer to the challenge o_i is correct. Consider the longest path ending in o_i, let it be $(v_1, v_2, \ldots, v_{k-1}, v_k = o_i)$, where $k = \mathsf{llp}(o_i, G \setminus B)$. As none of the nodes in this path are blue, all their labels were computed by asking the oracle for the corresponding value. Let f_j be the smallest index such that $Q_{x,f_j} = \ell^-(v_j)$ for some x. As $v_k = o_i$, then $t_i \geq f_k$.

We prove that $\forall j\colon 1 \leq j < k$ we have $f_j < f_{j+1}$. Each inequality can be proved using the same argument, so we prove only $f_1 < f_2$. As v_1 is not blue and v_2 is a successor of v_1, then the query $\ell^-(v_2)$ in round f_2 (which contains the label of v_1) must have happened after the query $\ell^-(v_1)$ in round f_1. This implies $f_2 > f_1$, as needed.

As $\forall i\colon f_i \geq 1$ then $f_k \geq k \implies T_i \geq k$ as was needed.

□

If the graph is (m,γ)-**depth-robust**, such paths are with high probability longer than γ if M is significantly smaller than $m \cdot w$.

Lemma 18. If the graph G is (m, γ)-**depth-robust**, then:

$$\Pr_{i \leftarrow\$ [m]}[T_i \geq \gamma] \geq 1 - |B| m^{-1}$$

Proof. We can assume $|B| < m$, since the result trivially holds otherwise. Therefore, from the definition of **depth-robustness**, there exist a set $O' \subseteq \{o_i\} \setminus B$ with $|O'| \geq m - |B|$ s.t. $\forall o_i \in O' \colon \mathsf{llp}(o_i, G \setminus B) \geq \gamma$. From Lemma 17, we deduce $\forall o_i \in O' \colon T_i \geq \gamma$. Therefore, we deduce

$$\Pr_{i \leftarrow\$ [m]}[T_i \geq \gamma] \geq \Pr_{i \leftarrow\$ [m]}[o_i \in O'] \geq 1 - |B| m^{-1}$$

as required. □

Given the previous results, we can bound the probability of success of the adversary in proportion to the number of blue nodes, as shown in the following lemma. This way we conclude the first step of the proof of Theorem 3.

Lemma 19. Given an (M, q)-*bounded* adversary $\mathcal{A}$, with $q < \gamma$, and a random oracle $h \in H$ such that $|B| \leq \bar{B}$, then its probability of success is bounded by $\bar{B} \cdot m^{-1}$.

Proof. From Lemma 18 we deduce that the probability that $T_i \geq \gamma$ is at least $1 - |B| m^{-1} \geq 1 - \bar{B} m^{-1}$. But given that $q < \gamma$, it follows that the adversary cannot reply to these challenges on time, which implies that its probability of success is bounded by $\bar{B} m^{-1}$, as claimed. □

8.2.4 Security proof: second step for graph-restricted adversaries

Next we show that, with high probability over the choice of the random oracle, we obtain the desired upper bound on the number of blue nodes. That is, for a big proportion of (good) random oracles, $|B|$ is smaller than the number of blocks stored by $\mathcal{A}_0$ in σ. This will help us bound the prediction ability of a cheating adversary that did not store enough memory.

Definition 32 (Good oracle)**.** We say that $h \in H$ is a good oracle for a *graph-restricted* $\mathcal{A}$ if $|B| \leq \left\lceil \frac{M}{w} \right\rceil$.

Intuitively, relying on the adversary $\mathcal{A}$ and its set of blue nodes, we will construct an encoder and a decoder to which we will be able to apply Lemma 20 to derive the desired bounds. Recall that H is the set of all random oracles from $\{0,1\}^\kappa$ to $\{0,1\}^w$. We will rely on the following well-known result.

Lemma 20 (adapted from Fact 8.1 in [48])**.** If there are deterministic encoding and decoding procedures

$$Enc \colon H \to \{0,1\}^s$$

and $Dec \colon \{0,1\}^s \to H$ such that $\Pr_{x \leftarrow\$ H}[Dec(Enc(x)) = x] \geq \delta$, then $s \geq \log|H| + \log \delta$.

Let $\mathcal{S}$ be the oracle machine in Figure 8.2. It executes $\mathcal{A}_1$ in parallel for all possible challenges. Notice that $\mathcal{S}$ executes basically the same operations as $\mathcal{A}_1$ and does not make repeated queries to the oracle. $\mathcal{S}$ terminates when it has finished processing all parallel executions of $\mathcal{A}_1$. In each round, $\mathcal{S}$ processes the respective oracle calls of each instance of $\mathcal{A}$, and outputs all calls to the environment at the end of the round. In the next lemma $\mathcal{S}$ will be used by the encoder and decoder that we construct, which will observe its outputs and process all the oracle calls of $\mathcal{S}$. The encoder will answer the oracle calls by looking directly at h, while the decoder will obtain the desired values from the encoded state. The fact that $\mathcal{A}_1$ is graph-restricted helps the decoder determine blue nodes directly.

```
run $\mathcal{A}_1^{\mathcal{O}_h}(1^w, \rho, \sigma, o_1), \dots, \mathcal{A}_1^{\mathcal{O}_h}(1^w, \rho, \sigma, o_m)$ in parallel :
for $j = 1$ to $\max\{t_1, \dots, t_n\} + 1$ :  Let $L$ be an empty list
  for $i = 1$ to $m$ :
    if $\mathcal{A}_1^{\mathcal{O}_h}(1^w, \rho, \sigma, o_i)$ has not finished :
      run $\mathcal{A}_1^{\mathcal{O}_h}(1^w, \rho, \sigma, o_i)$until the $j$-th query (or output) $Q_{ij}$
      - if $Q_{ij}$ is an output, add $(0, i, Q_{ij})$ to $L$
      - if $Q_{ij}$ is repeated, answer with the previous response
      - if $Q_{ij}$ is a possible pre-label of $v$, add $(1, v, Q_{ij})$ to $L$
      - else ask $\mathcal{O}_h(Q_{ij})$, relay the answer to $\mathcal{A}_1$ and store it
  for each $(t, v, q)$ in $L$ :  (ordered in inverse topological order with respect to $G$)
    output $(t, v, q)$
    if $t = 1$ ask $\mathcal{O}_h(q)$, relay the answer to $\mathcal{A}_1$ and store it
```

Figure 8.2: Procedure for $\mathcal{S}_{\mathcal{A}}^{\mathcal{O}_h}(1^w, \rho, \sigma)$

Lemma 21. Let $\mathcal{A}$ be an attacker with parameters $(m, 1, w)$ and $H_{\mathsf{good}} \subseteq H$ be the corresponding set of good oracles. Then $|H_{\mathsf{good}}| \geq (1 - 2^{-w}) \cdot |H|$.

Proof. We show there exist an encoder and a decoder algorithm that using $\mathcal{A}$ are able to compress a random function from H, as long as the size of B is greater than $\left\lceil \frac{M}{w} \right\rceil$, i.e. the function is not in H_{good}. Then we apply Lemma 20 to obtain an upper bound for the number of functions for which this is possible.

The encoder works as follows: for a function h, first run $\mathcal{A}_0$ to obtain σ. Second, run $\mathcal{S}$ with input σ and keep track of its outputs. As the adversary is graph-restricted, all these outputs are either tuples that correspond to labels of challenge nodes, or pre-labels. In both cases, store the blue nodes in order. Once S finishes, if the number of blue nodes stored is less than or equal $\left\lceil \frac{M}{w} \right\rceil$, output nothing. Else, output σ, the responses c to all oracles calls made by $\mathcal{S}$ in order (except the ones associated with blue nodes), and all the remaining oracle values c' (not asked by S, nor labels of blue nodes) in lexicographic order. Note that the labels of blue nodes are not stored explicitly but encoded in σ.

The decoder works as follows: given σ, c, c', it will output the whole function table for h. First, it executes $\mathcal{S}$ with input σ. When $\mathcal{S}$ makes an output, as the adversary is graph-restricted, the decoder can deduce the labels of the blue nodes associated to it (if any), and store these values. When $\mathcal{S}$ makes an oracle call, if the response to the oracle call is known (because it is the label of a node that was stored before, or is a repeated call), then respond with that value. Else respond with the next value from c. When S finishes, read all the remaining values in the input c'. At this point the decoder knows the whole table for h, and outputs it.

The size of the encoding is exactly $M + \log|H| - |B| \cdot w$, and it is correct with probability δ, which is the proportion of functions h such as the number of blue nodes is more than $\left\lceil \frac{M}{w} \right\rceil$ (oracles not in H_{good}). From Lemma 20 we deduce $M + \log|H| - |B| \cdot w \geq \log|H| + \log(\delta)$ which, together with $|B| > \left\lceil \frac{M}{w} \right\rceil$, implies $\delta \leq 2^{-w}$. We conclude that $|H_{\mathsf{good}}| = (1-\delta) \cdot |H| \geq (1 - 2^{-w}) \cdot |H|$. □

Putting together Lemma 21, Definition 32 for good oracles and Lemma 18, we obtain immediately the statement of Theorem 3 for graph-restricted adversaries.

8.2.5 Security proof: second step for general adversaries

The proof for general adversaries is similar to the one in the previous subsection. The main difference is that the decoder will need some additional advice to detect where the blue nodes are, which will imply worse bounds when applying Lemma 20. The next definition and lemma hold for general adversaries.

Definition 33 (Good oracle)**.** We say that $h \in H$ is a good oracle for an adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ if we have $|B| \leq \left\lceil \frac{M}{w - \log(m \cdot q)} \right\rceil$.

Lemma 22. If $\mathcal{A}$ has parameters $(m, 1, w)$ and $H_{\mathsf{good}} \subseteq H$ be the corresponding set of good oracles. Then $|H_{\mathsf{good}}| \geq |H| \cdot (1 - 2^{-w+\log(m \cdot q)})$.

Proof. The proof follows closely the one of Lemma 21, but must be extended for general adversaries.

As before, the encoder will output 0 if $h \in H_{\mathsf{good}}$. Else, it will output σ, p, c, c', where the new value p is a list of indices, where each index a_i corresponds to the i-th output of $\mathcal{S}$, and indicates that it contains the first appearance of the label of a blue node. Each of these pairs can be encoded using $\log(m \cdot q)$ bits.

The decoder will also execute $\mathcal{S}$ as before. The difference is that when $\mathcal{S}$ outputs a value, only if the corresponding index is in p, it will extract and store the labels of the blue nodes associated with that output. At the end of execution, the decoder will have stored all labels of blue nodes.

As the size of p is at most the number of blue nodes, the size of the encoding is at most $M + |B| \cdot \log(m \cdot q) + \log|H| - |B| \cdot w$, and it is correct with probability $\delta = 1 - |H_{\mathsf{good}}|/|H|$. From Lemma 20 we deduce:

$$
\begin{aligned}
M + \log|H| - |B| \cdot (w - \log(m \cdot q)) &\geq \log|H| + \log(\delta) \\
\implies \delta \leq 2^{-w+\log(m \cdot q)} \quad &\text{from } |B| > \left\lceil \frac{M}{w - \log(m \cdot q)} \right\rceil
\end{aligned}
$$

We conclude that $|H_{\mathsf{good}}| = (1 - \delta) \cdot |H| \geq (1 - 2^{-w+\log(m \cdot q)}) \cdot |H|$, as claimed. □

8.3 Depth-robust graphs that can be labelled in-place

In this section we look for a family of **depth-robust** graphs with respect to a subset of nodes that can be labelled **in-place** with respect to the same subset of nodes. Although some classes of graphs from previous works, e.g. [13, 81], satisfy pebbling-hardness notions related to depth-robustness and can be labelled in place, they don't satisfy the strong depth-robustness property that we require. We therefore propose a new construction. Our construction is based on the graph in [120], where a simple and recursively constructible depth-robust graph with respect to edge deletions was proposed. The recent results in [29] showed that it is possible in general to transform an edge depth-robust graph into a vertex depth-robust graph where only one single long path is guaranteed to exist after node deletions, but not several as required by our notion. Inspired by these results, although using a different (and ad-hoc) transformation on the graph from [120], we designed a new class of graphs with the properties by the protocol in Section 8.2. The main result of this section is the following:

Theorem 4. There exists a family of DAG $\{G_n\}$ where G_{n+1} has $\mathcal{O}(2^n \cdot n^2)$ nodes and a subset of nodes $O(G_{n+1}) \subset V(G_{n+1})$ of size 2^n such that:

1. G_{n+1} is $(2^n, 2^n)$-**depth-robust** with respect to $O(G_{n+1})$;
2. G_{n+1} can be labelled **in-place** with respect to $O(G_{n+1})$.

If Γ is a graph, we say that a graph G is a fresh copy of Γ if it is isomorphic and disjoint, and denote it by $G \cong \Gamma$. Let L_n be a list of 2^n nodes. For any DAG G, let $\mathsf{In}(G)$, $\mathsf{Out}(G)$ be its input and output nodes respectively. If Y and Z are two disjoint lists of nodes of the same size, denote by $X = Y : Z$ the concatenation of both lists. If $X = (x_1, \ldots, x_n)$ and $Y = (y_1, \ldots, y_n)$ are lists of nodes of size n, then $X \to Y = \{(x_i, y_i) \mid 1 \leq i \leq n\}$.

In our construction we use graph with high connectivity, so-called connectors.

Definition 34. [connector] A directed acyclic graph with bounded indegree, n inputs, and n outputs is an n-connector if for any $1 \leq k \leq n$ and any sequences $(s'_1, \ldots, s'_k)$ of inputs and $(t'_1, \ldots, t'_k)$ of outputs there are k vertex-disjoint paths connecting each s'_i to the corresponding t'_i.

Next, we introduce notation for connecting two lists of nodes using a connector graph. Let H_n be a 2^n-connector. Given two lists of nodes I and O of size 2^n, and a fresh copy of H_n named C, we denote by $I \overset{H}{\to} O$ the graph with nodes $I \cup O \cup V(C)$ and edges $E(C) \cup (I \to \mathsf{In}(C)) \cup (\mathsf{Out}(C) \to O)$.

Now we are ready to construct our graph family. For each graph G_n from our family, we also define a list of so-called base nodes, which will serve for connecting G_n with other graphs in order to construct the graph G_{n+1}. We define G_0 to be the graph formed of a single node. Then, G_{n+1} is constructed from two copies of G_n and a copy of H_n, which we connect with a set of additional edges. First, every base node in the first copy of G_n is connected through an edge to a corresponding input node in the copy of H_n. Second, the copy of H_n is connected to the second copy of G_n through a recursive edge construction described below and illustrated in Figure 8.3. Formally, we have:

- G_0: $V(G_0) = \{v\}$, $E(G_0) = \emptyset$ and $\mathsf{Base}(G_0) = (v)$.
- G_n contains three components (L, C, R), denoted by

$$\mathsf{Left}(G_n), \mathsf{Center}(G_n), \mathsf{Right}(G_n)$$

  where $L \cong R \cong G_{n-1}$, and $C \cong H_{n-1}$. We let $\mathsf{Base}(G_n) = \mathsf{Base}(L) \colon \mathsf{Base}(R)$ denote the base vertices.

- The operator $\triangleright$ denotes a recursively defined subgraph, $X \triangleright G_0 = X \to \mathsf{Base}(G_0)$, and:

$$(X_1 : X_2) \triangleright Y = (X_1 \triangleright \mathsf{Left}(Y)) \cup (X_2 \overset{H}{\to} \mathsf{In}(\mathsf{Center}(Y)))$$

  The vertices of G_n are:

$$V(G_n) = V(L) \cup V(C) \cup V(R) \cup V(\mathsf{Out}(C) \triangleright R)$$

  The edges of G_n include the edges in the components L, C, R, plus two new sets of edges: the first from $\mathsf{Base}(L)$ to $\mathsf{In}(C)$, and the second from $\mathsf{Out}(C)$ to R:

$$E(X) = E(L) \cup E(C) \cup E(R) \cup (\mathsf{Base}(L) \to \mathsf{In}(C)) \cup E(\mathsf{Out}(C) \triangleright R)$$

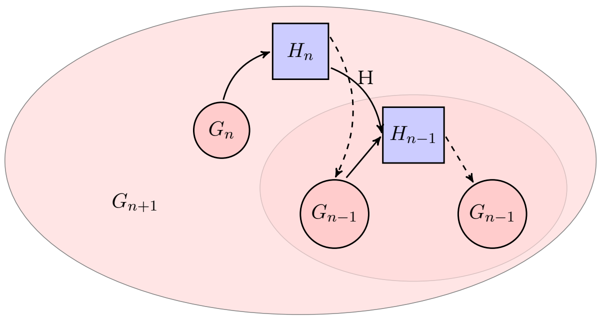


Figure 8.3: G_{n+1}, continuous arrows represent simple edges $\to$, H arrows represent connector subgraphs $\overset{H}{\to}$ and dashed arrows represent the operator $\triangleright$.

The next result is essential for our construction. It states that if some nodes are removed from a connector with the objective of disconnecting input/output pairs, then it is always possible to do so by just removing inputs or outputs. For a set of nodes R, let $\mathsf{Con}(R)$ be all pairs of nodes (v, w), where v is an input and w is an output, such that all paths from v to w contain at least one node in R.

Lemma 23. Let C_n be an n-connector and $R \subset V(C_n)$ be a subset of its nodes with $|R| < n$. Then there exists a subset of nodes R' containing only input and output nodes, such that $|R'| \leq |R|$ and $\mathsf{Con}(R) \subseteq \mathsf{Con}(R')$.

Proof. Consider a bipartite graph B_R with n nodes on each side such that two nodes are connected if and only if the corresponding pair is in $\mathsf{Con}(R)$. Consider a maximum matching M

and the minimum vertex cover VC in this graph. From König's theorem [86] we deduce that $|M| = |VC|$. Moreover, from the connector property of C_n we deduce that $|M| \leq |R|$: otherwise, using the connector property between the sets of nodes in the matching, we would find $|R| + 1$ disjoint paths from the input to the output, and those cannot be covered by the nodes in R. Then, $|VC| \leq |R|$. Consider the set $R' = VC$. If these nodes are removed from C_n, consider the bipartite graph $B_{R'}$ (analogous to B_R). Clearly B_R is a subgraph of $B_{R'}$ by the vertex cover property. This implies that $\mathsf{Con}(R) \subseteq \mathsf{Con}(R')$, as claimed. □

To prove the **depth-robustness** of G_n, we will prove first that after removing some nodes, there remains a long path containing nodes from $\mathsf{Base}(G_n)$.

Theorem 5. Consider G_n and any subset of nodes $R \subset V(G_n)$, with $|R| < 2^n$. There is a path L in $G_n \setminus R$ such that $|V(L) \cap \mathsf{Base}(G_n)| \geq 2^n - |R|$.

proof sketch. First, by using Lemma 23 we reduce the problem by showing that if there is a counterexample set R, then there is also a counterexample set R', but for which no node removed is strictly inside any connector graph, i.e. they must be part of the input or output. To conclude, we prove by induction a stronger statement, in which not only the path L exists, but the nodes in

$$(V(L) \cap \mathsf{Base}(\mathsf{Right}(\mathsf{Left}^k(G_n)))$$

also must be reachable from the nodes in $\mathsf{In}(\mathsf{Center}(\mathsf{Left}^k(G_n)))$ for $0 \leq k \leq n$. This property makes possible to glue together paths while proving the induction hypothesis. □

Proof. Assume the contrary, there exists a set R such that there is no path containing $2^n - |R|$ nodes from $\mathsf{Base}(G_n)$. Consider the partition $B \cup R_1 \cup \ldots \cup R_k$ of the vertices in R such that the nodes in each partition R_i belong to exactly one subgraph H_i (defined by some $\xrightarrow{H}$, or the Center of some component) and $B = R \cap \mathsf{Base}(G_n)$. If R'_i is the set of nodes guaranteed to exist by Lemma 23, then removing R'_i instead of R_i doesn't increase the connectivity between nodes in $\mathsf{Base}(G_n)$. We deduce that if the result is true for R', then after removing the nodes in $R' = B \cup R'_1 \cup \ldots \cup R'_k$ from G_n, there is a path containing $2^n - |R'| \geq 2^n - |R|$ nodes from $\mathsf{Base}(G_n)$. The vertices in R' are either in the input or output of some H_i, or in $\mathsf{Base}(G_n)$.

It remains to prove that if the nodes in R' are removed, there is a path with the required features. We prove this by induction in a stronger statement: in $G_n \setminus R'$ there exists a path L with the required properties such that the nodes in $V(L) \cap \mathsf{Base}(\mathsf{Right}(\mathsf{Left}^k(G_n)))$ are reachable from $\mathsf{In}(\mathsf{Center}(\mathsf{Left}^k(G_n)))$ for $0 \leq k < n$.

The base cases of the induction G_0 and G_1 can be checked manually. For the induction step assume in R' there are l, r, c nodes from $\mathsf{Left}(G_{n+1})$, $\mathsf{Right}(G_{n+1})$ and

$$\mathsf{Center}(G_{n+1}) \cup (\mathsf{Out}(\mathsf{Center}(G_{n+1})) \triangleright \mathsf{Right}(G_{n+1}) \setminus \mathsf{Right}(G_{n+1}))$$

respectively. By the induction hypothesis, there is a path L_l in $\mathsf{Left}(G_{n+1})$ with the required properties, i.e. $|V(L_l) \cap \mathsf{Base}(\mathsf{Left}(G_{n+1}))| \geq 2^n - l$. There is also L_r in $\mathsf{Right}(G_{n+1})$.

We will finish the proof by case analysis. If $r + c \geq 2^n$, then $2^n - l \geq 2^{n+1} - |R'|$ and $L = L_l$ has the required properties. Else, $r + c < 2^n$. Let $c_i = |R' \cap \mathsf{In}(\mathsf{Center}(G_{n+1}))|$.

Next we prove that there is a list of nodes $F \subseteq V(L_r) \cap \mathsf{Base}(\mathsf{Right}(G_{n+1}))$ such that $|F| \geq 2^n - r - c + c_i$ and all nodes in F are reachable from at least one node in $\mathsf{Out}(\mathsf{Center}(G_{n+1}))$.

Consider a node $v \in V(L_r) \cap \mathsf{Base}(\mathsf{Right}(G_{n+1}))$. Let k be the largest such that $v \in \mathsf{Left}^k(\mathsf{Right}(G_{n+1}))$, and $G_v = \mathsf{Left}^k(\mathsf{Right}(G_{n+1}))$. If $k = n$, then the only way that this node is not reachable from $\mathsf{Out}(\mathsf{Center}(G_{n+1}))$ is if its predecessor in that set belongs to R'. If $k < n$, then by the induction hypothesis, given that $v \in \mathsf{Base}(\mathsf{Right}(G_v))$, then v is reachable from a node in $\mathsf{In}(\mathsf{Center}(G_v))$, call this node w. The only way that v is not reachable from $\mathsf{Out}(\mathsf{Center}(G_{n+1}))$ through w is if all paths in the connector $\bar{H}$ (which is isomorphic to H_{n-k-1}) from this set to w are covered by nodes in R'. As w is not covered, this is only possible if all nodes from $\mathsf{Out}(\mathsf{Center}(G_{n+1}))$ connected to $\mathsf{In}(\bar{H})$ are in R', which are exactly 2^{n-k-1} nodes. But there are at most 2^{n-k-1} nodes in $\mathsf{Base}(\mathsf{Right}(G_v))$. Applying the previous deduction for all v, we deduce that the number of nodes in $V(L_r) \cap \mathsf{Base}(\mathsf{Right}(G_{n+1}))$ not reachable from $\mathsf{Out}(\mathsf{Center}(G_{n+1}))$ is at most the number of nodes that were removed from $\mathsf{Out}(\mathsf{Center}(G_{n+1}))$. As this number is at most $c - c_i$, the claim follows.

Consider now the path P_F determined by the list of nodes F. All these nodes are reachable from $\mathsf{Out}(\mathsf{Center}(G_{n+1}))$. Furthermore, as $c_i \leq c + r < 2^n$, these nodes are also reachable from some node in $\mathsf{In}(\mathsf{Center}(G_{n+1}))$. We will concatenate this path with a suitable subset of L_l. Let $D \subseteq L_l \cap \mathsf{Base}(\mathsf{Left}(G_{n+1}))$ be the set of nodes that are connected to a node in $\mathsf{In}(\mathsf{Center}(G_{n+1}))$. Then $|D| \geq 2^n - l - c_i$. Consider the path P_D determined by the list of nodes D. This path can be extended with P_F and the resulting path contains at least $2^n - l - c_i + 2^n - r - c + c_i = 2^{n+1} - |R'|$ nodes in $\mathsf{Base}(G_{n+1})$, as was needed. □

Corollary 4. G_{n+1} is $(2^n, 2^n)$-**depth-robust** with respect to $\mathsf{Base}(\mathsf{Right}(G_{n+1}))$.

Proof. Consider a set R of size less than 2^n and the path L given by the previous theorem (the theorem would be applicable even if $2^n \leq |R| < 2^{n+1}$, but we are only using a restricted result). Then $V(L) \cap \mathsf{Base}(G_{n+1}) \geq 2^{n+1} - |R|$. Let $M = V(L) \cap \mathsf{Base}(\mathsf{Right}(G_{n+1}))$. Then $|M| \geq 2^n - |R|$ given:

$$\begin{aligned} 2^n + |M| &= |\mathsf{Base}(\mathsf{Left}(G_{n+1}))| + |M| \\ &\geq |V(L) \cap \mathsf{Base}(G_{n+1})| \geq 2^{n+1} - |R| \end{aligned}$$

We conclude that the $2^n - |R|$ rightmost elements (with respect to L) in M contain each at least $2^{n+1} - |R| - (2^n - |R|) = 2^n$ predecessors in L, as claimed. □

As connectors, we use the butterfly family of graphs [29]. To formally describe these graphs, we need the operator $\bowtie$, defined next.

Definition 35. For two lists of nodes G_1, G_2 such that $|G_1| = |G_2|$ and $G_2 = L : R$, where $|L| = |R|$, we let $G_1 \bowtie G_2 = (G_1 \to G_2) \cup (G_1 \to L : R)$ be the set of edges obtained by taking $G' = G_1 \to G_2$ and then adding to G' an edge from each node in G_1 to the corresponding node in $R : L$.

Butterfly graphs can be defined recursively as follows:

- $V(H_0) = \{v_0, v_1\}, E(H_0) = \{(v_0, v_1)\}$: H_0 is a single edge

- The graph H_n contains 4 components (L, R, I, O), where:
  - I and O are two lists of 2^n nodes each
  - L and R are fresh copies of H_{n-1}

  The vertices of H_n are $V(H_n) = I \cup O \cup V(L) \cup V(R)$ and the edges are:

$$E(H_n) = E(L) \cup E(R) \cup I \bowtie (\mathsf{In}(L) : \mathsf{In}(R)) \cup (\mathsf{Out}(L) : \mathsf{Out}(R)) \bowtie O$$

  We let $\mathsf{In}(H_n) = I$ and $\mathsf{Out}(H_n) = O$. A pair of examples are shown in Figure 8.4.

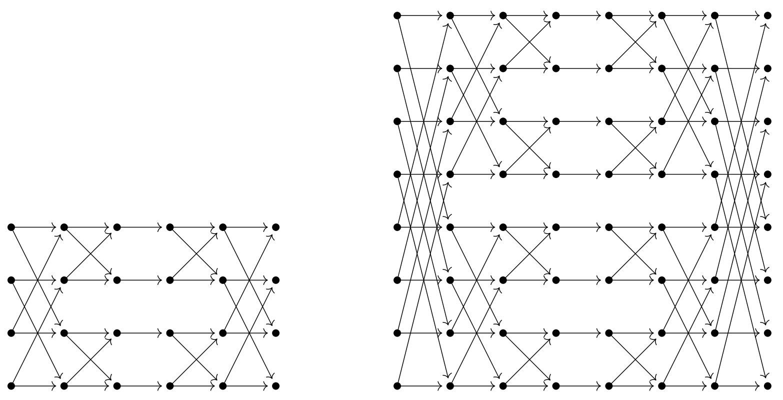


Figure 8.4: Butterfly graphs H_2 and H_3

Butterfly graphs can be easily labelled **in-place**, and allow the next result to hold:

Lemma 24. *G_{n+1} can be labelled* **in-place** *with respect to $\mathsf{Base}(G_{n+1})$, and also with respect to $\mathsf{Base}(\mathsf{Right}(G_{n+1}))$, in $\mathcal{O}(2^{n+1} \cdot (n+1)^2)$ time.*

proof sketch. We apply induction on both statements at the same time. For the induction step of the first statement, notice that in order to label G_{n+1} with respect to its Base, we can first label $\mathsf{Left}(G_{n+1})$ using $2^n \cdot w + \mathcal{O}(w)$ memory. Then, while keeping the previous labels stored, compute the labels of $\mathsf{In}(\mathsf{Center}(G_{n+1}))$ using $2^n \cdot w + \mathcal{O}(w)$ extra memory. From these we can compute **in-place** the labels of the nodes in $\mathsf{Center}(G_{n+1})$ and later $\mathsf{Out}(\mathsf{Center}(G_{n+1})) \triangleright \mathsf{Right}(G_{n+1})$, as these subgraphs only contain copies of butterfly graphs, which can be easily labelled **in-place**. Then, by the induction hypothesis, compute the labels of $\mathsf{Base}(\mathsf{Right}(G_{n+1}))$ using the same memory as before. The proof for the induction step for the second statement is the same, except that the labels of $\mathsf{Base}(\mathsf{Left}(G_{n+1}))$ are not stored, so this memory is overwritten afterwards. □

The proposed protocol for memory-erasure depends on a graph and a subset of its nodes $O(G)$. In particular, the memory size in words needs to be the same as the $O(G)$. Thus, the graphs constructed previously can only be used if the memory size is a power of two, which may not be true in practice. The next results solve this issue: they show how to construct **depth-robust** graphs that can be labelled **in-place** with respect to a subset of nodes of arbitrary size.

Lemma 25. Let G_1 be (a_1, b)-**depth-robust** with respect to $O(G_1)$ and G_2 be (a_2, b)-**depth-robust** with respect to $O(G_2)$. Then the graph G defined by the disjoint union G_1 and G_2 is $(a_1 + a_2, b)$-**depth-robust** with respect to $O(G) = O(G_1) \cup O(G_2)$.

Proof. Let R be a set of nodes in G with $|R| < a_1 + a_2$. Let $r_1 = |R \cap V(G_1)|$ and $r_2 = |R \cap V(G_2)|$. Notice that, if $r_1 < a_1$ then by the depth-robustness of G_1 there are at least $a_1 - r_1$ nodes v in $G_1 \setminus R$ with $\mathsf{llp}(v) \geq b$, and the analogous result is true for G_2. We conclude by case analysis:

- if $r_2 \geq a_2 \implies r_1 < a_1$, from which the result follows given that $a_1 - r_1 \geq a_1 + a_2 - r_1 - r_2$.
- if $r_1 \geq a_1$ the result follows by symmetry with the previous case.
- else $r_2 < a_2 \wedge r_1 < a_1$, which means that we can find $a - r_2$ paths in G_2 and $a - r_1$ paths in G_1 with length greater than b, ending in different vertices in $O(G_1) \cup O(G_2)$.

□

Lemma 26. Let G be (a, b)-**depth-robust** with respect to $O(G)$. Let O' be a subset of $O(G)$ of size a'. Then G is (a', b)-**depth-robust** with respect to O'.

Proof. Let R' be a set of nodes with $|R'| < a'$. Let $R = R' \cup (O(G) \setminus O')$. Notice that $|R| < a$. Then the nodes with longs paths guaranteed by the depth robustness of G in $G \setminus R$ will also be nodes with long paths in $G \setminus R'$. Moreover, by definition of R these nodes are in O' and there are at least $a - |R| = a' - |R'|$ such nodes, as needed. □

Lemma 27. Let G be a graph that can be labelled **in-place** with respect to $O(G)$, and G' the graph defined by the disjoint union of two copies of G, G_1 and G_2. If O' is a subset of $O(G_2)$, then G' can be labelled **in-place** with respect to $O(G_1) \cup O'$.

Proof. First compute the labels of $O(G_2)$ **in-place**. Then keep storing only the labels from O'. With the space left, compute the labels of $O(G_1)$ **in-place**. The result follows. □

Theorem 6. Let m be any memory size, and let n be the smallest integer such that $2^{n+1} \geq m$. Then there exists a $(m, 2^n)$-**depth-robust** graph G with respect to $O(G)$ that can be labelled **in-place** with respect to $O(G)$.

Proof. Take $G = C_1 \cup C_2$ where $C_1 \cong C_2 \cong G_{n+1}$ (as defined in our construction). By Corollary 4 and Lemma 25, G is $(2^{n+1}, 2^n)$-**depth-robust** with respect to $O(C_1) \cup O(C_2)$. Let O' be a subset of $O(C_2)$ with size $m - 2^n$, and let $O(G) = O(C_1) \cup O'$. Then by Lemma 26 G is $(m, 2^n)$-**depth-robust** with respect to $O(G)$. Furthermore, by Lemma 24 and Lemma 27 G can be labelled **in-place** with respect to $O(G)$, as claimed. □

8.4 Lightweight protocol

In this section we focus on improving our graph-based memory-erasure protocol. To this end, we construct new graphs with relaxed depth-robustness properties, which will help boost the performance of the protocol by a polylogarithmic factor. This will have the cost of offering slightly less security. Hence, we suggest this lightweight version is used when more performance is needed, and the original one when the highest level of security is required.

An insight into why a more efficient protocol is possible comes up by analysing the trade-off between depth of the graph and protocol security in Corollary 3, which we reproduce to the reader's advantage.

Corollary 3. Assume the graph-based PoSE scheme is instantiated with parameters (m, r, w) and an (m, γ)-**depth-robust** graph G. Then, for any (M, q)-bounded adversary $(\mathcal{A}_0, \mathcal{A}_1)$, with $q < \gamma$, we have:

$$\Pr_h[\mathsf{Exp}^{m,r,w}_{\mathcal{A}_0,\mathcal{A}_1} = \mathsf{true}] \leq (M'/m)^r + 2^{-w_0} \text{ where:}$$

- if $\mathcal{A}$ is *graph-restricted*: $w_0 = w$ and $M' = \lceil M/w \rceil$
- else: $w_0 = w - \log(m) - \log(q)$ and $M' = \lceil M/w_0 \rceil$

Notice that the parameter γ (which represents the depth of paths) in the **depth-robustness** property must be greater than the number of queries the adversary can do within a fast phase round. This is the only restriction this parameter has. For the family of graphs constructed in Section 8.3, γ is at least m, the number of output nodes in the graph. Note that this is quite high, considering that a realistic attacker may at most do a small constant amount of operations during the fast phase, and that the operations we bound are hash computations. Therefore, for κ a small constant and each m, we construct a (m, κ)-**depth-robust** graph G'_m. Applying Corollary 3 to G'_m we obtain a secure protocol against $(M, \kappa - 1)$-bounded adversaries with exactly the same security bounds.

For the sake of simplicity, assume κ is fixed in a small power of 2, for example 16. Then, the graph G_4 from the previous section is a $(16, \kappa)$-**depth-robust** graph with respect to $O(G_4) = \{o_1, o_2, \dots, o_{16}\}$.

Definition 36. Let Q_i for $1 \leq i \leq 16$ be the subgraph of G_4 that contains all nodes which belong to some path ending in a node from $\{o_1, \dots, o_i\}$, and $O(Q_i) = (o_1, \dots, o_i)$.

Lemma 28. Q_i is (i, κ)-**depth-robust** with respect to $O(Q_i)$.

Proof. Notice that, by definition, Q_i is a subgraph of G_4 (which is $(16, \kappa)$-**depth-robust**) and $O(Q_i) \subseteq O(G_4) \wedge |O(Q_i)| = i$. Then, the result follows from Lemma 26. □

Definition 37. Let G'_m be a graph and $O(G'_m)$ a subset of nodes defined as follows. Let $m = 16 \cdot k + i$ be the result of the Euclidean division lemma.

- If $i = 0 \wedge k = 0$: G'_m is the empty graph
- If $i = 0$ then: G'_m is the disjoint union of k graphs isomorphic with G_4, and $O(G'_m)$ the union of the corresponding nodes in each copy of $O(G_4)$.

- If $i > 0$: G'_m is the disjoint union of k copies of G'_{16} and Q_i, and $O(G'_m)$ the union of the corresponding nodes in each copy of $O(G'_{16})$ and $O(Q_i)$.

Lemma 29. The graph G'_m is (m, κ)-**depth-robust** with respect to $O(G'_m)$.

Proof. By definition, if $m = 16 \cdot k + i$, G'_m is the disjoint union of k copies of G_4, which is $(16, \kappa)$-**depth-robust**, and a copy of Q_i which is (i, κ)-**depth-robust**. The result follows by applying Lemma 25 to these disjoint graphs which form G'_m. □

Lemma 30. If $m \geq 16$, the graph G'_m can be labelled **in-place** with respect to $O(G'_m)$.

sketch. We label the output nodes of each component of G'_m independently. First, we compute and store the labels the nodes in Q_i, which can be done using at most 16 words (because this is a subgraph of G_4). For this to hold the condition $m \geq 16$ is necessary. The remaining space can be used to compute and store the output labels in each copy of G_4, using the **in-place** properties of these graphs. □

By the previous lemmas, we conclude that the graph G'_m has the required properties to serve as the base of our secure memory-erasure protocol. It remains to compute how many hash computations are done by a prover using our protocol instantiated with G'_m, in comparison to the original proposal. For simplicity, we analyse the case where $m = 2^k$ with $k > 3$.

Lemma 31. If $m = 2^k$, labelling G_k requires exactly $(k^2 + k + 3) \cdot 2^{k+1} - 2$ hash computations and labelling G'_m requires exactly $367 \cdot 2^{k-3}$ hash computations. Therefore, the lightweight protocol is approximately $\frac{k^2+k+3}{23}$ times more efficient.

Proof. Both graphs can be labelled optimally by the results shown in the previous section (Lemmas 24 and 27 and theorem 6). Therefore, it remains to count the exact number of nodes in G_k, from which we can easily obtain the exact amount for G'_m.

By the recursive definition of G_k, we have that its number of nodes is equal to:

$$2 \cdot |G_{k-1}| + \sum_{i=0}^{k} |H_i|$$

The butterfly graph H_i has exactly $2 \cdot (i + 1) \cdot 2^i$ nodes. Therefore, we can derive the recurrent equation:

$$|G_k| = 7 \cdot |G_{k-1}| - 18 \cdot |G_{k-2}| + 20 \cdot |G_{k-3}| - 8 \cdot |G_{k-4}|$$

with initial conditions $|G_0| = 4, |G_1| = 18, |G_2| = 70, |G_3| = 238$. The exact solution to this recurrence is $|G_k| = (k^2 + k + 3) \cdot 2^{k+1} - 2$. From there we obtain that $|G_4| = 23 \cdot 32 - 2$ and $|G'_m| = 2^{k-4} \cdot (23 \cdot 32 - 2)$. Therefore, by removing the small additive terms, we deduce that the lightweight protocol is approximately $\frac{k^2+k+3}{23}$ times more efficient, as claimed. □

8.5 Conclusions

We presented a graph-based provable secure PoSE protocol capable of deterring provers from outsourcing the erasure proof to an external conspirator. The protocol asks the prover to label a depth-robust graph from a random seed. Because the labelling algorithm is implemented by a resource-constrained device, we introduced a class of graphs with depth-robust properties that can be labelled in-place using hash functions. The protocol was proven secure within a formal model, guaranteeing that all the prover's memory, except for a small part, is erased. The security bounds are tighter against a restricted, yet plausible, adversary. Hence, future work directed to closing such gap is needed. Finally, we proposed a lightweight version of the protocol based on a smaller graph by trading security for performance.

9 Practical implementations of memory-erasure protocols

Although several memory-erasure protocols have been proposed in the literature during the last decade, we are not aware of any deployment of these protocols in the real world. Given that the protocols are not very hard to implement, we believe the reasons for this lack of popularity are the following: first, there are no available proof-of-concept implementations of the protocols, and second, it is not straightforward to compare their performance, feasibility, or security guarantees. This chapter aims to fill such gap. First we compare the most prominent memory-erasure protocols from the literature with respect to security assumptions and guarantees. Then we implement and deploy them in 3 different IoT devices. Finally, we compare their performance with respect to execution time and memory footprint.

9.1 Memory-erasure protocols

We selected 4 protocols from the literature and 3 protocols proposed in this thesis:

- **PT** [111]: This was the first PoSE protocol, proposed by Perito and Tsudik. At the beginning, the verifier sends a fresh random value of the same size as the prover's memory. The prover computes the HMAC of this value, using as key the last bits received. The security proof of this protocol relies on the fact that to start the computation of the HMAC of a random value the prover needs to first receive the key. Its main drawback being that the verifier needs to send a message as big as the prover's memory. Furthermore, the proof of security is informal, and does not specify which assumptions must the HMAC function fulfil to make the protocol secure.

- **DFKP** [60]: This is the first memory-erasure protocol based on pebbling games, by Dziembowski et al. Its main contribution is the reduction of the communication complexity from linear to constant. It uses the pyramid graph to construct the labelling function used to erase memory. Its proof of security is relatively straightforward, but the size of the constructed graph is quadratic in the prover's size, which might be too inefficient.

- **KK** [81]: This is another protocol based on pebbling games, by Karvelas and Kiayias. Its main contribution is the construction of a new graph with quasilinear size which leads to a more efficient protocol. The proof of security is complicated, as the authors prove that the pebbling complexity of a certain graph is a fraction of the size of the graph itself. Its main drawback is that it can only provable erase $\frac{1}{32}$ of the prover's memory.

- **KL** [79]: This protocol, by Karame and Li, is an improvement of the Perito's protocol that uses the all or nothing transform (AONT), a novel cryptographic primitive, to make the erasure procedure more efficient. Its main drawbacks remain the same though, as it requires linear communication complexity and does not include a formal proof of security.
- **PoSE$_{\textbf{random}}$**: The simple protocol proposed in Chapter 6.
- **PoSE$_{\textbf{graph}}$**: The graph-based protocol proposed in Chapter 8.
- **PoSE$_{\textbf{light}}$**: The lightweight version of the graph-based protocol proposed in Section 8.4.

We focused our selection in software-based memory-erasure protocols only, but did not include protocols such as SPEED [9], based on memory isolation techniques. Comparing against this type of protocol would require implementing the memory isolation components, which is more device dependent, thus considered out of the scope.

	No Isolation	Proof	Prob	Pebble	Erasure	Time	Comm
DFKP	✗	✓	✗	✓	1	$\mathcal{O}(n^2)$	$\mathcal{O}(1)$
KL	✗	✗	✗	✗	1	$\mathcal{O}(n)$	$\mathcal{O}(n)$
KK	✗	✓	✗	✓	$\frac{1}{32}$	$\tilde{\mathcal{O}}(n)$	$\mathcal{O}(1)$
PT	✗	✗	✗	✗	1	$\mathcal{O}(n)$	$\mathcal{O}(n)$
PoSE$_{\textbf{graph}}$	✓	✓	✓	✓	1	$\tilde{\mathcal{O}}(n+r)$	$\mathcal{O}(r)$
PoSE$_{\textbf{light}}$	✓	✓	✓	✓	1	$\mathcal{O}(n+r)$	$\mathcal{O}(r)$
PoSE$_{\textbf{random}}$	✓	✓	✓	✗	1	$\mathcal{O}(n+r)$	$\mathcal{O}(n+r)$

Table 9.1: Characteristics of the implemented memory erasure protocols. We measure the complexity of the execution time and communication using parameters n, the memory size, and r, the number of rounds

On Table 9.1 we compare the protocols with respect to:

- isolation: whether the protocol assumes that the device is isolated during the execution of the protocol
- proof: whether the proof of security is formal
- prob: whether there are upper bounds to the probability of success of the attacker
- pebble: whether the protocol is based on pebble games
- erasure: the proportion of the device's memory that can be erased
- time: the time complexity of running a single session of the protocol
- comm: the communication complexity of running a single session of the protocol

Notice that all protocols with less than linear communication complexity are based on pebble games.

Hash functions

In our experiments we tested each protocol with different hash functions. We selected these functions according to popularity, amenability to software-based implementations, and applicability to the IoT space. All of them output 256 bits of digest. The following hash functions were used:

- ascon [54]: a sponge based hash function selected in the Lightweight Cryptography Standardization Process by NIST[1].
- blake2 [14]: a widely deployed and highly efficient hash function. It was designed to be especially performant in software implementations.
- blake3[2]: a recent improvement on the blake2 hash function which is claimed to be much faster while offering similar security guarantees. This is the fastest (in software implementations) cryptographic hash function we could find.
- sha256 [61]: a well known and widely used hash function based on the Merkle-Damgård construction proposed by NIST. Up to this day, it is still considered secure, although is prone to length extension attacks [131].
- aeshash[3]: an unpublished hash function whose core utilizes AES instructions. It was selected mainly to check how much speed up could be achieved in a device with only an AES accelerator (FR5994). As far as we are aware it has not been thoroughly analysed, so it does not offer any security guarantees and is only used here for the purpose mentioned before.

9.2 Experiments

For our experiments we considered 3 IoT microcontrollers produced by Texas Instruments with different characteristics: FR5994, F5529, CC2652. These devices acted as provers in the memory-erasure protocol. All prover implementations were done in Portable C, making them usable in any platform or architecture. To create the binaries, we configured the compiler to optimize for code size. A personal Dell laptop acted as verifier, running Python 3 implementations of each protocol. The Bluetooth protocol was used as communication channel between the verifier and the prover. The distance from the device (prover) to the laptop (verifier) was approximately 1 meter. Each run of the protocol was done separately, as the main objective was to compare the protocols against each other in the simplest possible setting.

9.2.1 Device characteristics

In Table 9.2 we compare each device with respect to Memory (code size + data size), maximum clock speed, on device crypto accelerators, Bluetooth version, architecture, microcontroller unit family (MCU), and IoT class [30]. Notice that some devices have hardware accelerators. For these, we also experimented using a hardware accelerated version of the hash function.

[1]`https://csrc.nist.gov/News/2023/lightweight-cryptography-nist-selects-ascon`

[2]`https://github.com/BLAKE3-team/BLAKE3-specs/blob/master/blake3.pdf`

[3]`https://csrc.nist.rip/groups/ST/toolkit/BCM/documents/proposedmodes/aes-hash/aeshash.pdf`

Device	Memory	Clock	Crypto	BT	Arch	MCU	IoT
F5529	$(128+10)$KB	25MHz	—	2	RISC-16	MSP430	Class 1
FR5994	$(256+8)$KB	16MHz	AES	2	RISC-16	MSP430	Class 1-2
CC2652	$(352+88)$KB	48MHz	AES,SHA2	5.2	RISC-32	Arm Cortex-M	Class 2

Table 9.2: Microcontroller characteristics

Amount of memory erased

To be able to run all protocols in reasonable time, while erasing the same amount of memory with each, and avoid the need for device specific engineering, in our experiments we erase only a portion of the memory of each device. In particular for the devices F5529 and FR5994 we erase exactly 2^6 words, which is equivalent to 2 KB, and for CC2652 we erase exactly 8 KB. We acknowledge this is a limitation of our results, and something that needs to be solved before these protocols are deployed in a real setting.

Number of rounds

As seen before, the protocols proposed in this thesis use a parameter r, the number of rounds during the fast phase. This parameter, as shown in previous chapters, is directly related to the security guarantees of the protocol: the more rounds the lower the probability of success of the adversary. For our experiments we selected a very conservative number of rounds in order to test our protocols in worst case conditions, and set the number of rounds equal to the number of words in the prover's memory.

9.2.2 Memory footprint results

In this section we show the memory footprint of each combination of protocol, hash function and device. It is worth mentioned that the sizes vary between devices because their architectures are different. Another reason is that we used the best available implementation for each hash function and architecture.

	aeshash	ascon	blake2	blake3	sha256	sha256hw
DFKP	1249	5858	6363	21180	1040	1397
KL	1242	5882	6362	21221	1041	1396
KK	1255	5866	6364	21158	1043	1391
PT	—	—	—	—	3779	—
$\mathbf{PoSE_{graph}}$	1259	5878	6374	21181	1050	1408
$\mathbf{PoSE_{light}}$	1250	5866	6365	21193	1041	1400
$\mathbf{PoSE_{random}}$	—	—	—	—	—	—

Table 9.3: Memory footprint of the hash function in device CC2652

Table 9.3 shows the memory footprint of the hash in the CC2652 device for each combination of hash and each protocol. It can be observed that the memory occupied by each hash function is the same for each protocol, as expected, except sha256 and the **PT** protocol. The reason is that this protocol uses a HMAC, with sha256 as underlying hash, which requires more space than the plain sha256 implementation. One surprising result is that the memory footprint of the sha256hw function, which uses the hardware accelerator, actually occupies more space than the software-based implementation sha256. This must be the result of the extra code necessary to access the hardware module.

	aeshash	ascon	blake2	blake3	sha256
DFKP	634	2110	14022	24720	2292
KL	634	2128	13960	24656	2168
KK	634	2128	13960	24657	2230
PT	—	—	—	—	2189
$\mathbf{PoSE_{graph}}$	634	2110	14022	24720	2292
$\mathbf{PoSE_{light}}$	634	2110	14022	24721	2292
$\mathbf{PoSE_{random}}$	—	—	—	—	—

Table 9.4: Memory footprint of the hash function in device FR5994

	ascon	blake2	blake3	sha256
DFKP	2110	14022	22821	2292
KL	2128	13960	22760	2168
KK	2128	13960	22758	2230
PT	—	—	—	2197
$\mathbf{PoSE_{graph}}$	2110	14022	22822	2292
$\mathbf{PoSE_{light}}$	2110	14022	22823	2292
$\mathbf{PoSE_{random}}$	—	—	—	—

Table 9.5: Memory footprint of the hash function in device F5529

Tables 9.4 and 9.5 show similar results for devices FR5994 and F5529. One noticeable exception is that on these devices the blake2 memory footprint is much higher with respect to the other hash functions. This must be due to the use of a different C implementation, as the one used for the CC2652 is optimized for the 32 bit architecture, which is not suitable for the others. Taking into account the three devices, sha256 is the hash function with a lower memory footprint, followed by ascon and aeshash.

Table 9.6 shows the memory footprint of each algorithm on the CC2652 device. In this case the memory used by each algorithm does not change when a different hash function is used, as expected.

Similar results for devices FR5994 and F5529 are shown in Tables 9.7 and 9.8. In general, the CC2652 implementations take more space than the F5529 or FR5994 ones. This is probably due to the use of byte size variables throughout the implementations, which can be more efficiently represented in the 16 bits architecture than in the 32 bits architecture. Taking into account the three devices, the protocols with smaller memory footprint were **PT**, $\mathbf{PoSE_{random}}$ and **DFKP**.

	aeshash	ascon	blake2	blake3	none	sha256	sha256hw
DFKP	593	578	581	598	—	586	587
KL	1021	1037	1015	1074	—	1022	1021
KK	1357	1344	1340	1334	—	1347	1339
PT	—	—	—	—	—	281	—
PoSE$_{\text{graph}}$	1197	1192	1186	1193	—	1190	1192
PoSE$_{\text{light}}$	1190	1182	1179	1207	—	1183	1186
PoSE$_{\text{random}}$	—	—	—	—	320	—	—

Table 9.6: Memory footprint of the protocol implementation in device CC2652

	aeshash	ascon	blake2	blake3	none	sha256
DFKP	776	800	776	777	—	776
KL	1395	1437	1333	1332	—	1271
KK	1905	1947	1843	1843	—	1843
PT	—	—	—	—	—	357
PoSE$_{\text{graph}}$	1605	1629	1605	1606	—	1605
PoSE$_{\text{light}}$	1639	1663	1639	1641	—	1639
PoSE$_{\text{random}}$	—	—	—	—	416	—

Table 9.7: Memory footprint of the protocol implementation in device FR5994

	ascon	blake2	blake3	none	sha256
DFKP	795	771	772	—	771
KL	1436	1332	1334	—	1270
KK	1946	1842	1842	—	1842
PT	—	—	—	—	356
PoSE$_{\text{graph}}$	1628	1604	1606	—	1604
PoSE$_{\text{light}}$	1662	1638	1641	—	1638
PoSE$_{\text{random}}$	—	—	—	415	—

Table 9.8: Memory footprint of the protocol implementation in device F5529

9.2.3 Execution time results

For each protocol execution, the total time is composed by the computation time and the query time. Query time exists only for the protocols in this thesis, and measures the average time in each round of the interactive phase. The computation time is the time spent by the prover to erase the memory, as measured from the verifier's perspective.

The query time values are shown in Tables 9.9 to 9.11. Notice that these measurements do not depend on the hash function used, as expected. The difference in speed between the CC2652 device and the others can be explained by the version of the Bluetooth protocol used. The CC2652 device has a Bluetooth module included, able to run version 5.2, which means in each challenge-response round the message needs to go through the whole Bluetooth stack. On the other hand, for the FR5994 and F5529 devices a simple HC-05 module was used, which makes the Bluetooth protocol overhead much smaller, as it does not support as many features.

	aeshash	ascon	blake2	blake3	none	sha256	sha256hw
$\textbf{PoSE}_{\textbf{graph}}$	0.090	0.090	0.090	0.090	—	0.094	0.090
$\textbf{PoSE}_{\textbf{light}}$	0.091	0.103	0.101	0.091	—	0.090	0.090
$\textbf{PoSE}_{\textbf{random}}$	—	—	—	—	0.090	—	—

Table 9.9: Average query time in seconds on device CC2652

	aeshash	ascon	blake2	blake3	none	sha256
$\textbf{PoSE}_{\textbf{graph}}$	0.068	0.069	0.068	0.067	—	0.069
$\textbf{PoSE}_{\textbf{light}}$	0.070	0.070	0.069	0.070	—	0.068
$\textbf{PoSE}_{\textbf{random}}$	—	—	—	—	0.065	—

Table 9.10: Average query time in seconds on device FR5994

	ascon	blake2	blake3	none	sha256
$\textbf{PoSE}_{\textbf{graph}}$	0.068	0.066	0.067	—	0.068
$\textbf{PoSE}_{\textbf{light}}$	0.069	0.068	0.070	—	0.069
$\textbf{PoSE}_{\textbf{random}}$	—	—	—	0.066	—

Table 9.11: Average query time in seconds on device F5529

The compute time values for every combination of protocol and hash function are shown in Tables 9.12 to 9.14. Notice that for the protocol $\textbf{PoSE}_{\textbf{random}}$ there is no compute time because this protocol does not process any value during the initial phase.

	aeshash	ascon	blake2	blake3	sha256	sha256hw
DFKP	11.0	31.6	5.2	3.9	7.4	1.8
KL	0.1	0.1	0.1	0.1	0.1	0.1
KK	7.5	21.0	3.7	2.8	5.2	1.3
PT	—	—	—	—	0.1	—
$\textbf{PoSE}_{\textbf{graph}}$	12.5	35.6	6.1	4.6	8.8	2.1
$\textbf{PoSE}_{\textbf{light}}$	3.8	10.7	1.9	1.4	2.8	0.7
$\textbf{PoSE}_{\textbf{random}}$	—	—	—	—	—	—

Table 9.12: Compute time in seconds on device CC2652

In general, the protocols with faster compute time were **PT** and **KL**. It is notable faster to erase memory when the device has a higher clock frequency, as can be seen when comparing performance in the CC2652 with respect to the FF5529 and the FR5994 devices. Among the hash functions, sha265hw was the fastest, which was expected as it uses the hardware accelerator present in the CC2652 device. One surprising result is that ascon, which is meant to be used in lightweight

	aeshash	ascon	blake2	blake3	sha256
DFKP	3.2	80.7	20.2	15.7	27.9
KL	1.5	1.8	1.6	1.7	1.7
KK	5.1	131.0	34.6	27.1	48.2
PT	—	—	—	—	1.6
PoSE$_{\text{graph}}$	8.4	215.5	56.7	44.0	78.9
PoSE$_{\text{light}}$	4.3	106.9	28.7	22.5	39.8
PoSE$_{\text{random}}$	—	—	—	—	—

Table 9.13: Compute time in seconds on device FR5994

	ascon	blake2	blake3	sha256
DFKP	47.6	12.7	10.0	17.7
KL	2.2	2.1	2.0	2.1
KK	76.4	21.3	16.7	29.9
PT	—	—	—	2.1
PoSE$_{\text{graph}}$	125.5	34.4	26.6	48.4
PoSE$_{\text{light}}$	63.2	17.8	13.9	25.1
PoSE$_{\text{random}}$	—	—	—	—

Table 9.14: Compute time in seconds on device F5529

cryptography, had a consistently worst performance. In every device blake3 was faster than blake2, although the difference is around 25 percent only. Aeshash was much faster than other hash functions in the FR5994 device, as expected by the usage of the AES hardware module present on this device. On the contrary, it performed worse that blake2, blake3 and even sha256 in the CC2652 device. Therefore, we deduce that if the device is fast enough, pure software implementations might be more performant than hybrid implementations such as the one used in aeshash.

Tables 9.15 to 9.17 show the total execution time for every combination of protocol and hash function.

	aeshash	ascon	blake2	blake3	none	sha256	sha256hw
DFKP	11.1	31.7	5.3	4.0	—	7.5	1.9
KL	23.4	23.4	23.3	23.3	—	23.4	23.3
KK	7.6	21.1	3.8	2.9	—	5.3	1.3
PT	—	—	—	—	—	23.2	—
PoSE$_{\text{graph}}$	37.5	82.0	29.3	27.8	—	32.9	25.2
PoSE$_{\text{light}}$	27.7	44.1	27.9	24.9	—	25.9	23.8
PoSE$_{\text{random}}$	—	—	—	—	46.2	—	—

Table 9.15: Total execution time in seconds on device CC2652

With respect to the total time and the CC2652 device, the most performant protocol was **DFKP**, despite being the one with worst time complexity. This is only possible because the constants hidden in the asymptotic notation are very small for this protocol, and somewhat larger for

	aeshash	ascon	blake2	blake3	none	sha256
DFKP	3.2	80.7	20.2	15.7	—	27.9
KL	2.4	2.6	2.5	2.4	—	2.5
KK	5.1	131.0	34.6	27.1	—	48.2
PT	—	—	—	—	—	2.4
PoSE$_{\textbf{graph}}$	13.0	223.4	61.0	48.4	—	83.4
PoSE$_{\textbf{light}}$	9.0	113.1	33.1	27.1	—	44.2
PoSE$_{\textbf{random}}$	—	—	—	—	6.4	—

Table 9.16: Total execution time in seconds on device FR5994

	ascon	blake2	blake3	none	sha256
DFKP	47.6	12.7	10.0	—	17.7
KL	3.0	2.9	2.9	—	2.9
KK	76.4	21.3	16.7	—	29.9
PT	—	—	—	—	2.9
PoSE$_{\textbf{graph}}$	133.2	38.6	30.9	—	52.8
PoSE$_{\textbf{light}}$	69.2	22.2	18.4	—	29.5
PoSE$_{\textbf{random}}$	—	—	—	6.5	—

Table 9.17: Total execution time in seconds on device F5529

the others, and the amount of memory erased is not too big. The total time of **KL** and **PT** was consistently slower than **KK** and **DFKP**, possibly due to the fact that on this device communication complexity played a larger role. Our protocols were consistently slower than previous ones, as a result of using numerous of rounds during the fast phase.

9.3 Analysis

Our results have not revealed a clear winner with respect to security guarantees and performance. Therefore, we deduce that the use of a certain protocol depends on the specific conditions in which it will operate. In particular, clock speed, network cost, memory size, and security guarantees determine which protocol is more suitable.

To reach our conclusions, we considered the following facts:

- Protocols **KL**, **KK** and **PT** offer a low level of security
- Protocols **PoSE$_{\textbf{random}}$**, **PoSE$_{\textbf{graph}}$**, **PoSE$_{\textbf{light}}$** and **DFKP** offer a high level of security. Note, though, that the **DFKP** protocol is less secure than the others given that it is not resistant against distant attackers.
- Protocols **PoSE$_{\textbf{random}}$**, **PT** or **KL** require sending the full memory of the device through the network
- **DFKP** has quadratic complexity, therefore its performance is worse when the amount of memory is relatively large

				Network cost			
				High		Low	
				Clock speed		Clock speed	
				High	Low	High	Low
Memory size	Large	Security	High	$\mathbf{PoSE_{light}}$	$\mathbf{PoSE_{light}}$	$\mathbf{PoSE_{random}}$	$\mathbf{PoSE_{random}}$
			Low	**KK**	**KK**	**PT**	**PT**
	Small	Security	High	**DFKP**	**DFKP**	**DFKP**	$\mathbf{PoSE_{random}}$
			Low	**DFKP**	**DFKP**	**KK**	**PT**

Table 9.18: Summary of results

Next, we provide a granular analysis of the results, by projecting them into specific use-cases. These results are summarized in Table 9.18. They were obtained by extrapolating the behaviour of the protocols in each device.

- When the network cost is high, communication needs to be minimized, therefore protocols such as $\mathbf{PoSE_{random}}$, **PT** or **KL** are impractical.
  - When memory is small, protocol **DFKP** is the clear winner (see for example Table 9.17). Even if it has quadratic complexity, its very low constant factor makes it faster than the rest of the protocols. We are considering that this protocol has high security even though it assumes the isolation assumption. If security against distant attackers is necessary, the winner for this category is the $\mathbf{PoSE_{light}}$ protocol.
  - When memory is large, the most performant protocols are $\mathbf{PoSE_{light}}$ and **KK**, for high security and low security respectively.
- When the network cost is low, the most performant protocols are $\mathbf{PoSE_{random}}$ and **PT**, for high security and low security respectively. A surprising result in this setting is that **KL** had a very similar performance as **PT**, which should have not been the case as it was specifically designed to improve its performance. If the device in question has hardware accelerators, then for the high security case it is possible that **DFKP** is faster, as shown for example in Table 9.15.

It is noteworthy that the clock speed only changed the selection of the winner when the network cost is low and memory is small. In this scenario, the **DFKP** protocol outperforms $\mathbf{PoSE_{random}}$ and **KK** outperforms **PT** (see for example Table 9.15).

9.4 Conclusions

In this chapter we presented the outcome of our memory-erasure experiments. We implemented[4] 7 protocols, each with several variants depending on the hash function used, and tested them on 3 modern IoT devices. Furthermore, we compared the security guarantees provided by each protocol, and contrasted them with their performance in a practical setting.

Our results revealed that current memory-erasure protocols are practical, although erasing the full memory securely could take several minutes for the slower devices. Network speed might be

[4] `https://gitlab.uni.lu/regil/memory-erasure-experiments`

faster than local computation, therefore reducing the communication complexity in the protocol is not always the best choice. For protocols where hash functions are used, this choice may influence dramatically the protocol performance and memory footprint. Finally, the most performant protocol might not be the best according to the asymptotic complexity analysis, as for small memory sizes the hidden constants may play a determinant role.

10 Optimal distance-bounding

In previous chapters we analysed and designed memory-erasure protocols. This required studying trade-offs between time, memory and security. These type of computational results are also useful in other classes of protocols which also depend on time, such as the well-known distance-bounding protocols.

As mentioned in the introduction, distance-bounding protocols are security protocols with a time measurement phase used to detect relay attacks, whose security is typically measured against mafia-fraud and distance-fraud attacks. A prominent subclass of distance-bounding protocols, known as *lookup-based protocols*, use simple lookup operations to reduce the impact of the computation time in the distance calculation. Independent results have found theoretical lower bounds $\frac{1}{2^n}\left(\frac{n}{2}+1\right)$ and $\frac{1}{2^n}$, where n is the number of time measurement rounds, on the security of lookup-based protocols against mafia and distance-fraud attacks, respectively. However, it is still an open question whether there exists a protocol achieving both security bounds. This chapter closes this question in two ways. First, we prove that the two lower bounds are mutually exclusive, meaning that there does not exist a lookup-based protocol that provides optimal protection against both types of attacks. Second, we provide a lookup-based protocol that approximates those bounds by a small constant factor. Our experiments show that, restricted to a memory size that linearly grows with n, our protocol offers strictly better security than previous lookup-based protocols against both types of fraud.

10.1 Introduction

Distance-bounding protocols aim to counteract relay attacks against wireless authentication protocols by ensuring proximity between communicating parties. Because the speed of light represents a hard limit on the speed of radio waves, the distance from a transmitter to a receiver is (at most) half the round-trip-time multiplied by the speed of light, where round-trip-time is the time it takes for a signal to travel from the transmitter to the receiver and back. This physical law allows distance-bounding protocols to verify proximity by measuring the round-trip-time of a message exchange between a prover and a verifier [53].

Many protocols have suffered relay attacks in the past [17]. Famously, the EMV standard for contactless payments [62] now offers some protection against this type of attacks, and NXP's MIFARE Plus cards support proximity checks [127]. Both protocols use distance-bounding as their main protection mechanism against relay attacks and have been extensively studied.

In general, a distance-bounding protocol consists of several rounds, in each of which a verifier challenges a prover to answer a query based on a shared secret key. Each round is timed by the verifier to check whether the prover is close. As a case in point, consider the protocol introduced by Hancke and Kuhn [76] depicted in Figure 10.1. First, the parties exchange one nonce each (N_v and N_p). The two nonces and a shared secret key k are used by both parties as input to a pseudo-random function PRF, which outputs a $2n$-bit sequence $T_1 \cdots T_{2n}$[1]. Next, prover and verifier engage in n challenge-response rounds. At the i-th round, the verifier generates and sends a random bit-challenge c_i. If $c_i = 0$, the prover replies with T_{2i-1}, otherwise with T_{2i}. The protocol succeeds if all responses are correct and all round-trip-times are below a predefined threshold $\Delta_{\max}$. If $\Delta_{\max}$ is sufficiently small, then the protocol ensures proximity of the prover to the verifier, because the correct responses can only be generated by a nearby prover with the correct key k.

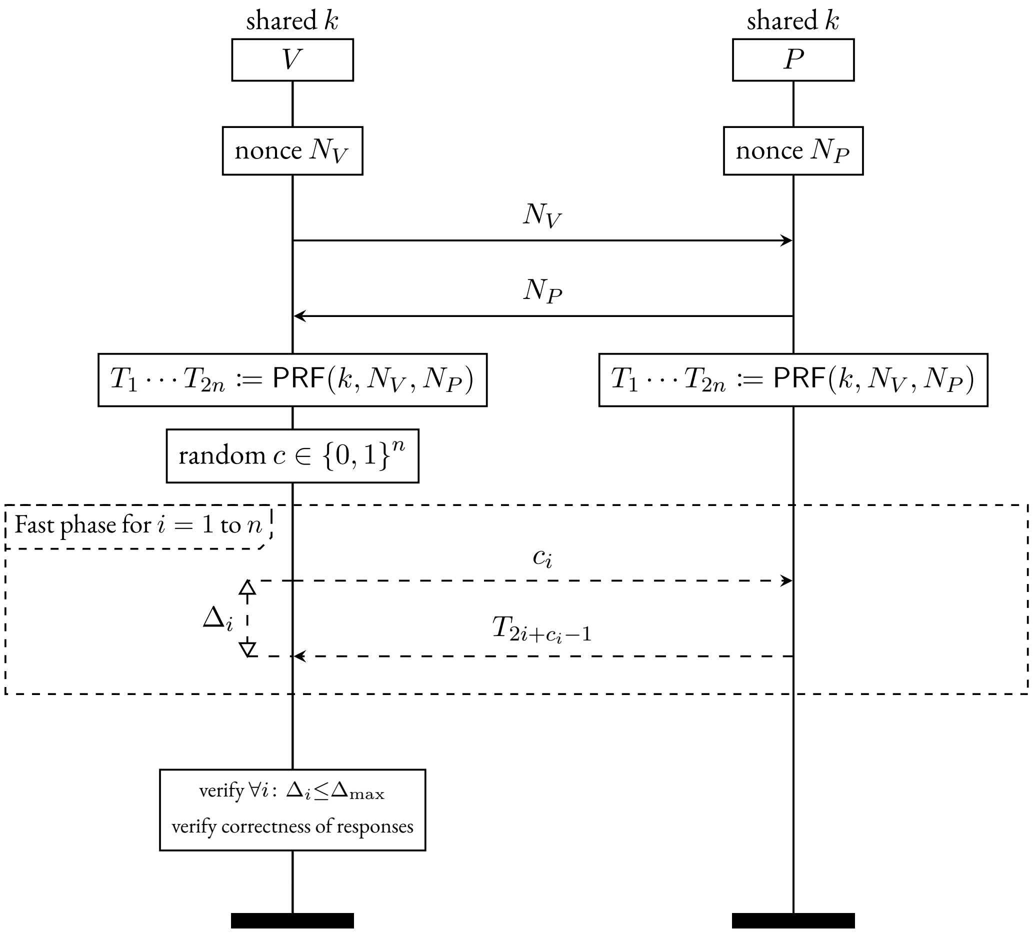
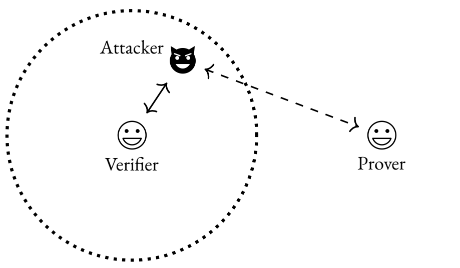

shared k
V
shared k
P
nonce N_V
nonce N_P
N_V
N_P
$T_1 \cdots T_{2n} := \mathsf{PRF}(k, N_V, N_P)$
$T_1 \cdots T_{2n} := \mathsf{PRF}(k, N_V, N_P)$
random $c \in \{0,1\}^n$
Fast phase for $i = 1$ to n
c_i
Δ_i
T_{2i+c_i-1}
verify $\forall i$: $\Delta_i \leq \Delta_{\max}$
verify correctness of responses

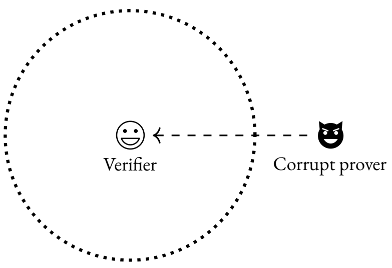
Figure 10.1: A lookup-based protocol by Hancke and Kuhn.

Hancke and Khun's protocol has the virtue of relying on a simple lookup operation by the prover during the time-measurement phase, minimizing the impact that the prover's computational cost has on the round-trip-time measurement. Moreover, it makes a single call to a keyed pseudo-random function, making it suitable for resource-constrained devices.

[1]For the sake of rigorousness, we should point out that [76] describes the protocol by using two registers of length n rather than a single bit-sequence of length $2n$.



(a) **Mafia fraud:** the verifier believes the prover is close.



(b) **Distance fraud:** the verifier believes the corrupt prover is close.

Figure 10.2: The difference between mafia fraud and distance fraud. The encircled area denotes proximity to the verifier.

Hancke and Khun's protocol, however, offers suboptimal security. Hence, distance bounding protocols improving upon Hancke and Khun's design have been extensively studied [20, 76, 80, 84, 96, 103, 108, 128, 130, 132], and given the name of *lookup-based protocols* [96]. These protocols usually include a precomputation phase and a fast phase. This is different from more general distance-bounding protocols that may include a verification phase. Moreover, during each round of the fast phase the prover executes a single lookup operation, i.e. the response is already computed in memory.

The analysis of lookup-based protocols has mainly focused on two potential threats: *mafia fraud* and *distance fraud* [15, 16]. The former is a man-in-the-middle attack aimed at convincing the verifier that an honest far-away prover is close (see Figure 10.2a); the latter is an attack executed by a corrupt prover to pass the protocol from far-away (see Figure 10.2b). It was proven in [96] that the optimal resistance against mafia fraud is $\frac{1}{2^n}\left(\frac{n}{2}+1\right)$, where n is the number of round-trip-time measurements, and in [128] that the optimal resistance against distance fraud is $\frac{1}{2^n}$. It remains an open question, however, whether there exists a lookup-based protocol achieving both optimal security bounds.

Contributions and structure. This chapter addresses the open question of whether lookup-based protocols can achieve optimal resistance against mafia fraud and distance fraud. We do so by extending the security model used in [96] to model lookup-based protocols with a game-based definition of security (Section 10.2). The resulting model is more general, in the sense that it makes fewer security assumptions, and allows us to prove that optimal security against mafia fraud and distance fraud are conflicting goals (Section 10.3). That is, optimality against one fraud moves the protocol away from optimality against the other fraud.

The second contribution of this work is the introduction of the first lookup-based protocol whose security bounds are either optimal or close to the optimal by a small constant factor (Section 10.4). More precisely, our protocol achieves $\frac{1}{2^n}$ security against distance fraud, which is optimal, and $\frac{1}{2^n}(n+1)$ security against mafia fraud, which is less than twice the optimal.

By comparing those bounds with the security offered by existing lookup-based protocols (Section 10.5), we show that our protocol represents a significant improvement over the state-of-the-art.

10.2 The security model

Our security model takes the specification and execution model of [96], which defines a lookup-based protocol as a set of automata, but defines the security properties of interest in a more general way.

Lookup-based protocols contain two phases: a precomputation phase and a fast phase. During the first phase the prover and verifier exchange messages containing cryptographic information. As a result, both agents generate the same fresh random automaton, which is then used during the fast phase by the prover to compute the responses to the challenges generated randomly by the verifier. Both challenges and responses are single bits. For the security analysis of this type of protocol, it is sufficient to analyse the set of all possible automata that can be generated during the precomputation phase, as in most cases the probability distribution used is (or is assumed to be) uniform.

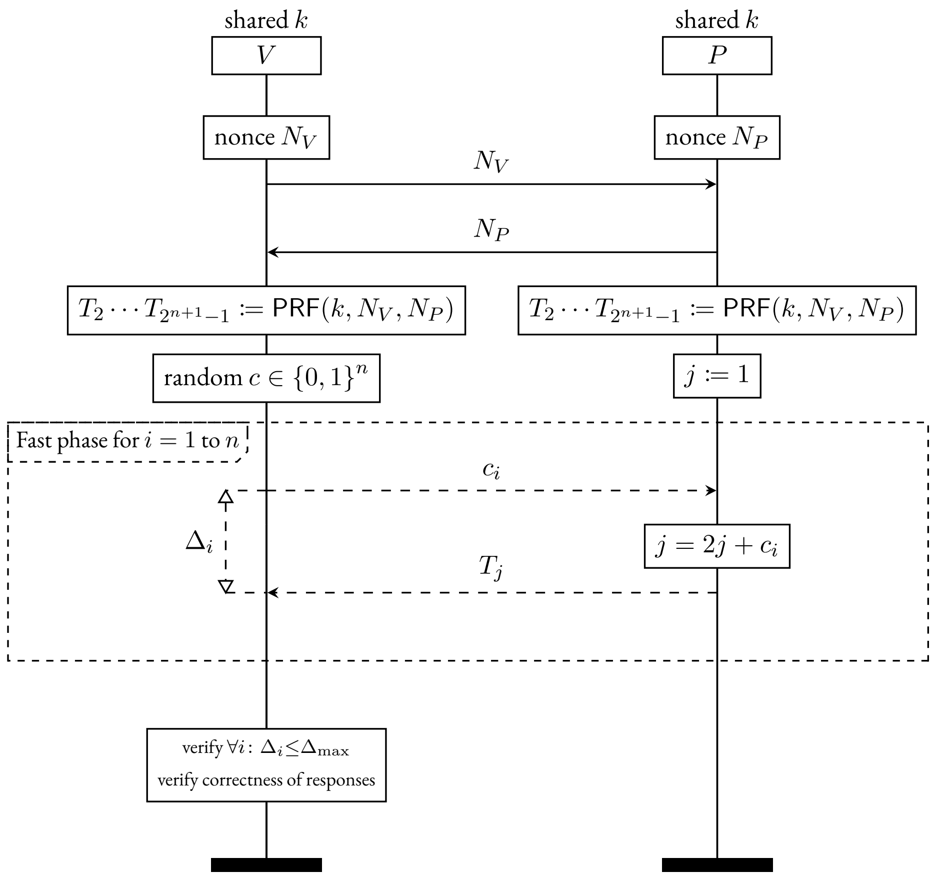
10.2.1 Specification

We use an automata-based representation of lookup-based protocols as introduced in [96]. An automaton, i.e., a state-labelled deterministic finite automaton (DFA), is a tuple $(\Sigma, \Gamma, Q, q_0, \delta, \ell)$, where Σ is a set of input symbols, Γ is a set of output symbols, Q is a set of states, $q_0 \in Q$ is the initial state, $\delta\colon Q \times \Sigma \to Q$ is a transition function, and $\ell\colon Q \to \Gamma$ is a labelling function. Given input and output symbol sets Σ and Γ, respectively, we use $\mathbf{U}_{\Sigma,\Gamma}$ to denote the universe of all DFAs over Σ and Γ.

Definition 38 (Lookup-based protocol [96])**.** A *lookup-based protocol* with input set Σ and output set Γ is a finite non-empty subset of $\mathbf{U}_{\Sigma,\Gamma}$.

Given an automaton $G = (\Sigma, \Gamma, Q, q_0, \delta, \ell)$ and a current state $q \in Q$, a lookup operation is regarded as a transition to a new state $q' = \delta(q, c)$ where $c \in \Sigma$ is a verifier's challenge. The corresponding response for such a challenge c_0 is the output symbol attached to the new state q', i.e., $\ell(q')$. This specification abstracts away from various protocol details that are present, for example, in the graphical descriptions given in Figure 10.1 and Figure 10.3, such as the precomputation phase and the round-trip-time measurements. These features of the protocol will be carefully considered when defining the execution model and security properties. For that, we need to introduce additional notation. For a sequence of input symbols $c = c_1 \cdots c_i \in \Sigma^i$:

- $\hat{\delta}(c) = \delta(\hat{\delta}(c_1 \ldots c_{i-1}), c_i)$ if $i > 1$, otherwise $\hat{\delta}(c_1) = \delta(q_0, c_1)$. In a nutshell, $\hat{\delta}(c)$ returns the state reached by the input sequence c.
- $\hat{\ell}(c) = \ell(\hat{\delta}(c))$, which is the output symbol attached to the state reached by the sequence c.
- $\Omega(c)$ is used to represent the sequence of labels attached to the states $\hat{\delta}(c_1), \hat{\delta}(c_1c_2), \ldots,$ $\hat{\delta}(c_1c_2 \cdots c_i)$ in that order, i.e., $\Omega(c) = r_1 \cdots r_i \in \Gamma^i$, where $r_j = \hat{\ell}(c_1 \cdots c_j)$ for $1 \leq j \leq i$.



shared k
V
shared k
P
nonce N_V
nonce N_P
N_V
N_P
$T_2 \cdots T_{2^{n+1}-1} := \mathsf{PRF}(k, N_V, N_P)$
$T_2 \cdots T_{2^{n+1}-1} := \mathsf{PRF}(k, N_V, N_P)$
random $c \in \{0,1\}^n$
$j := 1$
Fast phase for $i = 1$ to n
c_i
Δ_i
$j = 2j + c_i$
T_j
verify $\forall i\colon \Delta_i \leq \Delta_{\max}$
verify correctness of responses

Figure 10.3: The tree-based protocol by Avoine and Tchamkerten. The tree is encoded in the binary sequence $T_2 \cdots T_{2^{n+1}-1}$.

As a running example we model the tree-based protocol [20] (previously described in Chapter 2 and shown again in Figure 10.3 for the reader's advantage). In this protocol, each tree (representing the automaton) is the full binary tree of depth n, which contains $2^{n+1}-1$ nodes (representing the states of the automaton), where each node except the root is labelled randomly with either 0 or 1. This gives a total of $2^{2^{n+1}-2}$ different labelled trees.

Definition 39. The tree-based protocol is the set of automata $\{G_1, \ldots, G_{2^{2^{n+1}-2}}\}$ where each automaton $G_i = (\Sigma, \Gamma, Q, q_0, \delta, \ell)$ satisfies:

- $\Sigma = \Gamma = \{0, 1\}$,
- $Q = \{1, \ldots, 2^{n+1} - 1\}$ and $q_0 = 1$,
- For every $j \in Q$ and $c \in \Sigma$, $\delta(j, c) = 2j + c$
- For every $1 \leq j \leq 2^{n+1} - 2$, $\ell(j + 1)$ is the j-th least significant bit (right-most bit) of the binary representation of the integer number i, which is the index of the automaton G_i.

Observe that, for $n = 2$, the automaton G_{17} corresponds to the tree displayed in Figure 10.4. The labels follow from the binary representation of 17 (010001). Further, observe that the automaton G_{17} on input sequence $c = 01$ (represented in dashed arrows in Figure 10.4) gives:

- $\hat{\delta}(c) = \delta(\hat{\delta}(1), 1) = \delta(\delta(1, 0), 1) = \delta(2, 1) = 5$.
- $\hat{\ell}(c) = \ell(\hat{\delta}(01)) = 0$.
- $\Omega(c) = \hat{\ell}(0)\hat{\ell}(01) = 10$.

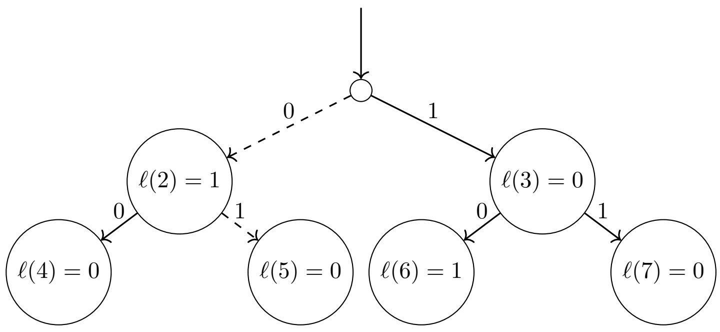


Figure 10.4: Example of a tree for an execution with two rounds. Dashed arrows indicate the path traversed on the challenges 0 and 1 (edge labels), resulting in the responses 1 and 0 (node labels), respectively.

Remark 3. Like in [96], we argue informally that Definition 39 is a correct formalization of the protocol depicted in Figure 10.3, as the theory does not allow for a formal proof of equivalence between the two specifications. Throughout the chapter we shall resort to a similar informal argumentation whenever we describe a protocol in graphical notation, as in Figure 10.3, and formalize it within the automata-based model, as in Definition 39.

Using the above formalism to describe lookup-based protocols, instead of, for example, Figure 10.3, makes it possible to formalize properties of the protocol, such as the following.

Proposition 3. Let P_{tree} be the tree-based protocol as in Definition 39. For any input sequence x, it holds that $\Pr\left[\hat{\ell}(x) = 0\right] = \Pr\left[\hat{\ell}(x) = 1\right] = \frac{1}{2}$, where the probability is taken over a random automaton sampled from P_{tree}. That is to say, if we look at $\hat{\ell}(x)$ as a random variable, then $\hat{\ell}(x)$ is uniformly distributed.

We will use this proposition later in our security proofs.

10.2.2 Execution

The model in [96] abstracts away the precomputation phase in lookup-based protocols by considering the following execution model.

```
$\text{Dist}^{\mathcal{A}}_{G,c_1\cdots c_n}(n)$
1 :  for $i = 1$ to $n$
2 :      $r_i \leftarrow \mathcal{A}.\mathsf{Resp}(G, c_1 \cdots c_{i-1})$
3 :  11return $r_1 \ldots r_n$
```

Figure 10.5: A security experiment for distance fraud.

Definition 40 (Execution model)**.** A *correct execution*, up to $n > 0$ rounds, of a lookup-based protocol P is a triple (G, C, R), where G is an automaton randomly selected from P, C is an input sequence randomly selected from Σ^n and R is an output sequence from Γ^n such that $R = \Omega(C)$.

The outcome of the precomputation phase is considered to be a random automaton $G \in_R P$ within the set of automata defining the lookup-based protocol P. The input sequence $C = c_1 \cdots c_n$ corresponds to the verifier's challenges, and the correct replies $R = r_1 \cdots r_n$ must satisfy that $\hat{\ell}(c_1) = r_1, \hat{\ell}(c_1c_2) = r_2, \ldots, \hat{\ell}(c_1 \cdots c_n) = r_n$.

Up to this point, the model has not established the role of the round-trip time of the exchanged messages and the location of prover and verifier. These features will be made part of the security properties by not allowing far-away participants to receive a verifier's challenge *in time*.

10.2.3 Security properties

In this chapter, we focus on the two most common and successful attack-strategies against lookup-based protocols: mafia fraud and distance fraud. Our focus on these two properties is because previous work [15, 16] have shown that terrorist fraud and distance-hijacking [45] pose no threat to lookup-based protocols.

Distance-fraud attack.

A distance-fraud attack consists of a corrupt prover trying to pass the protocol with a far-away verifier. To be successful, the corrupt prover cannot wait for the verifier's challenge to produce a response. Instead, the prover has to send its responses sufficiently in advance so that it passes the round-trip time condition.

Figure 10.5 shows the security experiment that defines distance fraud, where we use $\mathcal{A}.\mathsf{Resp}$ to denote the attacker's response algorithm (which in general may be randomized) and $\leftarrow\$$ to uniformly sample from a set. The experiment gives the adversary *white-box* access to an automaton G, representing the outcome of the precomputation phase of the protocol, and to the challenges of the verifier $c_1, \ldots, c_n$ in sequential order. At round i, the attacker knows all challenges sent by the verifier up to round $i - 1$, but not the challenge at the current round. This makes our model more conservative than previous models, such as [96], which assumes the corrupt prover is unaware of all the verifier's challenges.

Observe that the restriction of not letting the attacker receive the current challenge c_i, follows from the assumption that the corrupt prover (i.e. the attacker) is far away.

Definition 41 (Distance-fraud resistance)**.** The security of a lookup-based protocol P against a distance-fraud attacker $\mathcal{A}$, denoted $\mathsf{Adv}^{\mathrm{dist}}_{\mathcal{A},P}(n)$, is defined by,

$$\Pr_{G \leftarrow\!\$ P, c_1 \cdots c_n \leftarrow\!\$ \{0,1\}^n} \left[\mathrm{Dist}^{\mathcal{A}}_{G,c_1 \cdots c_n}(n) = \Omega(c_1 \cdots c_n) \right]$$

where the probability is over the coins (random source) of the adversary and the random choices of the automaton G and the verifier's challenges $c_1 \cdots c_n$.

In our security definition we do not let the attacker influence the choice of the automaton G. This may seem counter-intuitive, as the corrupt prover could, in practice, halt the precomputation phase various times looking for an automaton G with *good* properties for the attack. However, here we are being consistent with previous work on distance fraud [15, 16, 18, 76, 96, 108, 128], which considers an idealized precomputation phase whose outcome cannot be influenced by the attacker. We will informally discuss in our security analyses whether influencing the choice of G increases the adversary's probability of success.

Mafia fraud attack.

A mafia fraud attack is a man-in-the-middle attack in which the adversary, who is close to the verifier, interacts with a far-away prover to improve the odds on passing the time measurement phase with a legitimate verifier. This attack is fully defined by the choice of the adversary's challenges sent to the prover and the response function used by the adversary when trying to pass the time measurement phase. Therefore, we model the adversary $\mathcal{A}$ by two algorithms: $\mathcal{A}.\mathsf{Chall}$ and $\mathcal{A}.\mathsf{Resp}$. The former chooses the challenges the adversary sends to the prover, the latter chooses the responses the adversary provides to the verifier's challenges.

Figure 10.6 shows the security experiment we use to define mafia fraud. The experiment gives the adversary black-box access to an automaton G, representing the outcome of the precomputation phase of the protocol, and to the challenges of the verifier $c_1, \ldots, c_n$ in sequential order. At each round i, the adversary is let to choose a challenge y for the prover executing the protocol with automaton G by calling the algorithm $\mathcal{A}.\mathsf{Chall}$ on input all previous challenges sent by the attacker, denoted x, the correct responses from the prover to those challenges, denoted $\Omega(x)$, and all challenges sent by the verifier in previous rounds. Note that both x and y can be empty sequences. The only restriction in this step is that the adversary cannot query the prover having knowledge about c_i, the verifier's challenge in the current round, because the prover is far-away. Lastly, the adversary produces the response r_i at round i by seeing the challenge c_i and the pair $(x, \Omega(x))$ collected during the pre-ask step. The output of the experiment is the sequence of responses from the adversary.

Definition 42 (Mafia fraud resistance)**.** The security of a lookup-based protocol P against a mafia fraud attacker $\mathcal{A}$, denoted $\mathsf{Adv}^{\mathrm{mafia}}_{\mathcal{A},P}(n)$, is defined by:

$$\Pr_{G \leftarrow\!\$ P, c_1 \cdots c_n \leftarrow\!\$ \{0,1\}^n} [\mathrm{Mafia}^{\mathcal{A}}_{G,c_1 \cdots c_n}(n) = \Omega(c_1 \cdots c_n)]$$

where the probability is over the coins of the adversary and the random choices of the automaton G and the verifier's challenges $c_1 \cdots c_n$.

```
$\mathrm{Mafia}^{\mathcal{A}}_{G,c_1\cdots c_n}(n)$
1 :  $x = \varepsilon$
2 :  for $i = 1$ to $n$
3 :      $y \leftarrow \mathcal{A}.\mathsf{Chall}(x, \Omega(x), c_1 \cdots c_{i-1})$
4 :      $x = x \| y$
5 :      $r_i \leftarrow \mathcal{A}.\mathsf{Resp}(x, \Omega(x), c_1 \cdots c_i)$
6 :  return $r_1 \ldots r_n$
```

Figure 10.6: A security experiment for mafia fraud.

Unlike in distance fraud, where the attacker has white-box access to the prover, in mafia fraud both prover and verifier are assumed to be honest. Hence, the attacker has no influence on the prover's choice with respect to the automaton. At best, the adversary could desynchronize prover and verifier in such a way that the prover's automaton is different from the verifier's automaton. Such a strategy, however, does not give the attacker an advantage and, therefore, is not considered in Definition 42.

10.3 Impossibility result towards optimal protocols

The goal of this section is to prove that no lookup-based protocol exists that can simultaneously achieve optimal resistance to mafia fraud and distance fraud. We do so in three steps. First, we prove that $\frac{1}{2^n}\left(\frac{n}{2}+1\right)$ is a tight lower bound on the security of lookup-based protocols against mafia fraud. Second, we provide sufficient and necessary conditions for a lookup-based protocol to resist distance fraud with probability $\frac{1}{2^n}$, which is trivially optimal. Third, we prove that meeting the optimal bound with respect to one attack-strategy moves the protocol away from the optimal bound with respect to the other attack-strategy.

Theorem 7. Let P_{tree} be the tree-based protocol given in Definition 39. Then for all attackers $\mathcal{A}$ we have $\mathsf{Adv}^{\mathrm{mafia}}_{\mathcal{A},P_{tree}}(n) \leq \frac{1}{2^n}\left(\frac{n}{2}+1\right)$. Moreover, the bound is tight.

Proof. We start proving tightness of the bound. Consider the attacker $\mathcal{A}$ defined as follows, for every $i \in \{1, \ldots, n\}$, $c_1 \cdots c_i$, and an arbitrary sequence x (determined by $\mathcal{A}$),

$$\mathcal{A}.\mathsf{Resp}(x, \Omega(x), c_1 \cdots c_i) = \begin{cases} \hat{\ell}(x_1 \cdots x_i) & \text{if } x_1 \cdots x_i = c_1 \cdots c_i \\ \text{random} & \text{otherwise.} \end{cases}$$

This attacker succeeds with certainty in the i-th round when $x_1 \cdots x_i = c_1 \cdots c_i$ otherwise he succeeds with probability $\frac{1}{2}$. Let X be the random variable giving the round number i where the attacker does not correctly guess c_i. That is,

$$\Pr[X = i] = \Pr[x_1 \cdots x_{i-1} = c_1 \cdots c_{i-1} \wedge x_i \neq c_i] = \frac{1}{2^i}$$

where the last equality follows from $c_1 \cdots c_i$ being a random sequence independently generated. Then,

$$\Pr\left[\mathrm{Mafia}^{\mathcal{A}}_{G,c_1\cdots c_n}(n) = \Omega(c_1 \cdots c_n)\right] = \\ \sum_{i=1}^{n} \Pr\left[\mathrm{Mafia}^{\mathcal{A}}_{G,c_1\cdots c_n}(n) = \Omega(c_1 \cdots c_n) \,\middle|\, X = i\right] \Pr[X = i] + \frac{1}{2^n}$$

The last term in the sum $\frac{1}{2^n}$ accounts for the case where $x_1 \cdots x_n = c_1 \cdots c_n$.
Now, $\Pr\left[\mathrm{Mafia}^{\mathcal{A}}_{G,c_1\cdots c_n}(n) = \Omega(c_1 \cdots c_n) \,\middle|\, X = i\right] = \frac{1}{2^{n-i+1}}$. This gives,

$$\Pr\left[\mathrm{Mafia}^{\mathcal{A}}_{G,c_1\cdots c_n}(n) = \Omega(c_1 \cdots c_n)\right] = \sum_{i=1}^{n} \frac{1}{2^{n-i+1}} \cdot \frac{1}{2^i} + \frac{1}{2^n} = \frac{1}{2^n}\left(\frac{n}{2} + 1\right)$$

Observe that the above advantage is the same for every automaton G, which finishes this part of the proof. To prove that this is an optimal bound, consider the random variable $\hat{\ell}(c_1 \cdots c_i)$. According to Proposition 3, $\hat{\ell}(c_1 \cdots c_i)$ is uniform and, unless $\mathcal{A}.\mathsf{Chall}$ outputs $x_1 \cdots x_i = c_1 \cdots c_i$, then the adversary cannot guess $\hat{\ell}(c_1 \cdots c_i)$ with probability better than $\frac{1}{2}$. As we established earlier, the advantage in this case for the adversary is $\frac{1}{2^n}\left(\frac{n}{2} + 1\right)$. □

Once we have established that the tree-based protocol optimally resists mafia fraud with probability $\frac{1}{2^n}\left(\frac{n}{2} + 1\right)$, we move our attention to the analysis of distance fraud.

Theorem 8. A lookup-based protocol P satisfies $\mathsf{Adv}^{\mathrm{dist}}_{\mathcal{A},P}(n) \leq \frac{1}{2^n}$ for all attackers $\mathcal{A}$ if and only if for every $G \in P$ and every (possibly empty) sequence x of size lower than n it holds that $\hat{\ell}(x\|0) \neq \hat{\ell}(x\|1)$.

Proof. If for every $G \in P$ and every (possibly empty) sequence x of size lower than n it holds that $\hat{\ell}(x\|0) \neq \hat{\ell}(x\|1)$, then,

$$\Pr_{c \leftarrow\$ \{0,1\}}[\hat{\ell}(x\|c) = 0] = \Pr_{c \leftarrow\$ \{0,1\}}[\hat{\ell}(x\|c) = 1] = \frac{1}{2}$$

This means $r_i \leftarrow \mathcal{A}.\mathsf{Resp}(G, c_1 \cdots c_{i-1})$ has probability $\frac{1}{2}$ to be equal to $\hat{\ell}(c_1 \cdots c_{i-1}, c_i)$, because the adversary does not have c_i. This proves one direction of the double implication. To prove the other direction we proceed via contradiction.

Assume there exists $G \in P$ and a (possibly empty) sequence x such that $\hat{\ell}(x\|0) = \hat{\ell}(x\|1)$. We build the following adversary, for every G',

$$\mathcal{A}.\mathsf{Resp}(G', c_1 \cdots c_{i-1}) = \begin{cases} \hat{\ell}(x\|0) & \text{if } G = G' \text{ and } x = c_1 \cdots c_{i-1} \\ \text{random} & \text{otherwise.} \end{cases}$$

Note that such an adversary can be built because the adversary has white-box access to G'. Further note that we are considering syntactical equality between G and G', although the proof works with equality modulo isomorphism too.

If $G' = G$ and $x = c_1 \cdots c_{i-1}$, then the probability of success of this adversary is 1 at round i, because $\hat{\ell}(x\|0) = \hat{\ell}(x\|1)$. In any other situation, the adversary wins with probability $\frac{1}{2}$. Because the event $G' = G$ and $x = c_1 \cdots c_{i-1}$ has non-zero probability, it follows that the probability of success of the adversary is strictly larger than $\frac{1}{2^n}$. □

It is worth noting that the result above holds even if the adversary is allowed to choose the automaton G. The reason is that the necessary and sufficient condition stated in Theorem 8 has to be satisfied by all automata in the protocol, thereby giving the adversary the same success probability for all automata.

We now proceed to prove the main result of this section.

Theorem 9. For every lookup-based protocol P there exists a pair of adversaries $(\mathcal{A}_m, \mathcal{A}_d)$ such that either $\mathsf{Adv}^{\text{mafia}}_{\mathcal{A}_m,P}(n) \geq \frac{1}{2^n}(n+1)$ or $\mathsf{Adv}^{\text{dist}}_{\mathcal{A}_d,P}(n) > \frac{1}{2^n}$.

Proof. As established earlier, optimality in terms of distance fraud implies that, for every $G \in P$ and every (possibly empty) sequence x of size lower than n it holds that $\hat{\ell}(x\|0) \neq \hat{\ell}(x\|1)$. This allows us to build the following mafia fraud adversary, for every $i \in \{1, \ldots, n\}$, $c_1 \cdots c_i$, and sequence x,

$$\mathcal{A}.\mathsf{Resp}(x, \Omega(x), c_1 \cdots c_i) = \begin{cases} \hat{\ell}(x_1 \cdots x_i) & \text{if } x_1 \cdots x_i = c_1 \cdots c_i \\ 1 - \hat{\ell}(x_1 \cdots x_i) & \text{if } x_1 \cdots x_{i-1} = c_1 \cdots c_{i-1} \wedge x_i \neq c_i \\ \text{random} & \text{otherwise.} \end{cases}$$

Because $\hat{\ell}(c_1 \cdots c_{i-1}\|0) \neq \hat{\ell}(c_1 \cdots c_{i-1}\|1)$, it follows that the attacker succeeds with certainty if $x_1 \cdots x_{i-1} = c_1 \cdots c_{i-1}$, otherwise he succeeds with probability $\frac{1}{2}$. The rest of this proof is similar to the proof of Theorem 7. Let X be the random variable giving the round number i where the attacker does not correctly guess c_i. Then:

$$\Pr\Big[\text{Mafia}^{\mathcal{A}}_{G,c_1\cdots c_n}(n) = \Omega(c_1 \cdots c_n)\Big] = \sum_{i=1}^{n} \Pr\Big[\text{Mafia}^{\mathcal{A}}_{G,c_1\cdots c_n}(n) = \Omega(c_1 \cdots c_n) \,\Big|\, X = i\Big] \Pr[X = i] + \frac{1}{2^n}$$

From $\Pr\Big[\text{Mafia}^{\mathcal{A}}_{G,c_1\cdots c_n}(n) = \Omega(c_1 \cdots c_n) \,\Big|\, X = i\Big] = \frac{1}{2^{n-i}}$ we deduce that

$$\Pr\Big[\text{Mafia}^{\mathcal{A}}_{G,c_1\cdots c_n}(n) = \Omega(c_1 \cdots c_n)\Big] = \sum_{i=1}^{n} \frac{1}{2^{n-i}} \cdot \frac{1}{2^i} + \frac{1}{2^n} = \frac{1}{2^n}(n+1)$$

□

In addition to the impossibility result above, we obtain two main insights from the theoretical results in this section. First, there is a simple strategy to achieve optimality in terms of distance fraud resistance (see Theorem 8). Second, the variant of the tree-based protocol implementing such strategy seems to achieve a $\frac{1}{2^n}(n+1)$ security value against mafia fraud, which approximates the optimal value by a small constant factor lower than 2. Thus, our next step is to build a memory-efficient protocol with resistance to mafia fraud and distance fraud, respectively, equal to $\frac{1}{2^n}(n+1)$ and $\frac{1}{2^n}$.

10.4 A lookup-based protocol with nearly optimal security bounds

This section provides a protocol design that is memory efficient and nearly optimal in terms of mafia fraud and distance fraud.

10.4.1 Protocol specification in graphical notation

To specify our protocol in graphical notation we will use explicit lookup tables with domain the naturals and co-domain $\{0,1\}$. We write T_i instead of $T(i)$ to denote the output of T on index i, with i a natural number, and $T_i = b$ to denote the update of T with the value b on input the index i. Given a sequence of bits $b_1 \cdots b_n$ and an ordered set of keys $\Psi = \{i_1, \ldots, i_n\}$, we use $T_\Psi = b_1 \cdots b_n$ to denote the sequence of updates $T_{i_1} = b_1, \ldots, T_{i_n} = b_n$.

Our protocol (depicted in Figure 10.7) relies on a parameter d, which determines the maximum depth of the tree used throughout the protocol. Its most prominent feature is that it keeps a lookup table $T_2, \ldots, T_{2^{d+1}-1}$ of size $2^{d+1}-2$, which exponentially increases with d. All bits $T_2, T_4, \ldots, T_{2^{d+1}-2}$ are randomly sampled, while $T_i = 1 - T_{i-1}$ for every odd index $i \in [3, 2^{d+1}-1]$. During the fast phase, rounds are divided in blocks of exactly d rounds each. Throughout the protocol, the variable seed stores the index of the current node in the tree, which corresponds to the number whose binary representation is $c_1 \cdots c_i$, where i is the round number and $c_1 \cdots c_i$ the challenges sent by the verifier up to the current round. When the depth of seed is a multiple of d, which coincides with the condition $i \equiv 1 \mod d$, i.e. the first round in each block, the PRF is used to compute the $2^d - 1$ random bits necessary to run the protocol for the next d rounds. These bits will be assigned to the left child of each non-leaf node in the subtree of depth d rooted in the node seed. The right child will get the opposite value. Inside each block, the variable j represents the relative index of the current node with respect to the root node of the subtree rooted in seed.

Observe that, for $n = 4$, $d = 3$, the automaton used by our protocol corresponds to the tree displayed (partially) in Figure 10.8. The input sequence $c = 0101$ is represented in dashed arrows. The outputs of the pseudorandom function are: $\mathsf{PRF}(sk, 1) = T_2T_4T_6 = 110$ and $\mathsf{PRF}(sk, 5) = T_2T_4T_6 = 011$.

At first sight, the way bits are sampled in the protocol might look unnecessarily complex in comparison to the way bits are sampled in the tree-based protocol shown in Figure 10.3. While the tree-based protocol uses a single call to a pseudo-random function with output length $2^{n+1}-1$, ours uses several calls to a pseudo-random function with output length $2^{d+1}-1$, where d is a protocol parameter and divisor of n. Concretely, our protocol makes $\frac{n}{d}$ calls to the pseudo-random function. The advantage of our approach, however, is that we keep a lookup table of size $2^{d+1}-1$ as opposed to the lookup-table of size $2^{n+1}-1$ needed by the tree-based protocol. That is to say, for small values of d, say $d = \log_2 n$, our protocol requires little memory to be executed.

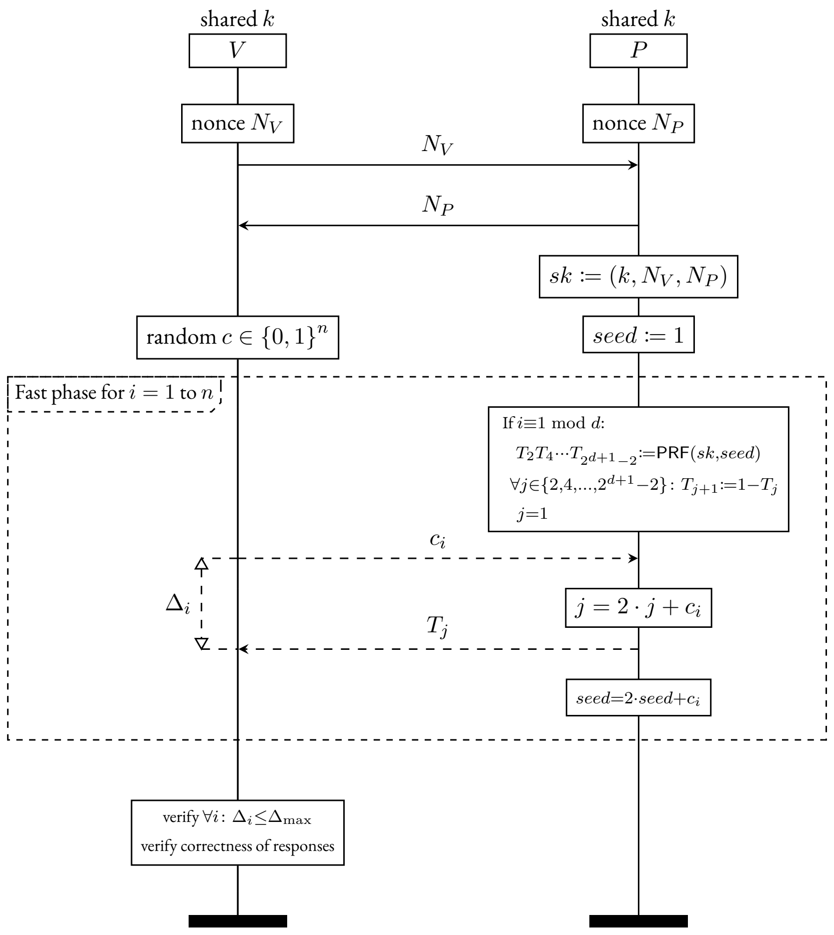

shared k
V
shared k
P
nonce N_V
nonce N_P
N_V
N_P
$sk := (k, N_V, N_P)$
random $c \in \{0,1\}^n$
$seed := 1$
Fast phase for $i = 1$ to n
If $i \equiv 1 \bmod d$:
$T_2T_4 \cdots T_{2^{d+1}-2} := \mathsf{PRF}(sk, seed)$
$\forall j \in \{2,4,\ldots,2^{d+1}-2\}: T_{j+1} := 1 - T_j$
$j = 1$
c_i
Δ_i
$j = 2 \cdot j + c_i$
T_j
$seed = 2 \cdot seed + c_i$
verify $\forall i: \Delta_i \leq \Delta_{\max}$
verify correctness of responses

Figure 10.7: Our nearly optimal protocol. The parameter d determines the output length of the pseudo-random function.

Next, we address whether this improvement in terms of memory size is traded-off by a decrease in resistance to mafia-fraud and distance-fraud attacks.

10.4.2 Security analysis

We analyse our protocol by reducing it to the protocol specified in Figure 10.9, which does not optimize memory, and it is simpler to analyse. Rather than calculating trees of depth $2^{d+1} - 1$ every d rounds, the protocol in Figure 10.9 calculates the entire tree prior the execution of the fast phase, just like the original tree-based protocol does. The protocol identifies a node by a sequence of bits $c_1 \cdots c_i$, denoting the path from the root to the node. Because lookup tables have domain the naturals, we use the auxiliary function $dec(.)$ to obtain the decimal representation of a binary sequence and identify a node $c_1 \cdots c_i$ with its decimal representation $dec(1||c_1 \cdots c_i)$ (see Figure 10.4 for an example). Based on this mapping from nodes in the tree to positive integers, the protocol samples bits from a pseudo-random function of fixed output length equal

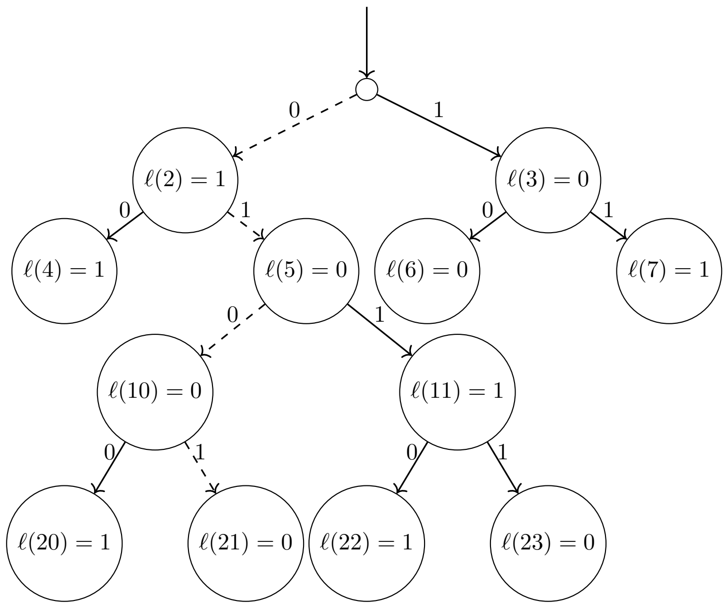


Figure 10.8: Example tree for an execution with 4 rounds. Dashed arrows indicate the path traversed on the challenges 0101 (edge labels), resulting in the responses 1000 (node labels), respectively.

to $2^{d+1} - 1$ bits as follows. For every node $c_1 \cdots c_i$ with i a multiple of d smaller than n, the protocol randomly sample half of the labels of the subtree with depth d rooted in $c_1 \cdots c_i$. Concretely, it samples the nodes whose identifiers are even numbers, assigning the opposite value to their siblings.

Lemma 32. For fixed nonces N_V and N_P, the protocols in Figure 10.9 and Figure 10.7 gives the same output on any sequence of challenges.

Proof. We proceed assuming that d is a divisor of n. Let $c_1 \cdots c_i$ be a sequence of challenges and $e = \lfloor \frac{i}{d} \rfloor$. Observe that the variable seed is identical in both protocols, in the sense that it stores the decimal representation of $1 \| c_1 \cdots c_{d \cdot e+1}$. This means that the sequence $T_2 \cdots T_{2^{d+1}-2}$ produced by $\mathsf{PRF}(sk, \mathsf{seed})$ in Figure 10.7 is equal to the sequence $T_{\Psi_x} = PRF(sk, \mathsf{seed})$ produced in Figure 10.9, where x is the binary representation of seed and

$$\Psi(x) = \{dec(1\|x\|y) | y \in \{0,1\}^i \wedge 1 \leq i \leq d \wedge y_i = 0\}$$

Although not explicitly mentioned earlier, we are assuming that $\Psi(x)$ is sorted in ascending order. This means that the position of j, as calculated in Figure 10.9, within $\Psi(x)$ is equal to the position of j, as calculated in Figure 10.7, within $T_2 \cdots T_{2^{d+1}-2}$, which implies that T_j stores the same value in both protocols. □

The lemma above proves that the protocols in Figure 10.7 and Figure 10.9 produce identical outputs after fixing the nonces N_V and N_P. This means that any adversary advantage when given black-box access to one protocol becomes an identical advantage when given black-box access to

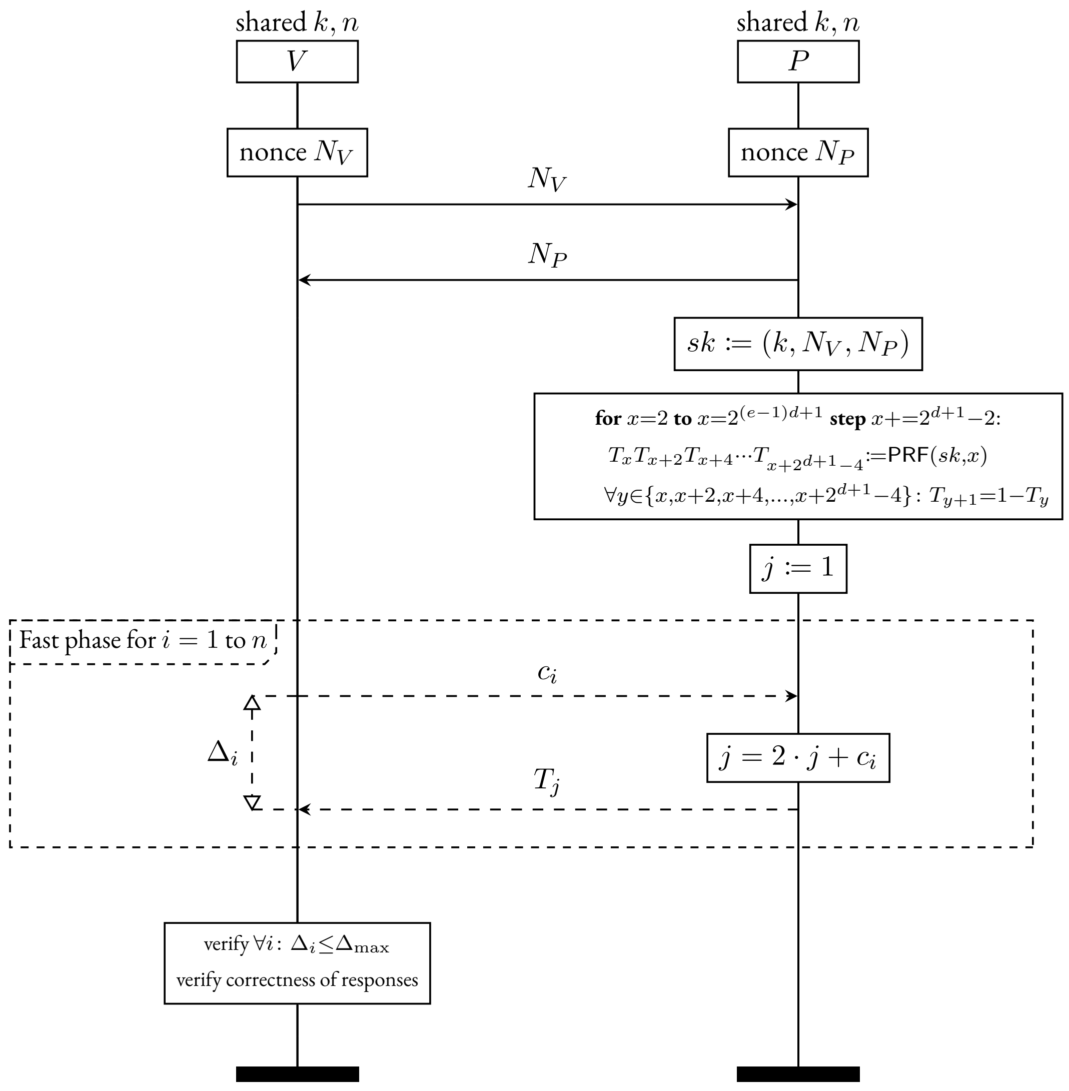

shared k, n
V
shared k, n
P
nonce N_V
nonce N_P
N_V
N_P
$sk := (k, N_V, N_P)$
for $x{=}2$ **to** $x{=}2^{(e-1)d+1}$ **step** $x{+}{=}2^{d+1}{-}2$:
$T_x T_{x+2} T_{x+4} \cdots T_{x+2^{d+1}-4} := \mathsf{PRF}(sk, x)$
$\forall y \in \{x, x+2, x+4, \ldots, x+2^{d+1}-4\}: T_{y+1} = 1 - T_y$
$j := 1$
Fast phase for $i = 1$ to n
c_i
Δ_i
$j = 2 \cdot j + c_i$
T_j
verify $\forall i: \Delta_i \leq \Delta_{\max}$
verify correctness of responses

Figure 10.9: An alternative specification of our protocol used to analyse its security properties.

the other protocol. Therefore, it is sufficient for us to analyse the adversary advantage over Figure 10.9.

To analyse the security properties of the protocol in Figure 10.9, we provide its formalization in Definition 43 using the automata-based model defined earlier. The formalization is identical to the tree-based protocol, except that it removes all automata that do not satisfy $\ell(s) \neq \ell(s+1)$ for every state s in $\{2, 4, \ldots, 2^{n+1} - 2\}$. That is to say, we make the labelling function guarantee that for every node in the tree the labels of its children are different. We argue that Definition 43 is a correct formalization of the protocol in Figure 10.9 by noting that all calls to the pseudo-random function in Figure 10.9 use a different seed. Therefore, the sequence $T_2T_4 \cdots T_{2^{n+1}-2}$ is indistinguishable from a random sequence, just like in the tree-based protocol.

Definition 43 (Specification of our protocol as a set of automata)**.** Our protocol is modelled by the set of automata $\{G_1, \ldots, G_{2^{2^{n+1}-2}}\}$ where each automaton $G_i = (\Sigma, \Gamma, Q, q_0, \delta, \ell)$ satisfies:

- $\Sigma = \Gamma = \{0, 1\}$,
- $Q = \{1, \ldots, 2^{n+1} - 1\}$ and $q_0 = 1$,
- For every $j \in \{1, \ldots, 2^n - 1\}$, and $c \in \Sigma$,
  - $\delta(j, c) = 2j + c$
  - $\ell(2 \cdot j)$ is the j-th least significant digit of the binary representation of the integer number i
  - $\ell(2 \cdot j + 1) = 1 - \ell(2 \cdot j)$

It follows directly from Theorem 8 that this protocol is optimal in terms of distance fraud. Next we prove that the protocol is nearly optimal in terms of mafia fraud.

Theorem 10. Let P_{opt} be the protocol given in Definition 43. Then for all attackers $\mathcal{A}$ we have $\mathsf{Adv}^{\text{mafia}}_{\mathcal{A},P_{opt}}(n) \leq \frac{1}{2^n}(n+1)$, and $\mathsf{Adv}^{\text{dist}}_{\mathcal{A},P_{opt}}(n) \leq \frac{1}{2^n}$.

Proof. It remains to prove the upper bound for mafia fraud attackers. The proof follows closely the one from Theorem 7. Consider an attacker $\mathcal{A}$. Let X be the random variable giving the round number i where the attacker does not correctly guess c_i. Then,

$$\begin{aligned}\Pr\Big[\text{Mafia}^{\mathcal{A}}_{G,c_1\cdots c_n}(n) = \Omega(c_1 \cdots c_n)\Big] &= \\ \sum_{i=1}^{n} \Pr\Big[\text{Mafia}^{\mathcal{A}}_{G,c_1\cdots c_n}(n) = \Omega(c_1 \cdots c_n) \,\Big|\, X = i\Big] \Pr[X = i] + \frac{1}{2^n}\end{aligned}$$

Now, $\Pr[X = i] = \frac{1}{2^i}$ because the challenges chosen independently and uniformly at random. The value $\Pr\Big[\text{Mafia}^{\mathcal{A}}_{G,c_1\cdots c_n}(n) = \Omega(c_1 \cdots c_n) \,\Big|\, X = i\Big]$ is upper bounded by $\frac{1}{2^{n-i}}$ because if the adversary guesses incorrectly c_i, then from the value of $\Omega(x_1 \cdots x_n)$ it gets no information about the labels of the nodes in the subtree determined by the path $c_1 \cdots c_{i+1}$, so the best it can do is randomly guess these values, and there are at least $n - i$ such values. We conclude:

$$\Pr\Big[\text{Mafia}^{\mathcal{A}}_{G,c_1\cdots c_n}(n) = \Omega(c_1 \cdots c_n)\Big] \leq \sum_{i=1}^{n} \frac{1}{2^{n-i}} \cdot \frac{1}{2^i} + \frac{1}{2^n} = \frac{1}{2^n}(n+1)$$

□

10.5 Evaluation

The goal of this section is to compare the security bounds of our protocol against previous lookup-based protocols. To make our comparison comprehensive, yet concise, we only consider the protocols regarded relevant by the decision-making methodology due to Avoine et al. [19], while adding the Hancke and Kuhn protocol (HK protocol for short) as a baseline. Concretely, we consider the protocols HK [76], Tree based [20], Modular [96] and Poulidor [130]. The security of these protocols in terms of mafia and distance fraud are taken from the framework available at `https://github.com/rolandotr/db_comparison`.

10.5.1 Evaluation setting

Because most distance-bounding protocols require a memory size that linearly grows with n, we shall restrict protocols to a linear memory size. This means that, for protocols whose memory requirement depends on protocol parameters, we set them such that the memory usage scales linearly with the number of rounds. In particular, for:

- The tree-based protocol: the depth of each tree is set to 6, which gives a memory size of approximately $21 \cdot n$ bits.
- The Modular protocol: the width of the graph is set to $u = 4$, which gives a memory size of $16 \cdot n$ bits.
- Our protocol: the depth of each tree is set to $\lfloor \log(n) \rfloor$, which gives a memory size of n bits.

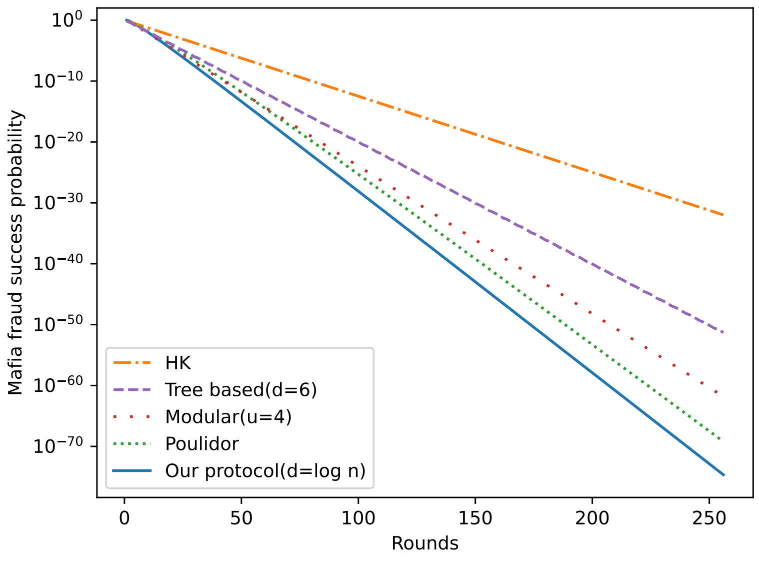


Figure 10.10: Mafia fraud success probability.

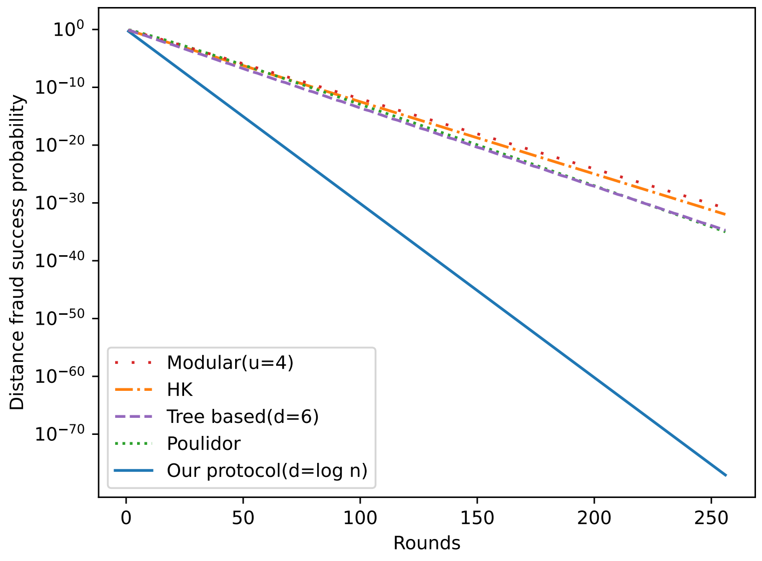


Figure 10.11: Distance fraud success probability.

10.5.2 Comparison results

In Figure 10.10 and Figure 10.11 we compare the protocols in terms of their resistance to mafia fraud and distance fraud, respectively. Our protocol clearly outperforms the rest in terms of both types of fraud, with Poulidor being its closest competitor. In terms of distance fraud, our protocol outperforms the others by a somewhat larger margin. This is not so surprising, as it was specifically designed to be optimal with respect to this fraud.

10.6 Conclusions

We presented a novel lookup-based distance-bounding protocol that provides (close to) optimal protection against distance fraud and mafia fraud attacks. We demonstrated the impossibility of achieving optimal protection against both types of fraud, and derived a tight lower bound for mafia fraud when distance fraud resistance is optimal. Furthermore, we conducted a comparative analysis of our protocol against previous lookup-based protocols in terms of their resistance to mafia fraud and distance fraud. The results show that our protocol outperforms others in mitigating both types of fraud.

Conclusions

In this thesis we addressed questions related to the security of protocols that depend on proximity and use round-trip-time measurements, such as key-exchange, memory-erasure and distance-bounding protocols. To provide security guarantees with mathematical rigour, we used formal analysis within symbolic and computational security models. In this regard, we reported on a number of flaws and shortcomings found by our analyses, and proposed novel memory-erasure and distance-bounding protocols with improved security.

Using the symbolic approach in the context of protocols with proximity assumptions, we identified and formalized the distant-attacker assumption. We also introduced a time-based security model where round-trip-time measurements and the location of agents is used to determine whether the neighbourhood of an agent is free of attackers. To enable computer-aided verification of protocols written in our specification language, we provided a reduction from the time-based to a causality-based model, which eliminates the notions of time and location, by defining proximity checks and the distant-attacker assumption as causal relations on the protocol events. We also introduced a class of security requirements for which the previous reduction is valid. Using Tamarin, we formally verified the security of five key-exchange protocols and two memory-erasure protocols, finding unreported vulnerabilities on three of them.

Using the computational approach in the context of software based memory-erasure protocols, we proposed protocols based on distance-bounding techniques, and proved them secure against distant attackers. Furthermore, we showed how to reduce the security analysis of this type of protocol to the case of a single round in the interactive phase. Our first protocol requires sending a random bitstring of the size of the prover's memory. The second one is a graph-based PoSE protocol with similar security guarantees that reduces the communication complexity during the initialization phase from linear to constant. The protocol asks the prover to label a depth-robust graph from a random seed. We introduced a class of graphs with depth-robust properties that can be labelled in-place using hash functions, a necessary condition to be usable in this context. The protocol was proven secure within a formal model, guaranteeing that all the prover's memory, except for a small part, is erased. Our third and final protocol is a lightweight version of the graph-based protocol based on a smaller graph that relaxes slightly the security requirements to achieve better performance.

In the search for tight security proofs for the graph-based PoSE protocols, we proved that a previous result on the relation between pebbling games and computational bounds is invalid [57], and showed a counterexample. We also found a correct variant that incurs only in a logarithmic loss [60]. We concluded that for applications where tightness is necessary (for example, memory-erasure), there is no satisfactory translation result without some loss in security.

Given the inexistence of freely available implementations of software-based memory-erasure protocols, we implemented the most prominent ones in a common framework. Our implementations provide necessary common ground to compare future protocols with respect to performance and security guarantees. We chose 7 protocols, each with several variants depending on the hash function used, and tested them on 3 modern IoT devices. Furthermore, we compared the security guarantees provided by each protocol, and contrasted them with their performance in a practical setting. Our results revealed that current memory-erasure protocols are practical, although erasing the full memory securely could take several minutes for the slower devices. Network speed might be faster than local computation, therefore reducing the communication complexity in the protocol is not always the best choice. For protocols where hash functions are used, this choice may influence dramatically the protocol performance and memory footprint. Finally, the most performant protocol might not be the best according to the asymptotic complexity analysis, as for small memory sizes the hidden constants may play a determinant role.

Although distance-bounding protocols have been widely studied, there are still some open questions regarding the existence of protocols with optimal security guarantees. We presented a new lookup-based distance-bounding protocol that provides (close to) optimal protection against distance fraud and mafia fraud attacks. We demonstrated the impossibility of achieving optimal protection against both types of fraud, and derived a tight lower bound for mafia fraud when distance fraud resistance is optimal. Furthermore, we conducted a comparative analysis of our protocol against previous lookup-based protocols in terms of their resistance to mafia fraud and distance fraud. The results shown that our protocol outperforms others in mitigating both types of fraud.

Future Work

During the development of this thesis we identified several issues for which we could not find solutions. Here we mention a few of them which might be of interest for future research.

In the symbolic analysis of protocols with round-trip-time measurements, the verifier checks whether all measurements were below a given threshold only after the protocol is finished and a security claim is made. If this decision is taken as soon as the fast phase finishes, our results are not applicable. In particular, in this case it is much harder to prove the equivalence between the model with time and the timeless model. The same has been the case with previous works on the symbolic analysis of distance-bounding protocols. Another improvement is generalizing our reduction proofs to a causality-based specification model that is a subset of the model supported by a state-of-the-art protocol verification tool, such as Tamarin, which would reduce the gap between theory and practice. Lastly, like previous verification frameworks for distance-bounding protocols, our methodology assumes that agents do not move. Dropping that assumption is of interest for both classes of protocols.

Our memory-erasure protocols cannot guarantee erasing the full memory of the device, because our proofs are not tight. In particular, for the general adversary and if the malware is small enough, our results do not guarantee the verifier will detect it, no matter how many rounds are executed during the fast phase. Nevertheless, we were not able not find any attack that would allow an adversary to save any non-constant amount of malware. Therefore, we believe that our protocols offer more security than what we can currently guarantee, and proving so is an interesting problem to solve. Such a result would find applications in related problems such as memory-hardness, proof of space and proof of work.

In our experiments on memory-erasure protocols we aimed to test the performance of several devices and protocols in a common setting. This made it unfeasible to erase all memory from the device, as erasure protocols aim by default. A more realistic experiment would be to focus on a single device, and try to erase as much memory as possible with every protocol. We believe multiple device specific optimizations might be needed to execute such experiment. Our experiments could also be enhanced by measuring the energy consumption of each protocol execution, a quantity that is important in some IoT applications.

Our new distance-bounding protocol has better security guarantees than previous works, but that might be at the expense of worse performance. Therefore, it would be interesting to implement it a practical setting to determine how much overhead in terms of time and energy it has with respect to previous proposals, in particular simpler protocols such as Hancke and Kuhn's. This would show for which applications the gain in security is worth the potential loss in performance.

Acronyms

AONT	all or nothing transform
BLBA	basic lower bound argument
CPA	chosen plaintext attack
DAG	directed acyclic graph
DDoS	distributed denial-of-service
DFA	deterministic finite automaton
HMAC	hash-based message authentication code
HTTP	Hypertext Transfer Protocol
IETF	Internet Engineering Task Force
IND-CCA2	indistinguishability under adaptive chosen ciphertext attack
IoT	Internet of Things
LTS	Labelled Transition System
MAC	message authentication code
MSC	message sequence chart
NFC	Near Field Communication
PoS	Proof of Space
PoSE	Proof of Secure Erasure
PoSE-DB	Proofs of Secure Erasure with Distance-Bounding
poset	partially ordered set
PoW	Proof of Work
PRF	Pseudorandom Function
ROM	Random Oracle Model
RTT	Time elapsed from sending a challenge and receiving a response
SAT	Boolean satisfiability problem
TPM	Trusted Platform Module

Bibliography

1. M. R. Albrecht, K. G. Paterson, and G. J. Watson. "Plaintext Recovery Attacks against SSH". In: *2009 30th IEEE Symposium on Security and Privacy*. IEEE, 2009, pp. 16–26.
2. M. A. Alturki, T. B. Kirigin, M. Kanovich, V. Nigam, A. Scedrov, and C. Talcott. "A Multiset Rewriting Model for Specifying and Verifying Timing Aspects of Security Protocols". In: *Foundations of Security, Protocols, and Equational Reasoning*. Springer, 2019, pp. 192–213.
3. J. Alwen, J. Blocki, and B. Harsha. "Practical Graphs for Optimal Side-Channel Resistant Memory-Hard Functions". In: *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. 2017, pp. 1001–1017.
4. J. Alwen, J. Blocki, and K. Pietrzak. "Depth-Robust Graphs and Their Cumulative Memory Complexity". In: *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2017, pp. 3–32.
5. J. Alwen, J. Blocki, and K. Pietrzak. "Sustained Space Complexity". In: *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2018, pp. 99–130.
6. J. Alwen, B. Chen, C. Kamath, V. Kolmogorov, K. Pietrzak, and S. Tessaro. "On the Complexity of Scrypt and Proofs of Space in the Parallel Random Oracle Model". In: *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2016, pp. 358–387.
7. J. Alwen, B. Chen, K. Pietrzak, L. Reyzin, and S. Tessaro. "Scrypt Is Maximally Memory-Hard". In: *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2017, pp. 33–62.
8. M. Ammar, B. Crispo, and G. Tsudik. "Simple: A Remote Attestation Approach for Resource Constrained Iot Devices". In: *2020 ACM/IEEE 11th International Conference on Cyber-Physical Systems (ICCPS)*. IEEE, 2020, pp. 247–258.
9. M. Ammar, W. Daniels, B. Crispo, and D. Hughes. "Speed: Secure Provable Erasure for Class-1 Iot Devices". In: *Eighth ACM Conference on Data and Application Security and Privacy*. 2018, pp. 111–118.
10. S. F. J. J. Ankergaard, E. Dushku, and N. Dragoni. "State-of-the-Art Software-Based Remote Attestation: Opportunities and Open Issues for Internet of Things". *Sensors* 21:5, 2021.

11. A. Armando, D. Basin, Y. Boichut, Y. Chevalier, L. Compagna, J. Cuellar, P. H. Drielsma, P. C. Heám, O. Kouchnarenko, J. Mantovani, S. Mödersheim, D. Von Oheimb, M. Rusinowitch, J. Santiago, M. Turuani, L. Viganò, and L. Vigneron. "The AVISPA Tool for the Automated Validation of Internet Security Protocols and Applications". In: *Computer Aided Verification*. Vol. 3576. Springer Berlin Heidelberg, Berlin, Heidelberg, 2005, pp. 281–285. ISBN: 978-3-540-27231-1 978-3-540-31686-2. DOI: `10.1007/11513988_27`.
12. F. Armknecht, A.-R. Sadeghi, S. Schulz, and C. Wachsmann. "A Security Framework for the Analysis and Design of Software Attestation". In: *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security*. 2013, pp. 1–12.
13. G. Ateniese, I. Bonacina, A. Faonio, and N. Galesi. "Proofs of Space: When Space Is of the Essence". In: *International Conference on Security and Cryptography for Networks*. Springer, 2014, pp. 538–557.
14. J.-P. Aumasson, S. Neves, Z. Wilcox-O'Hearn, and C. Winnerlein. "BLAKE2: Simpler, Smaller, Fast as MD5". In: *Applied Cryptography and Network Security*. Vol. 7954. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013, pp. 119–135. ISBN: 978-3-642-38979-5 978-3-642-38980-1. DOI: `10.1007/978-3-642-38980-1_8`.
15. G. Avoine, M. A. Bingöl, I. Boureanu, S. Čapkun, G. Hancke, S. Kardaş, C. H. Kim, C. Lauradoux, B. Martin, and J. Munilla. "Security of Distance-Bounding: A Survey". *ACM Computing Surveys (CSUR)* 51:5, 2018, pp. 1–33.
16. G. Avoine, M. A. Bingöl, S. Kardaş, C. Lauradoux, and B. Martin. "A Framework for Analyzing RFID Distance Bounding Protocols". *Journal of Computer Security* 19:2, 2011, pp. 289–317.
17. G. Avoine, I. Boureanu, D. Gérault, G. P. Hancke, P. Lafourcade, and C. Onete. "From Relay Attacks to Distance-Bounding Protocols". In: *Security of Ubiquitous Computing Systems*. Springer International Publishing, Cham, 2021, pp. 113–130. ISBN: 978-3-030-10590-7 978-3-030-10591-4. DOI: `10.1007/978-3-030-10591-4_7`.
18. G. Avoine, C. Floerkemeier, and B. Martin. "RFID Distance Bounding Multistate Enhancement". In: *Progress in Cryptology - INDOCRYPT 2009*. Vol. 5922. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009, pp. 290–307. ISBN: 978-3-642-10627-9 978-3-642-10628-6. DOI: `10.1007/978-3-642-10628-6_20`.
19. G. Avoine, S. Mauw, and R. Trujillo-Rasua. "Comparing Distance Bounding Protocols: A Critical Mission Supported by Decision Theory". *Computer Communications* 67, 2015, pp. 92–102.
20. G. Avoine and A. Tchamkerten. "An Efficient Distance Bounding RFID Authentication Protocol: Balancing False-Acceptance Rate and Memory Requirement". In: *Information Security*. Vol. 5735. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009, pp. 250–261. ISBN: 978-3-642-04473-1 978-3-642-04474-8. DOI: `10.1007/978-3-642-04474-8_21`.
21. D. Basin, S. Capkun, P. Schaller, and B. Schmidt. "Let's Get Physical: Models and Methods for Real-World Security Protocols". In: *International Conference on Theorem Proving in Higher Order Logics*. Springer, 2009, pp. 1–22.

22. M. Bellare and P. Rogaway. "Random Oracles Are Practical: A Paradigm for Designing Efficient Protocols". In: *Proceedings of the 1st ACM Conference on Computer and Communications Security*. 1993, pp. 62–73.

23. J. Benet. "Ipfs-Content Addressed, Versioned, P2p File System". 2014. arXiv: 1407.3561.

24. S. Bengio, G. Brassard, Y. G. Desmedt, C. Goutier, and J.-J. Quisquater. "Secure Implementation of Identification Systems". *Journal of Cryptology* 4:3, 1991, pp. 175–183. ISSN: 0933-2790, 1432-1378. DOI: 10.1007/BF00196726.

25. T. Beth and Y. Desmedt. *Identification Tokens—or: Solving the Chess Grandmaster Problem*. Springer, 1991.

26. K. Bhargavan, A. D. Lavaud, C. Fournet, A. Pironti, and P. Y. Strub. "Triple Handshakes and Cookie Cutters: Breaking and Fixing Authentication over TLS". In: *2014 IEEE Symposium on Security and Privacy*. IEEE, 2014, pp. 98–113.

27. B. Blanchet. "An Efficient Cryptographic Protocol Verifier Based on Prolog Rules." In: *Csfw*. Vol. 1. 2001, pp. 82–96.

28. B. Blanchet. "Modeling and Verifying Security Protocols with the Applied Pi Calculus and ProVerif". *Foundations and Trends® in Privacy and Security* 1:1-2, 2016, pp. 1–135.

29. J. Blocki and M. Cinkoske. "A New Connection Between Node and Edge Depth Robust Graphs". In: *12th Innovations in Theoretical Computer Science Conference (ITCS 2021)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2021.

30. C. Bormann, M. Ersue, and A. Keranen. "RFC 7228: Terminology for Constrained-Node Networks". *IETF Request For Comments*, 2014.

31. I. Boureanu, T. Chothia, A. Debant, and S. Delaune. "Security Analysis and Implementation of Relay-Resistant Contactless Payments". In: *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*. 2020, pp. 879–898.

32. I. Boureanu, A. Mitrokotsa, and S. Vaudenay. "Practical and Provably Secure Distance-Bounding". *Journal of Computer Security* 23:2, 2015, pp. 229–257.

33. S. Brands and D. Chaum. "Distance-Bounding Protocols". In: *Workshop on the Theory and Application of Cryptographic Techniques*. Springer, 1993, pp. 344–359.

34. S. Bursuc, R. Gil-Pons, S. Mauw, and R. Trujillo-Rasua. "Software-Based Memory Erasure with Relaxed Isolation Requirements". In: *Proc. 37th IEEE Computer Security Foundations Symposium (CSF'24)*. 2024, to appear.

35. S. Čapkun, L. Buttyán, and J.-P. Hubaux. "SECTOR: Secure Tracking of Node Encounters in Multi-Hop Wireless Networks". In: *Proceedings of the 1st ACM Workshop on Security of Ad Hoc and Sensor Networks*. 2003, pp. 21–32.

36. C. Castelluccia, A. Francillon, D. Perito, and C. Soriente. "On the Difficulty of Software-Based Attestation of Embedded Devices". In: *Proceedings of the 16th ACM Conference on Computer and Communications Security*. 2009, pp. 400–409.

37. K.-K. R. Choo, C. Boyd, and Y. Hitchcock. "Errors in Computational Complexity Proofs for Protocols". In: *Advances in Cryptology - ASIACRYPT 2005*. Vol. 3788. Springer Berlin Heidelberg, Berlin, Heidelberg, 2005, pp. 624–643. ISBN: 978-3-540-30684-9 978-3-540-32267-2. DOI: `10.1007/11593447_34`.

38. T. Chothia, J. De Ruiter, and B. Smyth. "Modelling and Analysis of a Hierarchy of Distance Bounding Attacks". In: *27th ${$USENIX$}$ Security Symposium (${$USENIX$}$ Security 18)*. 2018, pp. 1563–1580.

39. B. Cohen. "Incentives Build Robustness in BitTorrent". In: *Workshop on Economics of Peer-to-Peer Systems*. Vol. 6. Berkeley, CA, USA, 2003, pp. 68–72.

40. G. Cohen, I. Honkala, S. Litsyn, and A. Lobstein. *Covering Codes*. Elsevier, 1997.

41. K. Cohn-Gordon, C. Cremers, and L. Garratt. "On Post-Compromise Security". In: *2016 IEEE 29th Computer Security Foundations Symposium (CSF)*. IEEE, 2016, pp. 164–178.

42. M. Conti, D. Donadel, R. Poovendran, and F. Turrin. "Evexchange: A Relay Attack on Electric Vehicle Charging System". In: *Computer Security–ESORICS 2022: 27th European Symposium on Research in Computer Security, Copenhagen, Denmark, September 26–30, 2022, Proceedings, Part I*. Springer, 2022, pp. 488–508.

43. R. Corin, S. Etalle, P. H. Hartel, and A. Mader. "Timed Model Checking of Security Protocols". In: *Proceedings of the 2004 ACM Workshop on Formal Methods in Security Engineering*. 2004, pp. 23–32.

44. C. Cremers and D. Jackson. "Prime, Order Please! Revisiting Small Subgroup and Invalid Curve Attacks on Protocols Using Diffie-Hellman". In: *2019 IEEE 32nd Computer Security Foundations Symposium (CSF)*. IEEE, 2019, pp. 78–7815.

45. C. Cremers and S. Mauw. *Operational Semantics and Verification of Security Protocols*. Information Security and Cryptography. Springer, Berlin, 2012. 172 pp. ISBN: 978-3-540-78635-1 978-3-540-78636-8.

46. C. Cremers, K. B. Rasmussen, B. Schmidt, and S. Capkun. "Distance Hijacking Attacks on Distance Bounding Protocols". In: *2012 IEEE Symposium on Security and Privacy*. IEEE, 2012, pp. 113–127.

47. C. J. Cremers. "The Scyther Tool: Verification, Falsification, and Analysis of Security Protocols". In: *International Conference on Computer Aided Verification*. Springer, 2008, pp. 414–418.

48. A. De, L. Trevisan, and M. Tulsiani. "Time Space Tradeoffs for Attacks against One-Way Functions and PRGs". In: *Annual Cryptology Conference*. Springer, 2010, pp. 649–665.

49. A. Debant and S. Delaune. "Symbolic Verification of Distance Bounding Protocols". In: *International Conference on Principles of Security and Trust*. Springer, 2019, pp. 149–174.

50. A. Debant, S. Delaune, and C. Wiedling. "A Symbolic Framework to Analyse Physical Proximity in Security Protocols". In: *38th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2018)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2018.

51. A. Debant, S. Delaune, and C. Wiedling. “Proving Physical Proximity Using Symbolic Models”, 2018.
52. A. Debant, S. Delaune, and C. Wiedling. “So Near and Yet So Far – Symbolic Verification of Distance-Bounding Protocols”. *ACM Transactions on Privacy and Security* 25:2, 31, 2022, pp. 1–39. ISSN: 2471-2566, 2471-2574. DOI: `10.1145/3501402`.
53. Y. Desmedt, C. Goutier, and S. Bengio. “Special Uses and Abuses of the Fiat-Shamir Passport Protocol (Extended Abstract)”. In: *Advances in Cryptology — CRYPTO ’87*. Vol. 293. Springer Berlin Heidelberg, Berlin, Heidelberg, 1988, pp. 21–39. ISBN: 978-3-540-18796-7 978-3-540-48184-3. DOI: `10.1007/3-540-48184-2_3`.
54. C. Dobraunig, M. Eichlseder, F. Mendel, and M. Schläffer. “Ascon v1.2: Lightweight Authenticated Encryption and Hashing”. *Journal of Cryptology* 34:3, 2021, p. 33. ISSN: 0933-2790, 1432-1378. DOI: `10.1007/s00145-021-09398-9`.
55. D. Dolev and A. Yao. “On the Security of Public Key Protocols”. *IEEE Transactions on information theory* 29:2, 1983, pp. 198–208.
56. J. A. Donenfeld. “Wireguard: Next Generation Kernel Network Tunnel.” In: *NDSS*. 2017, pp. 1–12.
57. C. Dwork, M. Naor, and H. Wee. “Pebbling and Proofs of Work”. In: *Annual International Cryptology Conference*. Springer, 2005, pp. 37–54.
58. S. Dziembowski, S. Faust, V. Kolmogorov, and K. Pietrzak. “Proofs of Space”. In: *Annual Cryptology Conference*. Springer, 2015, pp. 585–605.
59. S. Dziembowski, T. Kazana, and D. Wichs. “One-Time Computable and Uncomputable Functions”. *Cryptology ePrint Archive*, 2010.
60. S. Dziembowski, T. Kazana, and D. Wichs. “One-Time Computable Self-Erasing Functions”. In: *Theory of Cryptography Conference*. 2011, pp. 125–143.
61. D. Eastlake and T. Hansen. *US Secure Hash Algorithms (SHA and HMAC-SHA)*. RFC4634. RFC Editor, 2006, RFC4634. DOI: `10.17487/rfc4634`.
62. E. M. V. EMVCo. “Contactless Specifications for Payment Systems”. *Book C-2, Kernel* 2, 2021.
63. P. Erdős, R. L. Graham, and E. Szemerédi. “On Sparse Graphs with Dense Long Paths”. *Comp. and Math. with Appl* 1, 1975, pp. 145–161.
64. A. Francillon, B. Danev, and S. Capkun. “Relay Attacks on Passive Keyless Entry and Start Systems in Modern Cars”. In: *Proceedings of the Network and Distributed System Security Symposium (NDSS)*. Eidgenössische Technische Hochschule Zürich, Department of Computer Science, 2011.
65. L. Francis, G. Hancke, K. Mayes, and K. Markantonakis. “Practical NFC Peer-to-Peer Relay Attack Using Mobile Phones”. In: *Radio Frequency Identification: Security and Privacy Issues*. Vol. 6370. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010, pp. 35–49. ISBN: 978-3-642-16821-5 978-3-642-16822-2. DOI: `10.1007/978-3-642-16822-2_4`.

66. L. Francis, G. Hancke, K. Mayes, and K. Markantonakis. “Practical Relay Attack on Contactless Transactions by Using NFC Mobile Phones”. *Cryptology ePrint Archive*, 2011.

67. R. Gil Pons, R. J. Horne, S. Mauw, R. Trujillo-Rasua, and A. Tiu. “Is Eve Nearby? Analysing Protocols under the Distant-Attacker Assumption”. In: *IEEE Computer Security Foundations Symposium, August 7-10, 2022, Haifa, Israel*. 2022. DOI: 10.1109/CSF54842.2022.9919655.

68. R. Gil-Pons, S. Mauw, and R. Trujillo-Rasua. “On the Optimal Resistance against Mafia and Distance Fraud in Distance-Bounding Protocols”. *Computer Communications* 210, 2023, pp. 69–78.

69. J. R. Gilbert, T. Lengauer, and R. E. Tarjan. “The Pebbling Problem Is Complete in Polynomial Space”. In: *Proceedings of the Eleventh Annual ACM Symposium on Theory of Computing*. 1979, pp. 237–248.

70. J. R. Gilbert and R. E. Tarjan. *Variations of a Pebble Game on Graphs.* Standford University, 1978.

71. J. A. Goguen and J. Meseguer. “Order-Sorted Algebra I: Equational Deduction for Multiple Inheritance, Overloading, Exceptions and Partial Operations”. *Theoretical Computer Science* 105:2, 1992, pp. 217–273.

72. S. Goldwasser and S. Micali. “Probabilistic Encryption”. *Journal of Computer and System Sciences* 28, 1984, pp. 270–299.

73. S. Goldwasser, S. Micali, and R. L. Rivest. “A Digital Signature Scheme Secure against Adaptive Chosen-Message Attacks”. *SIAM Journal on computing* 17:2, 1988, pp. 281–308.

74. P. Gutmann. “Secure Deletion of Data from Magnetic and Solid-State Memory”. In: *Proceedings of the Sixth USENIX Security Symposium, San Jose, CA*. Vol. 14. 1996, pp. 77–89.

75. G. P. Hancke. “Practical Attacks on Proximity Identification Systems”. In: *2006 IEEE Symposium on Security and Privacy (S&P’06)*. IEEE, 2006, 6–pp.

76. G. P. Hancke and M. G. Kuhn. “An RFID Distance Bounding Protocol”. In: *First International Conference on Security and Privacy for Emerging Areas in Communications Networks (SECURECOMM’05)*. IEEE, 2005, pp. 67–73.

77. J. Hopcroft, W. Paul, and L. Valiant. “On Time Versus Space”. *Journal of the ACM* 24:2, 1977, pp. 332–337. ISSN: 0004-5411, 1557-735X. DOI: 10.1145/322003.322015.

78. G. Jakubowska and W. Penczek. “Modelling and Checking Timed Authentication of Security Protocols”. *Fundamenta Informaticae* 79:3-4, 2007, pp. 363–378.

79. G. O. Karame and W. Li. “Secure Erasure and Code Update in Legacy Sensors”. In: *International Conference on Trust and Trustworthy Computing*. Springer, 2015, pp. 283–299.

80. S. Kardaş, M. S. Kiraz, M. A. Bingöl, and H. Demirci. “A Novel RFID Distance Bounding Protocol Based on Physically Unclonable Functions”. In: *RFID. Security and Privacy*. Vol. 7055. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012, pp. 78–93. ISBN: 978-3-642-25285-3 978-3-642-25286-0. DOI: 10.1007/978-3-642-25286-0_6.

81. N. P. Karvelas and A. Kiayias. “Efficient Proofs of Secure Erasure”. In: *International Conference on Security and Cryptography for Networks*. Springer, 2014, pp. 520–537.

82. Z. Kfir and A. Wool. “Picking Virtual Pockets Using Relay Attacks on Contactless Smartcard”. In: *First International Conference on Security and Privacy for Emerging Areas in Communications Networks (SECURECOMM’05)*. IEEE, 2005, pp. 47–58.

83. C. Kil, E. C. Sezer, A. M. Azab, P. Ning, and X. Zhang. “Remote Attestation to Dynamic System Properties: Towards Providing Complete System Integrity Evidence”. In: *2009 IEEE/IFIP International Conference on Dependable Systems & Networks*. IEEE, 2009, pp. 115–124.

84. C. H. Kim and G. Avoine. “RFID Distance Bounding Protocols with Mixed Challenges”. *IEEE Transactions on Wireless Communications* 10:5, 2011, pp. 1618–1626.

85. C. H. Kim, G. Avoine, F. Koeune, F.-X. Standaert, and O. Pereira. “The Swiss-Knife RFID Distance Bounding Protocol”. In: *International Conference on Information Security and Cryptology*. Springer, 2008, pp. 98–115.

86. D. Konig. “Graphs and Matrices”. *Matematikai és Fizikai Lapok* 38, 1931, pp. 116–119.

87. B. Kuang, A. Fu, W. Susilo, S. Yu, and Y. Gao. “A Survey of Remote Attestation in Internet of Things: Attacks, Countermeasures, and Prospects”. *Computers & Security* 112, 2022, p. 102498.

88. T. Lengauer and R. E. Tarjan. “Asymptotically Tight Bounds on Time-Space Trade-Offs in a Pebble Game”. *Journal of the ACM (JACM)* 29:4, 1982, pp. 1087–1130.

89. G. Lowe. “A Hierarchy of Authentication Specifications”. In: *Proceedings 10th Computer Security Foundations Workshop*. 10th Computer Security Foundations Workshop. IEEE Comput. Soc. Press, Rockport, MA, USA, 1997, pp. 31–43. ISBN: 978-0-8186-7990-2. DOI: `10.1109/CSFW.1997.596782`.

90. G. Lowe. “An Attack on the Needham- Schroeder Public- Key Authentication Protocol”. *Information processing letters* 56:3, 1995.

91. G. Lowe. “Breaking and Fixing the Needham-Schroeder Public-Key Protocol Using FDR”. In: *Tools and Algorithms for the Construction and Analysis of Systems*. Vol. 1055. Springer Berlin Heidelberg, Berlin, Heidelberg, 1996, pp. 147–166. ISBN: 978-3-540-61042-7 978-3-540-49874-2. DOI: `10.1007/3-540-61042-1_43`.

92. M. Mahmoody, T. Moran, and S. Vadhan. “Publicly Verifiable Proofs of Sequential Work”. In: *Proceedings of the 4th Conference on Innovations in Theoretical Computer Science*. 2013, pp. 373–388.

93. D. Malinowski and K. Żebrowski. “Disproving the Conjectures from “On the Complexity of Scrypt and Proofs of Space in the Parallel Random Oracle Model””. In: *International Conference on Information Theoretic Security*. Springer, 2017, pp. 26–38.

94. U. M. Maurer. “The Role of Information Theory in Cryptography”. In: *Fourth IMA Conference on Cryptography and Coding*. Citeseer, 1993, pp. 49–71.

95. S. Mauw, Z. Smith, J. Toro-Pozo, and R. Trujillo-Rasua. "Distance-Bounding Protocols: Verification Without Time and Location". In: *2018 IEEE Symposium on Security and Privacy (SP)*. 2018 IEEE Symposium on Security and Privacy (SP). 2018, pp. 549–566. DOI: 10.1109/SP.2018.00001.

96. S. Mauw, J. Toro-Pozo, and R. Trujillo-Rasua. "A Class of Precomputation-Based Distance-Bounding Protocols". In: *2016 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 2016, pp. 97–111.

97. V. Mavroeidis, K. Vishi, M. D. Zych, and A. Jøsang. "The Impact of Quantum Computing on Present Cryptography". 2018. arXiv: 1804.00200.

98. C. Meadows. "Formal Methods for Cryptographic Protocol Analysis: Emerging Issues and Trends". *IEEE journal on selected areas in communications* 21:1, 2003, pp. 44–54.

99. C. Meadows, R. Poovendran, D. Pavlovic, L. Chang, and P. Syverson. "Distance Bounding Protocols: Authentication Logic Analysis and Collusion Attacks". In: *Secure Localization and Time Synchronization for Wireless Sensor and Ad Hoc Networks*. Springer, 2007, pp. 279–298.

100. S. Meier, B. Schmidt, C. Cremers, and D. Basin. "The TAMARIN Prover for the Symbolic Analysis of Security Protocols". In: *International Conference on Computer Aided Verification*. Springer, 2013, pp. 696–701.

101. F. Meneghello, M. Calore, D. Zucchetto, M. Polese, and A. Zanella. "IoT: Internet of Threats? A Survey of Practical Security Vulnerabilities in Real IoT Devices". *IEEE Internet of Things Journal* 6:5, 2019, pp. 8182–8201.

102. J. Mitchell, A. Scedrov, N. Durgin, and P. Lincoln. "Undecidability of Bounded Security Protocols". In: *Workshop on Formal Methods and Security Protocols*. 1999.

103. J. Munilla and A. Peinado. "Distance Bounding Protocols for RFID Enhanced by Using Void-Challenges and Analysis in Noisy Channels". *Wireless Communications and Mobile Computing* 8:9, 2008, pp. 1227–1232. ISSN: 15308669, 15308677. DOI: 10.1002/wcm.590.

104. S. Nakamoto. "Bitcoin: A Peer-to-Peer Electronic Cash System". *Decentralized business review*, 2008, p. 21260.

105. R. M. Needham and M. D. Schroeder. "Using Encryption for Authentication in Large Networks of Computers". *Communications of the ACM* 21:12, 1978, pp. 993–999.

106. L. H. Newman. "An Elaborate Hack Shows How Much Damage IoT Bugs Can Do", 2010.

107. J. Nordström. "New Wine into Old Wineskins: A Survey of Some Pebbling Classics with Supplemental Results". *Available on line at http://people. csail. mit. edu/jakobn/research*, 2009.

108. A. Özhan Gürel, A. Arslan, and M. Akgün. "Non-Uniform Stepping Approach to RFID Distance Bounding Problem". In: *Data Privacy Management and Autonomous Spontaneous Security*. Vol. 6514. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011, pp. 64–78. ISBN: 978-3-642-19347-7 978-3-642-19348-4. DOI: 10.1007/978-3-642-19348-4_6.

109. B. Parno, J. M. McCune, and A. Perrig. "Bootstrapping Trust in Commodity Computers". In: *2010 IEEE Symposium on Security and Privacy*. IEEE, 2010, pp. 414–429.

110. W. J. Paul, R. E. Tarjan, and J. R. Celoni. "Space Bounds for a Game on Graphs". *Mathematical systems theory* 10:1, 1976, pp. 239–251.

111. D. Perito and G. Tsudik. "Secure Code Update for Embedded Devices via Proofs of Secure Erasure". In: *European Symposium on Research in Computer Security*. Springer, 2010, pp. 643–662.

112. T. Perrin and M. Marlinspike. "The Double Ratchet Algorithm". *GitHub wiki*, 2016, p. 10.

113. F. Pfeiffer, K. Finkenzeller, and E. Biebl. "Theoretical Limits of ISO/IEC 14443 Type A RFID Eavesdropping Attacks". In: *Smart SysTech 2012; European Conference on Smart Objects, Systems and Technologies*. VDE, 2012, pp. 1–9.

114. K. Pietrzak. "Proofs of Catalytic Space". *Cryptology ePrint Archive*, 2018.

115. A.-I. Radu, T. Chothia, C. J. Newton, I. Boureanu, and L. Chen. "Practical EMV Relay Protection". In: *2022 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2022, pp. 1737–1756.

116. K. B. Rasmussen and S. Capkun. "Realization of RF Distance Bounding." In: *USENIX Security Symposium*. 2010, pp. 389–402.

117. K. B. Rasmussen, C. Castelluccia, T. S. Heydt-Benjamin, and S. Capkun. "Proximity-Based Access Control for Implantable Medical Devices". In: *Proceedings of the 16th ACM Conference on Computer and Communications Security*. 2009, pp. 410–419.

118. E. Rescorla. *The Transport Layer Security (TLS) Protocol Version 1.3*. 2018.

119. J. E. Savage. *Models of Computation*. Vol. 136. Addison-Wesley Reading, MA, 1998.

120. G. Schnitger. "On Depth-Reduction and Grates". In: *24th Annual Symposium on Foundations of Computer Science (Sfcs 1983)*. IEEE, 1983, pp. 323–328.

121. A. Seshadri, A. Perrig, L. Van Doorn, and P. Khosla. "SWATT: Software-based Attestation for Embedded Devices". In: *IEEE Symposium on Security and Privacy, 2004. Proceedings. 2004*. IEEE, 2004, pp. 272–282.

122. D. R. Simon. "Finding Collisions on a One-Way Street: Can Secure Hash Functions Be Based on General Assumptions?" In: *Advances in Cryptology—EUROCRYPT'98: International Conference on the Theory and Application of Cryptographic Techniques Espoo, Finland, May 31–June 4, 1998 Proceedings 17*. Springer, 1998, pp. 334–345.

123. D. Singelee and B. Preneel. "Key Establishment Using Secure Distance Bounding Protocols". In: *2007 Fourth Annual International Conference on Mobile and Ubiquitous Systems: Networking & Services (MobiQuitous)*. IEEE, 2007, pp. 1–6.

124. D.-Z. Sun, Y. Mu, and W. Susilo. "Man-in-the-Middle Attacks on Secure Simple Pairing in Bluetooth Standard V5. 0 and Its Countermeasure". *Personal and Ubiquitous Computing* 22:1, 2018, pp. 55–67.

125. S. Szymoniak, O. Siedlecka-Lamch, and M. l. Kurkowski. "Timed Analysis of Security Protocols". In: *Information Systems Architecture and Technology: Proceedings of 37th International Conference on Information Systems Architecture and Technology–ISAT 2016–Part II*. Springer, 2017, pp. 53–63.

126. J. Talbot, D. Welsh, and D. J. A. Welsh. *Complexity and Cryptography: An Introduction*. Vol. 13. Cambridge University Press, 2006.

127. P. Thueringer, H. De Jong, B. Murray, H. Neumann, P. Hubmer, and S. Stern. "Decoupling of Measuring the Response Time of a Transponder and Its Authentication". Pat. 7, 2018.

128. R. Trujillo-Rasua. "Complexity of Distance Fraud Attacks in Graph-Based Distance Bounding". In: *Mobile and Ubiquitous Systems: Computing, Networking, and Services*. Vol. 131. Springer International Publishing, Cham, 2014, pp. 289–302. ISBN: 978-3-319-11568-9 978-3-319-11569-6. DOI: `10.1007/978-3-319-11569-6_23`.

129. R. Trujillo-Rasua. "Secure Memory Erasure in the Presence of Man-in-the-Middle Attackers". *Journal of Information Security and Applications* 57, 2019, p. 102730.

130. R. Trujillo-Rasua, B. Martin, and G. Avoine. "The Poulidor Distance-Bounding Protocol". In: *Radio Frequency Identification: Security and Privacy Issues*. Vol. 6370. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010, pp. 239–257. ISBN: 978-3-642-16821-5 978-3-642-16822-2. DOI: `10.1007/978-3-642-16822-2_19`.

131. G. Tsudik. "Message Authentication with One-Way Hash Functions". *ACM SIGCOMM Computer Communication Review* 22:5, 2, 1992, pp. 29–38. ISSN: 0146-4833. DOI: `10.1145/141809.141812`.

132. Y.-J. Tu and S. Piramuthu. "RFID Distance Bounding Protocols". In: *First International EURASIP Workshop on RFID Technology*. Citeseer, 2007, pp. 67–68.

133. D. Unruh. "Random Oracles and Auxiliary Input". In: *Annual International Cryptology Conference*. Springer, 2007, pp. 205–223.

134. M. Vanhoef and F. Piessens. "Key Reinstallation Attacks: Forcing Nonce Reuse in WPA2". In: *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. CCS '17: 2017 ACM SIGSAC Conference on Computer and Communications Security. ACM, Dallas Texas USA, 30, 2017, pp. 1313–1328. ISBN: 978-1-4503-4946-8. DOI: `10.1145/3133956.3134027`.

135. A. Varshavsky, A. Scannell, A. LaMarca, and E. De Lara. "Amigo: Proximity-based Authentication of Mobile Devices". In: *International Conference on Ubiquitous Computing*. Springer, 2007, pp. 253–270.

136. L. C. Washington. *Elliptic Curves: Number Theory and Cryptography*. CRC press, 2008.

137. A. C. Yao. "Theory and Application of Trapdoor Functions". In: *23rd Annual Symposium on Foundations of Computer Science (SFCS 1982)*. IEEE, 1982, pp. 80–91.

138. T. Ylonen and C. Lonvick. *The Secure Shell (SSH) Connection Protocol*. 2006.

139. J. Zhang, Z. Wang, Z. Yang, and Q. Zhang. “Proximity Based IoT Device Authentication”. In: *IEEE INFOCOM 2017-IEEE Conference on Computer Communications*. IEEE, 2017, pp. 1–9.