Università degli Studi di Salerno

Dottorato di Ricerca in Informatica e Ingegneria dell'Informazione
Ciclo 31 – a.a 2017/2018

TESI DI DOTTORATO / PH.D. THESIS

On Provable Security of Entity Authentication Schemes

MANUELA FLORES

SUPERVISOR: **PROF. BARBARA MASUCCI**

PHD PROGRAM DIRECTOR: **PROF. PASQUALE CHIACCHIO**

Dipartimento di Ingegneria dell'Informazione ed Elettrica
e Matematica Applicata
Dipartimento di Informatica

Contents

To my family,
to my father Tito Livio and my mother Vittoria,
for their example of life and for having always believed in me,
to my sister Germana,
for always helping me grow culturally,
to my brother Andrea,
for beginning this career with me,
and to my husband Costantino,
with whom I shared joys and difficulties of this challenge
and without his support I could never have reached this
long-awaited goal.

Abstract

Entity authentication is the process allowing a user, in a distributed system, to gain confidence in the identity of one (or more) communication user. Such a process may be either *unilateral* (the users are involved in a conversation in which only one of them, called the *verifier*, gains confidence that it is the other, called the *prover*, with whom he is speaking) or *mutual* (both users gain confidence about the identity of the communication partner). Moreover, the users might share some secret information, or might not.

A *one-message unilateral entity authentication scheme* allows one party, called the *prover*, to authenticate himself, i.e. to prove his identity, to another party, called the *verifier*, by sending a single *authentication message*. We consider schemes where the prover and the verifier do not share any secret information, such as a password, in advance.

We propose the *first theoretical characterization* for one-message unilateral entity authentication schemes, by formalizing the security requirements for such schemes with respect to different kinds of *passive* and *active* adversaries. More in details, we consider both *static* and *adaptive* adversaries for each kind of attack (passive/active). Afterwards, we explore the relationships between the security notions resulting from different adversarial behaviours for one-message unilateral entity authentication scheme.

Finally, we propose three different constructions for one-message unilateral entity authentication schemes and analyse their security with respect to the different security notions previously formalized in the work.

Acknowledgements

I would like to express many thanks to the person who mostly inspired me to work in cryptography, my supervisor Prof. Barbara Masucci. She has taught me how to consider problems and how to present ideas with clarity. Her generous support, motivation and numerous suggestions for improving this work have proved to be invaluable. For this, and for the time and effort she put into working with me and guiding me, I am very thankful and I hope that I will have many opportunities to continue working with her in the future.

Prof. Alfredo De Santis has also significantly influenced my research: I have had the pleasure of working with him and learning from him since I was an undergraduate student. He has been a great source of knowledge and advice for me.

Walking on the paths of the memories, I am also very grateful to Prof.ssa Filomena De Santis, Prof. Enrico Fischetti, Prof.ssa Margherita Napoli and Prof. Alfredo De Santis again, for making me love the Computer Science and for transmitting me their passion for this wonderful scope of knowledge.

Throughout all the time spent at the Dipartimento di Informatica I could count on the help of some kindly people, whenever I needed it. I have been inspired by them and I have enjoyed exchange of ideas and advices with them. Thus, I would like to thank Andrea Bruno, Aniello Castiglione, Arcangelo Castiglione, Prof. Paolo D'Arco, Prof. Roberto De Prisco, Christian Esposito, Prof. Francesco Palmieri and Raffaele Pizzolante for providing me with the ideal conditions to work.

Also I would like to thank the other Ph.D. students at the Dipartimento di Informatica, Marco Mannetta and Paolo Musmarra, who made my Ph.D. experience enjoyable, by providing me friendship, as well as technical expertise throughout the years.

Finally, I would like to thank my family, for the support and constant presence.

Chapter 1

Introduction

The mission of Cryptography is well explained by R. Rivest in a famous survey [1] with these words: "Cryptography is about communication in the presence of adversaries".

Indeed, "Communication" and "Adversary" are keywords to understand the reasons underlying Cryptography. Roughly speaking, without parties and the need of communication among parties, Cryptography is almost useless. A user talking and fighting against himself can be a curious scenario but not so interesting to justify Cryptography. Analogously, in an honest world, where people respect rules and perform perfectly operations required in order to achieve a certain functionality, it does not make sense an investigation of the "safety measures" to take in presence of deviated behaviours. Nevertheless, both conditions are not satisfied: in the real world people need to communicate to perform joint operations and several intriguing reasons drive some of them to misbehaviours and unpredictable actions.

To achieve a certain functionality, users usually perform some private computations on their own inputs and exchange messages among them, according to a well-defined sequence of steps, referred to as the protocol. Several communication media, like point-to-point channels or broadcast channels, can be available across the network and can be used to communicate. Assuming reliability of the communication infrastructure, a protocol is said to be correct

if the users compute the prescribed functionality, when each of them performs correctly the steps assigned to him by the protocol. Correctness is sufficient in an ideal world because misbehaviours do not exist.

But in the real world, for several reasons, other users of the network, an external part or someone else, can desire to avoid that part of the users reach their goal. Just to name few examples, if some users run a protocol to privately communicate, some others can be interested in eavesdropping. If some users need to agree on a common plan, one of them can work, broadcasting forged messages, in order to make impossible such an agreement. During an electronic vote competition, some players can be willing to know the vote given by some other players while, of course, the voting protocol should guarantee the privacy.

As stated by O. Goldreich [2], a cryptographic protocol enables users to compute a given functionality even in a hostile environment. In other words, it is *functionality-preserving* even if some users deviate from the protocol: or, we can even say that, it is strong enough to enable an honest majority of users to reach their goal.

Actually in Cryptography is common practice, rather than considering groups of users or other entities deviating from the protocol, to consider an "Adversary" who *controls and coordinates* a bunch of dishonest parties, according to a certain strategy, in order to break the protocol.

This notion is a "figurative" representation of attacks that can be performed, and enables a general and intuitive description of a certain hostile environment. Different characterizations of such a notion, give rise to different settings in which the analysis of a certain protocol, achieving some functionality, can be done. If the protocol preserves the functionality for which it has been designed, the protocol is considered secure against the considered Adversary. Thus, the concept of security is *Adversary-dependent.*

However, it does not make sense an analysis against a specific strategy the adversary can apply. The only reasonable assumption concerns with his own *computational power.* Two distinct assumptions have been considered and these assumptions give rise to two

different worlds. In the first case the Adversary is *unbounded*: he can use unlimited resources like computation time and memory space. In the second one the adversary is *bounded*, which means that the amount of available resources is upper-bounded by a "certain polynomial". According to the Complexity Theory, in the latter case, the adversary can perform only feasible computations, while in the former, even theoretical but infeasible computation can be carried out. The first assumption gives rise to the strongest security concept: a protocol proven secure with respect to the first Adversary is called *unconditionally secure*. On the other hand, a protocol proven secure under the second assumption is referred to as *computationally secure*.

From an historical point of view, the first notion of security was introduced by C. E. Shannon [3], while the second one can be found in the seminal paper by W. Diffie and M. Hellman [4], describing the idea of Public Key Cryptography.

Notice that, unconditionally secure means *perfectly secure*. The drawback is that such a notion requires a huge amount of resources. On the other hand, resources can be saved in the computational setting, paying in terms of a weaker security concept.

Almost all practical and commercial protocols refer to the computational setting. Assuming the existence of the so called "difficult" problems [5, 6, 7] like *integer factorization*, *discrete log computation*, *one-way functions* and so on, for which the best known algorithms require *exponential time*, it is possible to design protocols for any given functionality. And, even if a formal proof of the difficulty of such problems does not exist, there is enough evidence at the state of the current knowledge to believe that these assumptions, on which the protocols are proven to be secure, are solid ones. Moreover, the existence of an efficient algorithm, contradicting some of the well-accepted assumptions, would have unexpected and very surprising consequences in Complexity Theory and, generally, in Computer Science.

Cryptography and network security make it possible the porting on the Internet of many services that require some security properties in order to satisfy the requirements of the involved par-

ties. Many services on the Internet are restricted only to sets of "qualified users". This happens for example in subscription-based services (e.g., on-line magazines based). In such services when a qualified user is interested in a restricted resource, he contacts the *service provider* (i.e., the entity that manages the restricted resources) and proves the possession of a *credential* that only a qualified user can have. If the proof of possession of a credential is successfully verified by the service provider, the user is considered a qualified user and his subsequent requests will be honoured. Such a step is called *entity authentication* or *identification.*

Different network protocols guarantee different levels of security and different security properties. In general, strong security properties increase the complexity (in terms of computational power and amount of communication needed). More in details, many protocols supporting strong security properties are impractical if the power of the workstations connected to the Internet and the bandwidth of communication links are considered. Moreover, even if new technologies have been recently introduced such advances, they are balanced by the expansion of the Internet in terms of connected users and available services.

In general for service providers an authentication scheme in which the user exposes his identity is good since they can identify the user and then check that it is in the set of qualified users. In this case the authentication task is efficient in both computation and communication complexity.

But, when this kind of cryptographic protocols are used, then the damage that a malicious user could make must be considered. In fact, a setting in which users' activities are linkable to their identities may be exploited by a malicious user, in order to perform malicious activities in case of it cannot be identified as the author of the damage.

Therefore, it is clear the importance of an entity authentication protocol that is *efficient*, *correct* but above all *secure.*

1.1 Organization of the Thesis

In this introduction we have provided an overview concerning the scenarios and motivations of entity authentication. In the following chapters we develop the formal frameworks and present our results.

The results presented in this thesis are based on joint works with Barbara Masucci and Alfredo De Santis. The organization of the rest of it is as follows.

- **Chapter 2**: In this chapter we show some notations, conventions and definitions used throughout the thesis.
- **Chapter 3**: In this chapter we recall some formal definitions about Complexity Assumptions and Random Oracle Paradigm.
- **Chapter 4**: In this chapter we define Entity Authentication Schemes and show three constructions of these schemes. The results presented in this chapter can be found in [8].
- **Chapter 5**: In this chapter we prove security of Entity Authentication Schemes, against passive attacks for the first construction and active attacks for the other two constructions previously defined. Moreover, we show relations between all kinds of adversaries. The results presented in this chapter can be found in [9].
- **Chapter 6**: Finally, in this chapter we conclude the thesis, by providing discussions and some final remarks.

Chapter 2

Cryptographic Preliminaries

Every cryptosystem needs to be proven secure. The common practice of proving a scheme secure is to show that breaking it implies solving a mathematical problem that is known or believed to be hard. The mathematical problems are referred to as complexity assumptions. In Section 2.1 we provide a set of preliminaries that will make the thesis self contained. In Section 3.1 we list down some of the assumptions required for this thesis.

We start this chapter with some notations and conventions used throughout the thesis. The definitions in the following section are taken from [5, 10, 11, 12, 13]

2.1 Notations and Conventions

A finite string of bits is notated as $\{0,1\}^*$. If a and b are finite strings of bits then $|a|$ is the length of a and $a||b$ is their concatenation. Let $\mathbb{Z}$ denote integers and gcd represent the greatest common divisor then:

Definition 2.1.1. *The integers modulo n, denoted $\mathbb{Z}_n$, is the set of (equivalence classes of) integers $\{0,1,2,n-1\}$. Addition, subtraction, and multiplication in $\mathbb{Z}_n$ are performed modulo n.*

Definition 2.1.2. (Groups) *A group G is a set with an operation (for instance in multiplicative groups the operation is often written ".") which*

- *Is closed.* $\forall a, b \in G$ *then* $a.b \in G$
- *Has an identity* I. $\forall a \in G, a.I = I.a = a$
- *Is associative.* $\forall a, b, c \in G, (a.b).c = a.(b.c)$
- *Every element has an inverse.* $\forall a \in G$, *there exist an* a^{-1} *where* $a.a^{-1} = a^{-1}.a = I$.

A group which is commutative is often called abelian. Commutative implies that $\forall a, b \in G, a.b = b.a$. The majority of groups that are cryptographically interesting are abelian. A finite group is a group which has finite number of elements.

Definition 2.1.3. (Fields) *A field is an additive abelian group* $\mathbb{F}$ *with identity* 0, *such that* $F\backslash\{0\}$ *also forms an abelian group with respect to another operation (which is usually written multiplicatively). The two operations, addition and multiplication, are linked via the distributive law:* $a.(b + c) = a.b + a.c$.

It is easy to prove that the multiplicative group of $\mathbb{Z}_n$ *is* $\mathbb{Z}_n^* = \{a \in \mathbb{Z}_n | \gcd(a, n) = 1\}$. *In particular if* n *is a prime, then* $\mathbb{Z}_n^* = \{a \in \mathbb{Z}_n | 1 \leq a \leq n - 1\}$.

Let G be an arbitrary group. Throughout the thesis we will be making use of multiplicative groups (i.e. the binary operation of a group is ".").

Definition 2.1.4. *The order of a finite group* G *is the number of elements in the group, namely* $|G|$.

Definition 2.1.5. *The order of* $\mathbb{Z}_n^*$ *is the number of elements in the group, namely* $|\mathbb{Z}_n^*|$.

Definition 2.1.6. *A group* G *is cyclic if there is an element* $g \in G$ *such that for each* $\overline{g} \in G$ *there is a positive integer* a *with* $\overline{g} = g^a$. *Such an element* g *is called a generator of* G.

Groups of prime order have useful properties and are widely used in cryptography. All groups of prime order are cyclic. An

element h of a group G is called non-trivial if it is not equal to the identity element of the group. Suppose G is a group of order p where p is a prime, and let h be a non-trivial member of G, then h is a generator of G. There are more properties for such groups; however it is beyond our thesis scope to go through all of them.

A group isomorphism is a map between two groups that sets up a one-to-one correspondence between the elements of the groups in a way that respects the given group operations.

Definition 2.1.7. *A computable isomorphism (or bijective homomorphism) ψ from G_2 to G_1 exists, if given an element in G_2 it is possible to map it to an element in G_1. In other words $g_1 = \psi(g_2)$ where $g_1 \in G_1$ and $g_2 \in G_2$.*

A pair $(a, b) \in G^2$ means that elements a and b in the pair belong to group G. The set $\{a_1, a_2, \ldots, a_u\} \in \mathbb{Z}^u$ implies all u elements in the set belong to Z. The notation $a \in_R X$ means choosing the element a randomly from X where X can be a group G, a set S, a set of integers $\mathbb{Z}_p$, etc.

2.2 Probabilistic Algorithms

After covering some group theory terminologies we should define different types of algorithms that are relevant to our thesis and widely used in cryptography and complexity theory. We shall explain the difference between probabilistic and deterministic algorithms and define the terms "Probabilistic Algorithms", "Polynomial Time Algorithms" and "Probabilistic Polynomial Time Algorithms".

Probabilistic algorithms are important in cryptography since it is often that the algorithms of encryption and signature schemes are randomized and furthermore in studying the security of schemes the adversaries are also modeled as probabilistic (see Chapter 3). Therefore we clarify what is meant by probabilistic and deterministic algorithms [14].

The output of a deterministic algorithm, say y, is completely determined by its input, for example x. In other words, a sequence

of predefined steps are used in order to calculate y from x. A probabilistic algorithm, on the other hand is partly effected by a random event. The following is a definition of probabilistic algorithm.

Definition 2.2.1. (Probabilistic Algorithms) *Given an input x, a probabilistic algorithm A may toss a coin a finite number of times during its computation of the output y. The outcome of tossing the coin affects the next step in the calculation and affects the number of times the coin is tossed where the maximum bound is determined from the input x. The coin tosses are independent and fair (i.e. each side appears with probability of $1/2$).*

Following the definition of probabilistic algorithms we define a polynomial time algorithm and a probabilistic polynomial time algorithm as follows:

Definition 2.2.2. (Polynomial Time Algorithm) *An algorithm is called polynomial time if its worst-case running time function is polynomial in the input size. Any algorithm whose running time cannot be bounded by a polynomial is called super polynomial time.*

An algorithm is probabilistic polynomial time (PPT) if it uses randomness (e.g. flipping coins) and its worst case running time is polynomial in input size. The following is a formal definition:

Definition 2.2.3. (Probabilistic Polynomial Time Algorithm) *A probabilistic algorithm A is a probabilistic polynomial time algorithm if the running time of $A(x)$ is bounded by $P(|x|)$ where P is a polynomial. The running time is measured by the number of steps in the model algorithm (i.e. The number of steps in a probabilistic Turing machine). Tossing a coin is one step in this model.*

In any cryptographic scheme we have some parameters used in the setup of the system that determines the length of keys, messages and running times of honest parties and attackers. Everything is typically polynomially bounded by such a parameter, that is referred to as the security parameter and can take arbitrary large

values. The notion of negligible functions, as Bellare defines it in his work [15], is used in theoretical cryptography to formalize the notion of a function asymptotically "Too Small to Matter". A formal definition of a negligible function helps in saying that a cryptographic primitive or scheme have a certain level of security and that is by providing a robust notion of rareness. A rare event should occur rarely even when repeating an experiment for feasible number of times. In this case the experiment involves an adversary trying to break a scheme. A function is called negligible if it vanishes faster than the reciprocal of any polynomial. A more formal definition is given below:

Definition 2.2.4. (Negligible Function) *A function* $\epsilon : \mathbb{N} \to \mathbb{R}$ *is* negligible *if for every constant* $\ell > 0$ *there exists an integer* τ_ℓ *such that* $\epsilon(\tau) < \tau^{-\ell}$ *for all* $\tau \geq \tau_\ell$.

The advantage of an adversary is the measure of how successful it is in attacking the scheme. To assume a complexity assumption is hard or a cryptosystem is secure, the advantage of an attacker succeeding should be $Adv(k) \leq \epsilon$ where ϵ is negligible and k is a security parameter.

In this thesis we use the standard notation, introduced in [16], for the description of probabilistic algorithms and experiments. Let $A(\cdot, \cdot, \ldots)$ be a probabilistic algorithm, we denote by $a \leftarrow A(x, y, \ldots)$ the experiment of running A on inputs $x, y, \ldots$ and letting a be the output, where the probability is taken over the coin tosses of A. Similarly, given a set X, we denote by $x \leftarrow X$ the experiment of choosing uniformly at random an element x from X. If w is neither an algorithm nor a set, then $x \leftarrow w$ denotes a simple assignment statement.

Chapter 3

Provable Security

In the past, cryptosystems offered very little security guarantees. They were designed, in an ad hoc fashion where the system is proposed, attacked, broken, and repaired [17]. Cryptographers were not satisfied with such an approach. They needed the scheme to be proved mathematically secure before being used. In 1949 Shannon came up with the first significant attempt to prove a scheme secure [18]. He came up with the term "perfectly" secure cipher. Informally, we can say an encryption scheme is perfectly secure if a ciphertext does not reveal anything about the plaintext without the key.

The modern approach to proving schemes secure is referred to as "Provable Security". The basis of such a method is to prove that if there were an adversary capable of breaking a certain security concept then that adversary is able to solve a computationally intractable problem. By that we imply that such an adversary solves a complexity assumption that is believed to be hard (see Section 3.1).

Usually the cryptographic system has an adversarial model represented as some game with an adversarial goal. The goal should capture what it means to break a scheme and the game itself represents what capabilities are given to the adversary in order to achieve such a goal.

In such game models the assumption is there exist an adversary

that can break the security condition of a scheme. The adversary in this case is a "black box simulation" where assuming that black box is given certain inputs, the output is returned and that output is the result of achieving the adversarial goal. The same simulation is then used to refute the complexity assumption.

Loosely speaking, to prove that a scheme is secure one shows that breaking it can be no easier than solving some mathematical problem that is assumed to be intractable.

The following section explains different complexity assumptions used in this thesis.

3.1 Complexity Assumptions

3.1.1 One-Way Function Families

The notion of one-way function was first proposed by Diffie and Hellman [4] and later formalized by Yao [19]. Such kind of functions, which are, informally, easy to compute and hard to invert, are among the most fundamental primitives in cryptography and many other cryptographic primitives, such as one-way hash functions and pseudorandom generators, can be constructed under the sole assumption that one-way functions exist [20, 21]. We first recall the definition of function families.

Definition 3.1.1. *A tuple* $\Pi = (Gen, Samp, f)$ *of polynomial-time algorithms is a* function family *if:*

- *The probabilistic* parameter-generation algorithm *Gen, on input a security parameter* 1^τ*, outputs a set of parameters* I*, where* $|I| \geq \tau$*. Each value of* I *defines two sets* D_I *and* R_I *which corresponds to the domain and range of a function* $f_I : D_I \rightarrow R_I$*.*

- *The probabilistic* sampling algorithm *Samp, on input* I*, outputs a uniformly distributed element of* D_I*.*

- *The deterministic* evaluating algorithm f, *on input* I *and* $x \in D_I$, *outputs an element* $y \in R_I$. *We denote this by* $y = f_I(x)$.

We are interested in function families which are *hard to invert*, i.e., such that it is infeasible for any probabilistic polynomial-time adversary to find a preimage of a given value $y \in R_I$. This is formalized in the next definition.

Definition 3.1.2. *Let* $\Pi = (Gen, Samp, f)$ *be a function family and let* A *be an adversary. Consider the following* inversion experiment*:*

Experiment $\mathbf{Invert}_{A,\Pi}(1^\tau)$
$I \leftarrow Gen(1^\tau)$
$x \leftarrow Samp(I)$
$y \leftarrow f_I(x)$
$x' \leftarrow A(I, y)$
if $f_I(x') = y$, *then output* 1
else output 0

The advantage of A *in the inversion experiment is defined as*

$$\mathbf{Adv}^{\mathbf{Invert}}_{A,\Pi}(1^\tau) = Pr[\mathbf{Invert}_{A,\Pi}(1^\tau) = 1].$$

The function family Π *is said to be* one-way *if for each adversary* A, *whose time complexity is polynomial in* τ, *the function* $\mathbf{Adv}^{\mathbf{Invert}}_{A,\Pi}(1^\tau)$ *is negligible.*

3.1.2 One-Way Permutation Families

One-way permutation families represent a fundamental cryptographic primitive enabling elegant constructions of a wide variety of other primitives, such as universal one-way hash functions and pseudorandom generators [22, 23]. Despite the fundamental role of one-way permutation families in cryptography, only few candidates, based on the hardness of problems related to discrete logarithms

and factoring [24, 25], have been proposed over the years. Only recently, it has been shown how to construct a one-way permutation family that is not based on such assumptions [26].

One-way permutation families are explained in the following.

If in a function family, for each value of I output by $Gen(1^\tau)$, it holds that $D_I = R_I$ and the function $f_I : D_I \to D_I$ is a bijection, then the function family $\Pi = (Gen, Samp, f)$ is said to be a *permutation family.* This is formalized in the next definitions.

Definition 3.1.3. *A tuple* $\Pi = (Gen, Samp, f)$ *of polynomial-time algorithms is a* permutation family *if:*

- *The probabilistic* parameter-generation algorithm *Gen, on input a security parameter* 1^τ*, outputs a set of parameters* I*, where* $|I| \geq \tau$*. Each value of* I *defines a set* D_I *which corresponds to the domain and range of a permutation* (i.e., bijection) $f_I : D_I \to D_I$.

- *The probabilistic* sampling algorithm *Samp, on input* I*, outputs a uniformly distributed element of* D_I.

- *The deterministic* evaluating algorithm f*, on input* I *and* $x \in D_I$*, outputs an element* $y \in D_I$*. We denote this by* $y = f_I(x)$.

Definition 3.1.4. *Let* $\Pi = (Gen, Samp, f)$ *be a permutation family and let* A *be an adversary. Consider the following* inversion experiment*:*

Experiment $\mathbf{Invert}_{A,\Pi}(1^\tau)$
$I \leftarrow Gen(1^\tau)$
$x \leftarrow Samp(I)$
$y \leftarrow f_I(x)$
$x' \leftarrow A(I, y)$
if $f_I(x') = y$*, then output* 1
else output 0

The advantage of A in the inversion experiment is defined as

$$\mathbf{Adv}^{\mathbf{Invert}}_{A,\Pi}(1^\tau) = Pr[\mathbf{Invert}_{A,\Pi}(1^\tau) = 1].$$

The permutation family Π *is said to be* one-way *if for each adversary A, whose time complexity is polynomial in* τ*, the function* $\mathbf{Adv}^{\mathbf{Invert}}_{A,\Pi}(1^\tau)$ *is negligible.*

3.1.3 Trapdor Permutation Families

Trapdoor permutation families are one-way permutation families where the parameter-generation algorithm *Gen* outputs some additional information *td*, called the *trapdoor*, along with the set of parameters I. Such an additional information *td* enables efficient inversion of f_I. This is formalized in the next definition.

Definition 3.1.5. *A tuple of polynomial-time algorithms* $\Pi = (Gen, Samp, f, Inv)$ *is a* family of trapdoor permutations *if:*

- *The probabilistic* parameter-generation algorithm *Gen, on input the security parameter* 1^τ*, outputs the pair* (I, td)*, where* $|I| \geq \tau$*. Each value of* I *defines a set* D_I *which corresponds to the domain and range of a permutation* $f_I : D_I \to D_I$*.*
- *Let* Gen_I *denote the algorithm that results by running* $Gen(1^\tau)$ *and outputting only the set of parameters* I*. Then* $(Gen_I, Samp, f)$ *is a family of one-way permutations.*
- *Let* (I, td) *be the output of* $Gen(1^\tau)$*. The deterministic* inverting algorithm *Inv, on input td and* $y \in D_I$*, outputs* $x \in D_I$*. We denote this by* $x = Inv_{td}(y)$*. It is required that with all but negligible probability over* (I, td) *output by* $Gen(1^\tau)$ *and uniform choice of* $x \in D_I$*, we have* $Inv_{td}(f_I(x)) = x$*.*

3.1.4 Hash Functions

Hash functions are deterministic functions which map a bit-string of an arbitrary length to a hashed value which is another bit-string of a fixed size. The following is a definition of hash functions [10]:

Definition 3.1.6. (Hash Function) *A hash function H is a function with the following properties below:*

- *The function H takes a message M of finite length represented as a bit-string and maps it to a bit-string of fixed length. The result is the hash-value or simply the hash of M.*
- *Given H and $M \in \{0,1\}^*$, it is easy to compute $H(M)$.*

The properties we require in the used hash functions are defined below:

Definition 3.1.7. Let H be a hash function

- *Given y from the range of H, the hash function is "preimage resistant" if it is hard to calculate any M such that $H(M) = y$.*
- *Given M_0, the hash function is "second-preimage resistant" if it is hard to find any $M_1 \neq M_0$ such that $H(M_0) = H(M_1)$.*
- *The hash function is "collision resistant" if it is computationally "hard" to find a pair M_0 and M_1 such that $H(M_0) = H(M_1)$ and $M_0 \neq M_1$.*

Hash functions are used frequently as one of the building blocks of cryptosystems and they affect highly the security of the scheme. For example they are used to create "message fingerprints".

3.2 Random Oracle Model and Standard Model

In 1993, Bellare and Rogaway proposed the Random Oracle Model [27]. Random oracle is a cryptographic theoretic tool used in proving the security of a scheme. More in details, it is an oracle that when queried responds with a random reply, subject to the condition that the reply is different for various queries but it is the same when the same input is queried again. Such an oracle has the desired properties such as preimage resistance and collision resistance. It is frequently used as provable security model. Cryptographers assume that all entities in the game model have access to the random oracle. Then the scheme is proved secure under that random oracle assumption.

Any security proof that does not require the random oracle model is referred to as a proof under the Standard Model. Proofs under standard models are considered stronger since no assumption is made.

The results of our thesis are proved in the Standard Model.

Chapter 4

Entity Authentication

The topic of this work is *entity authentication*, which we can also call *identification*. In short, an entity authentication wants to allow confirmation of someone's identity. Normally this happens in a synchronous way. In contrast, other cryptographic tools, such as signature schemes, allow data authentication, which can be carried out in an asynchronous way.

4.1 Entity Authentication Schemes

An entity authentication scheme may be either unilateral (the users are involved in a conversation in which only one of them, called the *verifier*, gains confidence of who is the other, called the *prover*, with whom he is speaking) or mutual (both users gain confidence about the identity of the communication partner). Moreover, the users might share some secret information, or might not.

To prove the identity of someone, you can behave in three ways, based on:

- "What you are" (physical attributes): to identify an individual it is very useful to consider unique characteristics of the individual, such as fingerprint and retinal scan; automatic authentication schemes sometimes take advantage of these biometric features, but in the future they are expected to be used more and more often;
- "What you have" (documents or credentials): legal documents

such as identity card, passport or driver's license are often used as credentials; often these contain photographs that allow the physical recognition of their owner;

- "What you know" (passwords, personal information and any other knowledge): in this context, by knowledge we mean password or pin; the difficulty is that such knowledge may not be secret or could be part of the authentication process and this could allow malicious impersonations; thanks to cryptography it is possible to build secure schemes that prevent this type of attacks.

Typical entity authentication scenarios are:

– *ATM money withdrawals or credit card payments*
Only who knows the PIN is the owner of the bankcard. This is a case of "what you have" (the card) with "what you know" (the PIN).

– *Credit Card buying without physically present card*
On internet, to buy with credit card is necessary a valid credit card number, the validity date and the three digit number on the back of the card (CCV2). This is a case of "what you know".

– *Contactless credit card purchases*
In this kind of purchase, communication is done by RFID and the card has to be only near the card reader. This is a case of "what you have".

– *Remote login*
To log in to a computer or site on the web, is needed a valid username and a related password. The name is frequently only an email address. This is a case of "what you know".

– *Access to a workplace*
Controlled access workplaces use systems such as user badges or biometric recognition; the first case is "what you have" (badge), the second case is "what you are" (biometrics).

Recently, two-factor authentication is a technique that has become widespread; in addition to the password, the user is asked to specify a random number generated by an hardware token or received by SMS on the mobile number owned by the user. In this case, one factor is of type "what you know" (password) and other factor is of type "what you have" (token/sms).

4.2 Passwords

The most used method for remote entity authentication is *password*. One of its critical issues depends on the choice of weak passwords by users. For this reason, they are imposed rules on the length and on the type of symbols. These rules make online attacks more difficult, where a malicious user tries to guess the victim's password in real time.

Offline attacks are more difficult to avoid. They can occur when a data breach occurs, that is when, for example, the attacker gets hold of the file containing the user's id and password. This suggests encrypting this file, but the attacker could also get hold of the decryption key. Therefore, it is suggested to hash the passwords and store only their *fingerprint* (the hashed password). The hash function that is used has to be secure (we introduced this idea in Section 3.1.4).

Therefore, the password file contains usernames and password fingerprints. If an attacker gets hold of the password file, he can hash the possible passwords and compare it with the value saved on the file. The two most common attacks are that of the *dictionary* and that of *brute force*. In the first, you try all the possible words as they appear on a dictionary and in the second, you try all the possible strings of prefixed length.

It is important to note that the attacker may have plenty of time to make these attempts, or he can pre-calculate a table made of hypothetical passwords and their fingerprints. Therefore, he can compare the fingerprint on the password table with that on

his own table, thus go back to the original password and speed up the search process. Different users can choice a weak password and all occurrences of the same weak password can be detected simultaneously (*password-guessing attack*). To avoid this, there is a further strategy, used by Unix systems since 1970s: add a random value (*salt*) at the end of the password before calculating the hash. So, we compute *fingerprint* = *h(password* $||$ *salt)* instead of *fingerprint* = *h(password)*. Thus, password table has username, salt and fingerprint, where the salt may not be secret. Since the hash function is collision resistant, we will not have two identical fingerprints, even if they come from the same password. Another beneficial aspect given by the salt is that it makes it unreasonable to precompute an helpful table of fingerprints.

However, the salt technique does not add security against an attacker who wants to find out the password of a given user. In fact, he can perform the same attacks as described above (dictionary, brute force) because the salt is not said to be confidential information, so it is assumed that anyone can know it.

To overcome this problem, another method is used, called *key stretching*, which applies the hash to the password (and possibly also to the salt) a very large number of times, for example from 1000 to 10000. Therefore, the calculation of the fingerprint is slowed down and exhaustive research is complicated. With this technique, however, it is not possible to recover the forgotten password, but only to reset it.

4.2.1 Lamport Password Scheme

An intruder cannot learn the user password if he gain access to information in the system and the password is stored by its fingerprint instead of in clear. Nevertheless, such schemes are vulnerable to *reply attacks*: if a passive adversary is able to eavesdrop a password sent by the prover to the verifier, then later, he will be able to use it to impersonate the prover. Therefore, an eavesdropper can intercept the user's communication with system, listening over the channel, so discover the hash of the password and subsequently

impersonate the user. To prevent this *reply attack*, one must use a sequence of passwords $x_1, x_2, \ldots, x_{1000}$, each one for each session, where x_i is the password by which the user identifies himself for the ith time.

(Of course, the value 1000 is quite arbitrary. The assumption we will tacitly make is that 1000 is small enough so that it is "feasible" to perform 1000 "easy" computations.)

The system must know the sequence $y_1, y_2, \ldots, y_{1000}$, where $y_i = F(x_i)$, and each y_i must be distinct to prevent an intruder from reusing a previous password (*replay attack*).

F is a one-way function, i.e. a function such that:

(1) Given a word x, it is easy to compute $F(x)$.

(2) Given a word y, it is not feasible to compute a word x such that $y = F(x)$.

There are two obvious schemes for choosing the passwords x_i:

1. All the x_i are chosen initially, and the system maintains the entire sequence of values $y_1, y_2, \ldots, y_{1000}$ in its storage.

2. The user sends the value y_{i+1} to the system during the ith session, after logging on with x_i.

Neither scheme is completely satisfactory: the first because both the user and the system must store 1000 pieces of information, and the second because it is not robust, communication failure or interference from an intruder could prevent the system from learning the correct value of y_{i+1}.

In 1981, L. Lamport [28] presented a method that combines the best features of both ideas without these drawbacks.

His solution is to let the ith password $x_i = F^{1000-i}(x)$ for some fixed word x, where F^n denotes n successive applications of F.

Thus, the sequence of 1000 passwords is:

$$F^{999}(x), \ldots, F(F(F(x))), F(F(x)), F(x), x$$

where $F^{999}(x)$ is the first password and x is the last one.

The sequence of y_i needed by the system to authenticate these passwords is:

$F^{1000}(x), \ldots, F(F(F(x))), F(F(x)), F(x)$
where $F^{1000}(x)$ is the first value of the system and $F(x)$ is the last one.

Since it is feasible to compute F^n for $n \leq 1000$, property (2) of the one-way function implies that these y_i are distinct. It follows from the above definition that $y_i = x_{i-1}$ for $i \geq 2$. In other words, each user password is the value needed by the system to authenticate the next password. Hence, the system must initially be given the value $y_1 = F^{1000}(x)$ and need subsequently remember only the last password sent by the user, because $x_1 = y_2$, and y_2 is useful to the next password authentication. If an eavesdropper can intercept one password $x_i = F^{1000-i}(x) = y_{i+1}$, he cannot compute the next password $x_{i+1} = F^{-1}(y_{i+1})$, because F is a one-way function so it is hard to invert. Thus, each password can be used just once and later is discarded.

Such a kind of authentication, based on one-time passwords (OTP), is not subject to *reply attacks*. The protocol was then used by Bellcore (Bell Communication Research) to realize S/KEY [29] and later led to Internet RFC 2289 [30]. A subsequent implementation by the U.S. Naval Research Labs was OPIE (One-time Passwords In Everything) [31]. Afterwards, different kinds of one-time password authentication schemes have been proposed, either to improve the security or to increase the efficiency [32, 33, 34, 35]. Some proposals use smart cards [36, 37, 38, 39, 40, 41, 42, 43] to authenticate a legitimate user.

Despite the wide set of different proposals, lots of them lack a rigorous proof of security and have been proven to be insecure. It is therefore desirable that confidence in a one-time password authentication scheme should derive from more than somebody's inability to break it. In fact, according to the Goldwasser-Micali paradigm [16], a scheme has provable-security, under a given com-

plexity assumption, if an adversary which breaks the scheme can be turned into another adversary which breaks the computational assumption.

4.3 One-Message Unilateral Entity Authentication Schemes

Now we consider *one-message unilateral entity authentication schemes*, to allow the prover to authenticate himself to the verifier by sending a single authentication message. This scheme belongs to the "what you know" category. Thus, it allows one party, called the *prover*, to authenticate himself, i.e. to prove his identity, to another party, called the *verifier*. More in details, we consider one-message unilateral entity authentication schemes where the prover and the verifier do not share any secret information, such as a password, in advance. On the other hand, we assume that the prover has some secret information, which is used to compute an *authentication message* which is then sent to the verifier, who only knows some public information. Such a public information is used by the verifier in order to check the validity of the authentication message. A successful execution of the entity authentication scheme convinces the verifier that he is communicating with the prover rather than with an impostor.

A one-message unilateral entity authentication scheme is defined as follows.

Definition 4.3.1. A *one-message unilateral entity authentication scheme* is a triple of polynomial-time algorithms $\Pi = (Gen, Auth, Ver)$ such that

- The probabilistic *parameter-generation algorithm* Gen, on input a security parameter 1^τ, outputs a pair $(pub, priv)$. We refer to the first element as the *public information* and to the second as the *private information for the prover.*
- The probabilistic *message authentication algorithm* $Auth$, on input a session counter $i \geq 1$, the private information $priv$

for the prover, the public information *pub*, and the sequence $hist_{i-1} = (a_1, \ldots, a_{i-1})$ of authentication messages generated by the previous $i-1$ authentication sessions, where $hist_0$ corresponds to the empty sequence, outputs the authentication message a_i for the i-th session.

- The deterministic *verification algorithm* Ver, on input a session counter $i \geq 1$, an authentication message a_i, the public information *pub*, and the sequence of authentication messages $hist_{i-1}$ generated by the previous $i-1$ authentication sessions, where $hist_0$ corresponds to the empty sequence, outputs a bit b, with $b = 1$ meaning *valid* and $b = 0$ meaning *invalid*.

It is required that for every pair $(pub, priv)$ output by $Gen(1^\tau)$, every session counter i and every a_i output by $Auth(i, priv, pub, hist_{i-1})$, it holds that

$$Ver(i, a_i, pub, hist_{i-1}) = 1.$$

Now we describe how to construct a one-message unilateral entity authentication scheme using some cryptographic primitives. In the following chapter we formalize security requirements for one-message unilateral entity authentication schemes. We consider security against *passive* and *active* adversarial behaviours.

4.4 Constructions

In this section we propose three constructions for one-message unilateral entity authentication schemes, based on three different cryptographic primitives. The first construction uses as a building block a family of one-way functions, whereas, the second one is based on a family of one-way permutations. Finally, the third construction is based on a family of trapdoor permutations. In the following chapter, we analyze the security of these constructions.

4.4.1 A Construction based on One-Way Function Families

In this section we show how to construct a one-message unilateral entity authentication scheme using as a building block a family of one-way functions. We recalled the notion of one-way function in Section 3.1.1.

We refer to the proposed construction as to the *One-Way Function Family based Entity Authentication Construction (OWFF-EAC).*

The One-Way Function Family based Entity Authentication Construction (OWFF-EAC). Let $\Pi = (Gen, Samp, f)$ be a family of one-way functions. We construct a one-message unilateral entity authentication scheme $\hat{\Pi} = (\hat{Gen}, \hat{Auth}, \hat{Ver})$ as follows:

- The parameter-generation algorithm $\hat{Gen}$, on input a security parameter 1^τ, outputs the pair $(priv, pub)$ constructed in the following way:
  - Runs the parameter-generation algorithm $Gen(1^\tau)$ to obtain the parameter I;
  - Runs the sampling algorithm $Samp(I)$ to obtain a uniformly distributed element $x_1 \in D_I$;
  - Runs the evaluating algorithm f to compute the value $y_0 = f_I(x_1)$;
  - Sets the public information pub to be equal to the pair (I, y_0);
  - Sets the secret information $priv$ to be equal to x_1.
- The authentication message-generation $\hat{Auth}$, on inputs a session counter $i \geq 1$, the prover's secret key $priv$, the public information pub, and the sequence of authentication messages $hist_{i-1} = (a_1, \ldots, a_{i-1})$, generated by the previous $i-1$ authentication sessions, where $hist_0$ is the empty sequence, outputs the authentication message $a_i = (x_i, y_i)$, as follows:

- Extracts the value x_i from $priv$;
- Runs the sampling algorithm $Samp(I)$ to obtain a uniformly distributed element $x_{i+1} \in D_I$;
- Updates $priv$ to be equal to x_{i+1};
- Runs the evaluating algorithm f to compute the value $y_i = f_I(x_{i+1})$;

- The verification algorithm $\hat{Ver}$, on inputs a session counter $i \geq 1$, an authentication message $a_i = (x_i, y_i)$, the public information pub, and the sequence of authentication messages $hist_{i-1}$, generated by the previous $i-1$ authentication sessions, where $hist_0$ is the empty sequence, outputs a bit b constructed as follows:
  - Extracts the $(i-1)$-th authentication message $a_{i-1} = (x_{i-1}, y_{i-1})$ from the sequence $hist_{i-1}$;
  - If $f_I(x_i) = y_{i-1}$ outputs 1, else outputs 0.

In the following chapter we will show that the OWFF-EAC is secure against passive attacks, but insecure against active ones. Motivated by the need to avoid active attacks, in the next subsection we propose a different construction for one-message unilateral entity authentication schemes. It is more difficult and less efficient to construct, because it uses a family of one-way permutations as a building block, but it is more safe than the previous one.

4.4.2 A Construction based on One-Way Permutation Families

In this section we show how to construct a one-message unilateral entity authentication scheme using as a building block a family of one-way permutations. More in details, the proposed construction is based on indistinguishability obfuscation [44, 45] and one-way permutations. We recalled the notion of one-way permutation in Section 3.1.2.

Now we describe how to construct a one-message unilateral entity authentication scheme using as a building block a family of one-way permutations. The proposed construction, which we refer to as the *One-Way Permutation Family based Entity Authentication Construction (OWPF-EAC)*, can be used for at most t consecutive authentications, where t is a parameter of the scheme.

The One-Way Permutation Family based Entity Authentication Construction (OWPF-EAC). Let $\Pi = (Gen, Samp, f)$ be a family of one-way permutations. We construct a one-message unilateral entity authentication scheme $\hat{\Pi} = (\hat{Gen}, \hat{Auth}, \hat{Ver})$ allowing at most t consecutive authentications as follows:

- The parameter-generation algorithm $\hat{Gen}$, on input a security parameter 1^τ, outputs the pair $(priv, pub)$ constructed in the following way:
  - Runs the parameter-generation algorithm $Gen(1^\tau)$ to obtain the parameter I;
  - Runs the sampling algorithm $Samp(I)$ to obtain a uniformly distributed element $x_0 \in D_I$;
  - Iteratively runs the evaluating algorithm f to compute the value $y_0 = f_I^{t+1}(x_0)$;
  - Sets the public information pub to be equal to the pair (I, y_0);
  - Sets the secret information $priv$ for the prover to be equal to x_0.
- The authentication message-generation $\hat{Auth}$, on inputs a session counter $i \geq 1$, the prover's secret key $priv$, the public information pub, and the sequence of authentication messages $hist_{i-1} = (a_1, \ldots, a_{i-1})$, generated by the previous $i-1$ authentication sessions, where $hist_0$ is the empty sequence, outputs the authentication message $a_i = f_I^{t-i+1}(x_0)$.
- The verification algorithm $\hat{Ver}$, on inputs a session counter $i \geq 1$, an authentication message a_i, the public information

pub, and the sequence of authentication messages $hist_{i-1}$, generated by the previous $i-1$ authentication sessions, where $hist_0$ is the empty sequence, outputs a bit b. In particular, if $i = 1$, the algorithm $\hat{Ver}$, on input the tuple $(1, a_1, pub)$, works as follows:

- Extracts the value y_0 from *pub*;
- If $f_I(a_1) = y_0$ outputs 1, else outputs 0.

If $i \geq 2$, the algorithm $\hat{Ver}$, on input the tuple $(i, a_i, pub, hist_{i-1})$, works as follows:

- Extracts the $(i-1)$-th authentication message a_{i-1} from the sequence $hist_{i-1}$;
- If $f_I(a_i) = a_{i-1}$ outputs 1, else outputs 0.

The drawback of the OWPF-EAC is that it can be used for at most t consecutive authentications, where t is a parameter of the scheme. Motivated by the need to avoid such a limitation, in the next section we propose a different construction for one-message unilateral entity authentication schemes. It is more difficult and less efficient to construct, because it uses a family of trapdoor permutations as a building block, but it can be used for an unlimited number of authentications.

4.4.3 A Construction based on Trapdoor Permutation Families

In this section we show how to construct a one-message unilateral entity authentication scheme by using as a building block a family of *trapdoor permutations*. We recalled the notion of trapdoor permutation in Section 3.1.3.

We refer to the proposed construction as to the *Trapdoor Permutation Family based Entity Authentication Construction (TPF-EAC)*.

The Trapdoor Permutation Family based Entity Authentication Construction (TPF-EAC). Let $\Pi = (Gen, Samp, f, Inv)$ be a family of trapdoor permutations. We construct a one-message unilateral entity authentication scheme $\hat{\Pi} = (\hat{Gen}, \hat{Auth}, \hat{Ver})$ as follows:

- The parameter-generation algorithm $\hat{Gen}$, on input a security parameter 1^τ, outputs the pair $(priv, pub)$ constructed in the following way:
  - Runs the parameter-generation algorithm $Gen(1^\tau)$ to obtain the pair (I, td);
  - Runs the sampling algorithm $Samp(I)$ to obtain a uniformly distributed element $y_0 \in D_I$;
  - Sets the public information pub to be equal to the pair (I, y_0);
  - Sets the secret information $priv$ for the prover to be equal to td.
- The authentication message-generation $\hat{Auth}$, on inputs a session counter $i \geq 1$, the prover's secret key $priv$, the public information pub, and the sequence of authentication messages $hist_{i-1} = (a_1, \ldots, a_{i-1})$, generated by the previous $i-1$ authentication sessions, where $hist_0$ is the empty sequence, outputs the authentication message a_i. In particular, if $i = 1$, the algorithm $\hat{Auth}$, on input the tuple $(1, priv, pub)$, works as follows:
  - Extracts the trapdoor td from the secret key $priv$ and the value y_0 from pub;
  - Runs the inversion algorithm Inv on inputs td and y_0, and sets a_1 to be equal to $Inv_{td}(y_0)$.

  if $i \geq 2$, the algorithm $\hat{Auth}$, on input the tuple $(i, priv, pub, hist_{i-1})$, works as follows:

- Extracts the trapdoor td from the secret key $priv$ and the $(i-1)$-th authentication message a_{i-1} from the sequence $hist_{i-1}$;
- Runs the inversion algorithm Inv on inputs td and a_{i-1}, and sets a_i to be equal to $Inv_{td}(a_{i-1})$.

- The verification algorithm $\hat{Ver}$, on inputs a session counter $i \geq 1$, an authentication message a_i, the public information pub, and the sequence of authentication messages $hist_{i-1}$, generated by the previous $i-1$ authentication sessions, where $hist_0$ is the empty sequence, outputs a bit b. In particular, if $i = 1$, the algorithm $\hat{Ver}$, on input the tuple $(1, a_1, pub)$, works as follows:
  - Extracts the value y_0 from pub;
  - If $f_I(a_1) = y_0$ outputs 1, else outputs 0.

  If $i \geq 2$, the algorithm $\hat{Ver}$, on input the tuple $(i, a_i, pub, hist_{i-1})$, works as follows:
  - Extracts the $(i-1)$-th authentication message a_{i-1} from the sequence $hist_{i-1}$;
  - If $f_I(a_i) = a_{i-1}$ outputs 1, else outputs 0.

Chapter 5

Provable Security of Entity Authentication Schemes

5.1 Attack Models

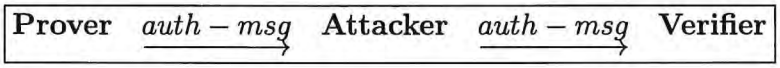

Figure 5.1 Man In The Middle Attack

There are two kinds of attack models and adversarial goals in an entity authentication scheme. In Figure 5.1 we can see a possible *intruder-in-the-middle* scenario.

Attacker, before he actually tries to fool Verifier, carries out an *information-gathering phase*. Attacker is a *passive adversary* if, during this phase, he simply observes sessions between Prover and Verifier. Alternatively, we might consider an attack model in which Attacker is an *active adversary* during the information-gathering phase. For example, Attacker might be given temporary access to an oracle that computes authentication tags for Prover and Verifier. During this period, Attacker can successfully fool Prover and Verifier, of course, by using the oracle to respond to challenges. However, after the information-gathering phase, the

oracle is confiscated, and then Attacker carries out his attack, trying to get Verifier to "accept" him in a new session in which Attacker does not have an oracle.

Due to the fact that we are designing new paradigms of cryptography, we need to design such security models that capture security notions in which the new systems require.

5.2 Security against Passive Attacks

The first security requirement we consider for an entity authentication scheme is that an adversary who does not know the prover's secret information *priv* should be unable to fool the verifier into accepting a fake authentication message. Such a requirement should hold even if the adversary is able to *passively eavesdrop* on multiple honest executions of the scheme between the prover and the verifier.

More in details, we consider the case where a *passive static adversary* attacks the i-th authentication session, where $i \geq 1$. Such an adversary, denoted by $\mathtt{PAS} - \mathtt{STAT}_i$, on input the public information *pub* generated by the algorithm *Gen*, is able to obtain the sequence $hist_{i-1}$ of authentication messages generated for the first $i-1$ sessions, where $hist_0$ denotes the empty sequence. We assume the existence of an *authentication oracle* $\mathtt{Auth}_{priv}$ that, when called without any input, runs an honest execution of the entity authentication scheme and outputs the authentication message generated for that execution. In order to get the sequence $hist_{i-1}$, the adversary $\mathtt{PAS} - \mathtt{STAT}_i$ queries the authentication oracle $\mathtt{Auth}_{priv}$ for $i-1$ times. We denote with $\mathtt{PAS} - \mathtt{STAT}_i^{\mathtt{Auth}_{priv}}()$ the interaction between the adversary $\mathtt{PAS} - \mathtt{STAT}_i$ and the oracle $\mathtt{Auth}_{priv}$. Afterwards, $\mathtt{PAS} - \mathtt{STAT}_i$ on input the public information *pub* and the sequence $hist_{i-1}$ obtained by all oracle queries, outputs a fake authentication message a_i' and succeeds whether the verifier accepts it as a valid message. We require that $\mathtt{PAS} - \mathtt{STAT}_i$ will succeed with only negligible probability.

Definition 5.2.1. [IMP-P-ST] *Let* $\Pi = (Gen, Auth, Ver)$ *be a*

one-message unilateral entity authentication scheme, let $i \geq 1$ *be a session counter, and let* $\mathtt{PAS}-\mathtt{STAT}_i$ *be a passive static adversary. Consider the following* impersonation experiment*:*

$$
\begin{array}{l}
\text{Experiment } \mathbf{Imp}_{\mathtt{PAS}-\mathtt{STAT}_i,\Pi}(1^\tau) \\
\quad (pub, priv) \leftarrow Gen(1^\tau) \\
\quad hist_{i-1} \leftarrow \mathtt{PAS}-\mathtt{STAT}_i^{\mathtt{Auth}_{priv}}(pub) \\
\quad a_i' \leftarrow \mathtt{PAS}-\mathtt{STAT}_i(pub, hist_{i-1}) \\
\quad \textit{if } Ver(i, a_i', pub, hist_{i-1}) = 1, \textit{ then output } 1 \\
\qquad\qquad\qquad\qquad\qquad\qquad \textit{else output } 0
\end{array}
$$

The advantage of $\mathtt{PAS}-\mathtt{STAT}_i$ *in the impersonation experiment is defined as*

$$\mathbf{Adv}^{\mathbf{Imp}}_{\mathtt{PAS}-\mathtt{STAT}_i,\Pi}(1^\tau) = Pr[\mathbf{Imp}_{\mathtt{PAS}-\mathtt{STAT}_i,\Pi}(1^\tau) = 1].$$

The one-message unilateral entity authentication scheme $\Pi = (Gen, Auth, Ver)$ *is said to be* secure in the sense of IMP-P-ST *if for each session counter* $i \geq 1$ *and each passive static adversary* $\mathtt{PAS}-\mathtt{STAT}_i$*, whose time complexity is polynomial in* τ*, the function* $\mathbf{Adv}^{\mathbf{Imp}}_{\mathtt{PAS}-\mathtt{STAT}_i,\Pi}(1^\tau)$ *is negligible.*

A different kind of passive adversary is the *adaptive* one. Such an adversary, denoted by $\mathtt{PAS}-\mathtt{ADAPT}$, after asking a polynomial number q of queries to the authentication oracle $\mathtt{Auth}_{priv}$, resulting in the sequence $hist_q = (a_1, \ldots, a_q)$ of authentication messages, chooses the session counter i for which the attack will be mounted. In order to avoid *replay attacks*, in which the adversary could simply reply a previous authentication message, fooling the verifier into accepting it, we require that $i \geq q + 1$. After this stage, the adversary $\mathtt{PAS}-\mathtt{ADAPT}$ is still allowed to query the authentication oracle $\mathtt{Auth}_{priv}$, in order to get the sequence $(a_{q+1}, \ldots, a_{i-1})$ of authentication messages. Afterwards, $\mathtt{PAS}-\mathtt{ADAPT}$ on input the public information *pub*, the chosen session counter i and the sequence $hist_{i-1}$ obtained by all oracle queries, outputs a fake authentication message a_i' and succeeds whether the verifier accepts it as a valid message. We require that $\mathtt{PAS}-\mathtt{ADAPT}$ will succeed with only negligible probability.

Definition 5.2.2. [IMP-P-AD] *Let* $\Pi = (Gen, Auth, Ver)$ *be a one-message unilateral entity authentication scheme and let* $\mathtt{PAS-ADAPT} = (\mathtt{PAS-ADAPT}_1, \mathtt{PAS-ADAPT}_2)$ *be a passive adaptive adversary. Consider the following* impersonation experiment*:*

$$\begin{array}{l}\text{Experiment } \mathbf{Imp}_{\mathtt{PAS-ADAPT},\Pi}(1^\tau)\\ \quad (pub, priv) \leftarrow Gen(1^\tau)\\ \quad i \leftarrow \mathtt{PAS-ADAPT}_1^{\mathtt{Auth}_{priv}}(pub)\\ \quad a_i' \leftarrow \mathtt{PAS-ADAPT}_2^{\mathtt{Auth}_{priv}}(i, pub)\\ \quad \textit{if } Ver(i, a_i', pub, hist_{i-1}) = 1, \textit{ then output } 1\\ \qquad\qquad\qquad\qquad\qquad\qquad \textit{else output } 0\end{array}$$

The advantage of $\mathtt{PAS-ADAPT}$ *in the impersonation experiment is defined as*

$$\mathbf{Adv}^{\mathbf{Imp}}_{\mathtt{PAS-ADAPT},\Pi}(1^\tau) = Pr[\mathbf{Imp}_{\mathtt{PAS-ADAPT},\Pi}(1^\tau) = 1].$$

The one-message unilateral entity authentication scheme $\Pi = (Gen, Auth, Ver)$ *is said to be* secure in the sense of IMP-P-AD *if for each passive adaptive adversary* $\mathtt{PAS-ADAPT}$*, whose time complexity is polynomial in* τ*, the function* $\mathbf{Adv}^{\mathbf{Imp}}_{\mathtt{PAS-ADAPT},\Pi}(1^\tau)$ *is negligible.*

5.3 Security against Active Attacks

In the previous section we have only considered security against adversaries which are able to *passively eavesdrop* on multiple honest executions of the schemes and whose goal is to fool the verifier into accepting an authentication message which has not been generated by the prover.

A more powerful kind of adversary is the *active* one, which is also allowed to modify authentication messages generated by the prover before they reach their destination. More in details, we consider the case where an *active static adversary* attacks the i-th authentication session, where $i \geq 2$, and is allowed to *modify* the last $i - j \geq 0$ authentication messages before they reach their

destination. Such an adversary, denoted by $\mathtt{ACT}-\mathtt{STAT}_{i,j}$, on input the public information *pub* generated by the algorithm *Gen*, is able to obtain the sequence $hist_{i-1} = (a_1, \ldots, a_{i-1})$ of authentication messages generated for the first $i-1$ sessions by interacting with the *authentication oracle* $\mathtt{Auth}_{priv}$. Afterwards, $\mathtt{ACT}-\mathtt{STAT}_{i,j}$ on input the public information *pub* and the sequence $hist_{i-1}$ obtained by all oracle queries, outputs a sequence of $i-j+1$ *fake authentication messages* $(a'_j, \ldots, a'_{i-1}, a'_i)$ and succeeds whether the verifier accepts all of them as valid messages. More in details, $a'_j, a'_{j+1}, \ldots, a'_{i-1}$ will replace the authentication messages $a_j, a_{j+1}, \ldots, a_{i-1}$, which are the last $i-j$ elements of the sequence $hist_{i-1}$ generated by the authentication oracle $\mathtt{Auth}_{priv}$; we denote by $hist'_\ell$ the modified sequence $(a_1, \ldots, a_{j-1}, a'_j \ldots . a'_\ell)$, for any $\ell = j, \ldots, i-1$. Clearly, if $j = i$, the sequence of modified messages $(a'_j, \ldots, a'_{i-1})$ is empty. We require that $\mathtt{ACT}-\mathtt{STAT}_{i,j}$ will succeed with only negligible probability.

Definition 5.3.1. [IMP-A-ST] *Let* $\Pi = (Gen, Auth, Ver)$ *be a one-message unilateral entity authentication scheme, let* i *and* j *be two session counters such that* $i \geq 2$ *and* $1 \leq j \leq i$, *and let* $\mathtt{ACT}-\mathtt{STAT}_{i,j}$ *be an active static adversary. Consider the following* impersonation experiment*:*

Experiment $\mathbf{Imp}_{\mathtt{ACT}-\mathtt{STAT}_{i,j},\Pi}(1^\tau)$
$(pub, priv) \leftarrow Gen(1^\tau)$
$hist_{i-1} \leftarrow \mathtt{ACT}-\mathtt{STAT}_{i,j}^{\mathtt{Auth}_{priv}}(pub)$
$(a'_j, \ldots, a'_{i-1}, a'_i) \leftarrow \mathtt{ACT}-\mathtt{STAT}_{i,j}(pub, hist_{i-1})$
if $Ver(j, a'_j, pub, hist_{j-1}) = 1$
and
$Ver(\ell+1, a'_{\ell+1}, pub, hist'_\ell) = 1$, *for any* $\ell = j, \ldots, i-1$,
then output 1
else output 0

The advantage of $\mathtt{ACT}-\mathtt{STAT}_{i,j}$ *in the impersonation experiment is defined as*

$$\mathbf{Adv}^{\mathbf{Imp}}_{\mathtt{ACT}-\mathtt{STAT}_{i,j},\Pi}(1^\tau) = Pr[\mathbf{Imp}_{\mathtt{ACT}-\mathtt{STAT}_{i,j},\Pi}(1^\tau) = 1].$$

The one-message unilateral entity authentication scheme $\Pi = (Gen, Auth, Ver)$ *is said to be* secure in the sense of IMP-A-ST *if for each pair of session counters* i *and* j *such that* $i \geq 2$ *and* $1 \leq j \leq i$, *each active static adversary* $\mathtt{ACT} - \mathtt{STAT}_{i,j}$, *whose time complexity is polynomial in* τ, *the function* $\mathbf{Adv}^{\mathbf{Imp}}_{\mathtt{ACT}-\mathtt{STAT}_{i,j},\Pi}(1^{\tau})$ *is negligible.*

A different kind of active adversary is the *adaptive* one. Such an adversary, denoted by $\mathtt{ACT} - \mathtt{ADAPT}$, after asking a polynomial number $q \geq 1$ of queries to the authentication oracle $\mathtt{Auth}_{priv}$, resulting in the sequence $hist_q = (a_1, \ldots, a_q)$ of authentication messages, chooses two session counters i and j, where $1 \leq j \leq q$ and $i \geq j$; the index i denotes the session counter for which the attack will be mounted, whereas, the index j corresponds to the session counter from which the adversary $\mathtt{ACT} - \mathtt{ADAPT}$ will start to modify previously seen authentication messages. After this stage, if $i \geq q + 1$, the adversary $\mathtt{ACT} - \mathtt{ADAPT}$ is still allowed to query the authentication oracle $\mathtt{Auth}_{priv}$, in order to get the sequence $(a_{q+1}, \ldots, a_{i-1})$ of authentication messages. Afterwards, $\mathtt{ACT} - \mathtt{ADAPT}$ on input the public information *pub*, the chosen session counters i, j and the sequence $hist_{i-1}$ obtained by all oracle queries, outputs $i - j + 1$ *authentication messages* $(a'_j, a'_{j+1}, \ldots, a'_{i-1}, a'_i)$ and succeeds whether the verifier accepts all of them as valid messages. We require that $\mathtt{ACT} - \mathtt{ADAPT}$ will succeed with only negligible probability.

Definition 5.3.2. [IMP-A-AD] *Let* $\Pi = (Gen, Auth, Ver)$ *be a one-message unilateral entity authentication scheme and let* $\mathtt{ACT} - \mathtt{ADAPT} = (\mathtt{ACT} - \mathtt{ADAPT}_1, \mathtt{ACT} - \mathtt{ADAPT}_2)$ *be an active adaptive adversary. Consider the following* impersonation experiment*:*

Experiment $\mathbf{Imp}_{\mathtt{ACT-ADAPT},\Pi}(1^\tau)$
$(pub, priv) \leftarrow Gen(1^\tau)$
$(i, j) \leftarrow \mathtt{ACT-ADAPT}_1^{\mathtt{Auth}_{priv}}(pub)$
$(a'_j, \ldots, a'_{i-1}, a'_i) \leftarrow \mathtt{ACT-ADAPT}_2^{\mathtt{Auth}_{priv}}(i, j, pub)$
if $Ver(j, a'_j, pub, hist_{j-1}) = 1$
and
$Ver(\ell + 1, a'_{\ell+1}, pub, hist'_\ell) = 1$, *for any* $\ell = j, \ldots, i-1$,
then output 1
else output 0

The advantage of ACT − ADAPT *in the impersonation experiment is defined as*

$$\mathbf{Adv}^{\mathbf{Imp}}_{\mathtt{ACT-ADAPT},\Pi}(1^\tau) = Pr[\mathbf{Imp}_{\mathtt{ACT-ADAPT},\Pi}(1^\tau) = 1].$$

The one-message unilateral entity authentication scheme $\Pi = (Gen, Auth, Ver)$ *is said to be* secure in the sense of IMP-A-AD *if for each active adaptive adversary* ACT − ADAPT, *whose time complexity is polynomial in* τ, *the function* $\mathbf{Adv}^{\mathbf{Imp}}_{\mathtt{ACT-ADAPT},\Pi}(1^\tau)$ *is negligible.*

5.4 Implications and Separations

In this section we explore the relationships between the security notions resulting from different adversarial behaviours for one-message unilateral entity authentication scheme. More in details, we show whether one notion implies another and vice-versa. Figure 5.2 summarizes our results.

We first prove that security against passive adaptive attacks is (polynomially) equivalent to security against passive static attacks.

Theorem 5.4.1. [IMP-P-AD⇔IMP-P-ST] *A one-message unilateral entity authentication scheme is secure in the sense of* IMP-P-AD *if and only if it is secure in the sense of* IMP-P-ST.

Proof. The first implication is trivial, since any passive adaptive adversary could behave as a passive static one attacking an authentication session i, simply by querying the oracle $\mathtt{Auth}_{priv}$ for

$i-1$ times and by choosing the authentication session i in the first stage of the attack.

Now we prove that security against passive static attacks implies security against passive adaptive attacks. Let $\Pi = (Gen, Auth, Ver)$ be a one-message unilateral entity authentication scheme which is secure against passive static attacks and assume by contradiction the existence of a passive adaptive adversary $\mathtt{PAS-ADAPT} = (\mathtt{PAS-ADAPT}_1, \mathtt{PAS-ADAPT}_2)$ whose advantage $\mathbf{Adv}^{\mathbf{Imp}}_{\mathtt{PAS-ADAPT},\Pi}(1^\tau)$ is non-negligible.

Let i be a session counter output by $\mathtt{PAS-ADAPT}_1$ with probability at least $\frac{1}{poly(\tau)}$, where the probability is taken over the coin flips of Gen and $\mathtt{PAS-ADAPT}_1$. This means that i belongs to the set of the most likely choices made by $\mathtt{PAS-ADAPT}_1$. We show how to construct a static adversary $\mathtt{PAS-STAT}_i$, using $\mathtt{PAS-ADAPT}$, such that $\mathbf{Adv}^{\mathbf{Imp}}_{\mathtt{PAS-STAT}_i,\Pi}(1^\tau)$ is non-negligible. In particular, we show that $\mathtt{PAS-STAT}_i$'s advantage is polynomially related to $\mathtt{PAS-ADAPT}$'s advantage.

The algorithm $\mathtt{PAS-STAT}_i$, on inputs the public information pub output by the algorithm Gen and the sequence $hist_{i-1}$, runs the algorithm $\mathtt{ADAPT}_1$, on input pub. Notice that $\mathtt{PAS-STAT}_i$ is able to simulate the interaction between $\mathtt{PAS-ADAPT}_1$ and the oracle $\mathtt{Auth}_{priv}$ for $i-1$ times. Indeed, for the ℓ-th query made by $\mathtt{PAS-ADAPT}_1$ to $\mathtt{Auth}_{priv}$, where $\ell \leq i-1$, $\mathtt{PAS-STAT}_i$ simply retrieves from $hist_{i-1}$ the ℓ-th authentication message a_ℓ and gives it to $\mathtt{PAS-ADAPT}_1$. On the other hand, further queries to the oracle cannot be simulated by $\mathtt{PAS-STAT}_i$, since it is not able to reply with the correct authentication message, which is not included in $hist_{i-1}$. In such a case i cannot be the output by $\mathtt{PAS-ADAPT}_1$. Let h be the session counter output by $\mathtt{PAS-ADAPT}_1$. If $i = h$, then $\mathtt{PAS-STAT}_i$ outputs the same output as $\mathtt{PAS-ADAPT}_2$. On the other hand, if $i \neq h$, $\mathtt{PAS-STAT}_i$ outputs 0. It is easy to see that

$$\mathbf{Adv}^{\mathbf{Imp}}_{\mathtt{PAS-STAT}_i,\Pi}(1^\tau) = Pr[i = h] \cdot \mathbf{Adv}^{\mathbf{Imp}}_{\mathtt{PAS-ADAPT},\Pi}(1^\tau).$$

Since i is chosen by $\mathtt{PAS-ADAPT}_1$ with probability at least $\frac{1}{poly(\tau)}$ and $\mathbf{Adv}^{\mathbf{Imp}}_{\mathtt{PAS-ADAPT},\Pi}(1^\tau)$ is non-negligible, it follows that also

$\mathbf{Adv}^{\mathbf{Imp}}_{\mathtt{PAS-STAT}_i,\Pi}(1^\tau)$ is non-negligible. Contradiction. □

Since security against passive adaptive attacks is (polynomially) equivalent to security against passive static attacks, in the rest of the paper we will only consider the second type of passive adversary.

It is easy to see that security against active static attacks implies security against passive static attacks, whilst the opposite does not hold.

Theorem 5.4.2. [IMP-A-ST⇒IMP-P-ST] *If a one-message unilateral entity authentication scheme is secure in the sense of* IMP-A-ST*, then it is also secure in the sense of* IMP-P-ST*.*

Proof. Let $\Pi = (Gen, Auth, Ver)$ be a one-message unilateral entity authentication scheme which is secure against active static attacks and assume by contradiction the existence of a passive static adversary $\mathtt{PAS-STAT}_i$ whose advantage $\mathbf{Adv}^{\mathbf{Imp}}_{\mathtt{PAS-STAT}_i,\Pi}(1^\tau)$ is non-negligible.

We show how to construct an active static adversary $\mathtt{ACT-STAT}_{i,i}$, using $\mathtt{PAS-STAT}_i$, such that $\mathbf{Adv}^{\mathbf{Imp}}_{\mathtt{ACT-STAT}_{i,i},\Pi}(1^\tau)$ is non-negligible. The algorithm $\mathtt{ACT-STAT}_{i,i}$, on inputs the public information *pub* output by the algorithm *Gen*, runs the algorithm $\mathtt{PAS-STAT}_i$ on input *pub*. Then, $\mathtt{PAS-STAT}_i$ interacts with the authentication oracle $\mathtt{Auth}_{priv}$ in order to obtain the sequence $hist_{i-1}$, and, on input *pub* and $hist_{i-1}$, outputs the fake message a'_i. Thus, $\mathtt{ACT-STAT}_{i,i}$ outputs the same output a'_i as $\mathtt{PAS-STAT}_i$. Since $\mathbf{Adv}^{\mathbf{Imp}}_{\mathtt{ACT-STAT}_{i,i},\Pi}(1^\tau) = \mathbf{Adv}^{\mathbf{Imp}}_{\mathtt{PAS-STAT}_i,\Pi}(1^\tau)$ and $\mathbf{Adv}^{\mathbf{Imp}}_{\mathtt{PAS-STAT}_i,\Pi}(1^\tau)$ is non-negligible, the theorem follows. □

In the following section, after Theorem 5.5.1, we will show a one-message unilateral entity authentication scheme which is secure against passive attacks but insecure against active ones. So, we may enunciate the following theorem.

Theorem 5.4.3. [IMP-P-ST⇏IMP-A-ST] *There exists a one-message unilateral entity authentication scheme which is secure in the sense of* IMP-P-ST *but which is not secure in the sense of* IMP-A-ST*.*

Finally, we prove that security against active adaptive attacks is (polynomially) equivalent to security against active static attacks.

Theorem 5.4.4. [IMP-A-AD⇔IMP-A-ST] *A one-message unilateral entity authentication scheme is secure in the sense of* IMP-A-AD *if and only if it is secure in the sense of* IMP-A-ST.

Proof. The first implication is trivial, since any active adaptive adversary could behave as an active static one attacking an authentication session i, simply by querying the oracle $\mathtt{Auth}_{priv}$ for $i-1$ times and by choosing the pair $(i, i-1)$ of authentication sessions in the first stage of the attack.

Now we prove that security against active static attacks implies security against active adaptive attacks. Let $\Pi = (Gen, Auth, Ver)$ be a one-message unilateral entity authentication scheme which is secure against active static attacks and assume by contradiction the existence of an active adaptive adversary $\mathtt{ACT-ADAPT} = (\mathtt{ACT-ADAPT}_1, \mathtt{ACT-ADAPT}_2)$ whose advantage $\mathbf{Adv}^{\mathbf{Imp}}_{\mathtt{ACT-ADAPT},\Pi}(1^\tau)$ is non-negligible. Let (i, j) be a pair of session counters output by $\mathtt{ACT-ADAPT}_1$ with probability at least $\frac{1}{poly(\tau)}$, where the probability is taken over the coin flips of *Gen* and $\mathtt{ACT-ADAPT}_1$. This means that (i, j) belongs to the set of the most likely choices made by $\mathtt{ACT-ADAPT}_1$. We show how to construct an active static adversary $\mathtt{ACT-STAT}_{i,j}$, using ACT − ADAPT, such that $\mathbf{Adv}^{\mathbf{Imp}}_{\mathtt{ACT-STAT}_{i,j},\Pi}(1^\tau)$ is non-negligible. In particular, we show that $\mathtt{ACT-STAT}_{i,j}$'s advantage is polynomially related to ACT − ADAPT's advantage.

The algorithm $\mathtt{ACT-STAT}_{i,j}$, on inputs the public information *pub* output by the algorithm *Gen* and the sequence $hist_{i-1}$, generated by the interaction with the authentication oracle $\mathtt{Auth}_{priv}$, runs the algorithm $\mathtt{ACT-ADAPT}_1$, on input *pub*. Notice that $\mathtt{ACT-STAT}_{i,j}$ is able to simulate the interaction between $\mathtt{ACT-ADAPT}_1$ and the oracle $\mathtt{Auth}_{priv}$ for $i-1$ times. Indeed, for the ℓ-th query made by $\mathtt{ACT-ADAPT}_1$ to $\mathtt{Auth}_{priv}$, where $\ell \leq i-1$, $\mathtt{ACT-STAT}_{i,j}$ simply retrieves from $hist_{i-1}$ the ℓ-th authentication message a_ℓ and gives it to $\mathtt{ACT-ADAPT}_1$. On the other hand, further queries to the oracle cannot be simulated by $\mathtt{ACT-STAT}_{i,j}$, since it is not able to reply with the correct authentication message, which is not included in $hist_{i-1}$. In such a case i cannot be the session counter for which the attack will be mounted by ACT − ADAPT. Let (h, k) be the pair of session counters output by $\mathtt{ACT-ADAPT}_1$. If

$h = i$ and $k = j$, then $\mathtt{ACT} - \mathtt{STAT}_{i,j}$ outputs the same output as $\mathtt{ACT} - \mathtt{ADAPT}_2$. On the other hand, if $h \neq i$ or $k \neq j$, then $\mathtt{ACT} - \mathtt{STAT}_{i,j}$ outputs 0. It is easy to see that

$$\mathbf{Adv}^{\mathbf{Imp}}_{\mathtt{ACT-STAT}_{i,j},\Pi}(1^\tau) = Pr[h = i \text{ and } k = j] \cdot \mathbf{Adv}^{\mathbf{Imp}}_{\mathtt{ACT-ADAPT},\Pi}(1^\tau).$$

Since (i, j) is chosen by $\mathtt{ACT} - \mathtt{ADAPT}_1$ with probability at least $\frac{1}{poly(\tau)}$ and $\mathbf{Adv}^{\mathbf{Imp}}_{\mathtt{ACT-ADAPT},\Pi}(1^\tau)$ is non-negligible, it follows that also $\mathbf{Adv}^{\mathbf{Imp}}_{\mathtt{ACT-STAT}_{i,j},\Pi}(1^\tau)$ is non-negligible. Contradiction. □

Figure 5.2 shows the hierarchy of security definitions for one-message unilateral entity authentication schemes, resulting from Theorems 5.4.1, 5.4.2, 5.4.3, 5.4.4.

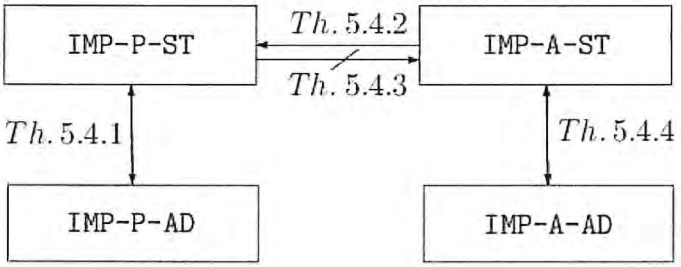

Figure 5.2 Relationships between the security notions for one-message unilateral entity authentication schemes

5.5 Security of OWFF-EAC

In the following we will show that the OWFF-EAC is secure against passive static attacks.

Theorem 5.5.1. *If* $\Pi = (Gen, Samp, f)$ *is a one-way function family, then the one-message unilateral entity authentication scheme* $\hat{\Pi}$ *obtained by the OWFF-EAC is secure in the sense of* IMP-P-ST.

Proof. Assume by contradiction that the one-message unilateral entity authentication scheme $\hat{\Pi} = (\hat{Gen}, \hat{Auth}, \hat{Ver})$ obtained by the OWFF-EAC is not secure in the sense of IMP-P-ST. Thus, there exists a polynomial-time adversary $\mathtt{PAS} - \mathtt{STAT}_i$ whose advantage $\mathbf{Adv}^{\mathbf{Imp}}_{\mathtt{PAS}-\mathtt{STAT}_i,\hat{\Pi}}(1^\tau)$ in the impersonation experiment $\mathbf{Imp}_{\mathtt{PAS}-\mathtt{STAT}_i,\hat{\Pi}}(1^\tau)$ is non-negligible. We will show how to construct a polynomial-time adversary A which, on input a set of parameters I generated by $Gen(1^\tau)$ and a uniformly distributed value $y \in R_I$, uses the adversary $\mathtt{PAS} - \mathtt{STAT}_i$ to compute the value $f_I^{-1}(y)$.
The adversary A, on input I and y, generates the input for the adversary $\mathtt{PAS} - \mathtt{STAT}_i$ as shown in the following.

- Run the sampling algorithm $Samp(I)$ to obtain a uniformly distributed element $x_1 \in D_I$;
- Run the evaluating algorithm f to compute the value $y_0 = f_I(x_1)$;
- Set the public information pub to be equal to the pair (I, y_0);
- For each $j = 1, \ldots, i-2$, computes the j-th authentication message $a_j = (x_j, y_j)$, as follows:
  - Run the sampling algorithm $Samp(I)$ to obtain a uniformly distributed element $x_{j+1} \in D_I$;
  - Run the evaluating algorithm f to compute the value $y_j = f_I(x_{j+1})$;
- Set the $(i-1)$-th authentication message a_{i-1} to be equal to (x_{i-1}, y), where y is the challenge value for adversary A.

The adversary $\mathtt{PAS} - \mathtt{STAT}_i$, on input the tuple $(i, pub, hist_{i-1})$, outputs the authentication message a_i. Then, adversary A outputs the same value a_i as its guess for $f_I^{-1}(y)$. Indeed, it holds that $f_I(x_i) = y$, since $\hat{Ver}(i, a_i, pub, hist_{i-1}) = 1$.

It is easy to see that the view of adversary $\mathtt{PAS} - \mathtt{STAT}_i$, when run as a subroutine by A, is distributed identically as in the impersonation experiment $\mathbf{Imp}_{\mathtt{PAS}-\mathtt{STAT}_i,\hat{\Pi}}(1^\tau)$. Indeed the second

element of the public information *pub* is a uniformly distributed element in R_I, since it is obtained by evaluating the function f_I at the point x_0, which is uniformly distributed in D_I. Moreover, for the same reasons, the second element of each authentication message in the $hist_{i-1}$ sequence is also a uniformly distributed element in R_I. Since the advantage of adversary $\mathtt{PAS} - \mathtt{STAT}_{i,j}$ in the impersonation experiment $\mathbf{Imp}_{\mathtt{PAS}-\mathtt{STAT}_i,\tilde{\Pi}}(1^\tau)$ is non-negligible, it follows that also the advantage of A in the inversion experiment $\mathbf{Invert}_{A,\Pi}(1^\tau)$ is non-negligible. Contradiction. □

In Theorem 5.5.1 we have considered security against adversaries which are able to *passively eavesdrop* on multiple honest executions of the schemes and whose goal is to fool the verifier into accepting an authentication message which has not been generated by the prover.

It is easy to see that security against active adversaries does not hold for the OWFF-EAC. Indeed, assume that an active static adversary $\mathtt{ACT} - \mathtt{STAT}_{i,j}$, on input the public information *pub* and the sequence $hist_{i-1} = (a_1, \ldots, a_{i-1})$, is able to modify the last $i - j$ authentication messages $a_j, \ldots, a_{i-1}$ before they reach their destination, with the goal of fooling the verifier into accepting a fake authentication message a'_i. More precisely, for any $\ell = j, \ldots, i-1$, the adversary replaces each message $a_\ell = (x_\ell, y_\ell)$ with a different message $a'_\ell = (x'_\ell, y'_\ell)$, which is sent to the verifier. In particular, the first modified message $a'_j = (x'_j, y'_j)$ can be constructed as follows: first, $\mathtt{ACT} - \mathtt{STAT}_{i,j}$ sets $x'_j = x_j$, then, it chooses an element x'_{j+1} in D_I and sets y'_j to be equal to $f_I(x'_{j+1})$. Subsequent modified messages $a'_{j+1}, \ldots, a'_{i-1}$, including the fake one a'_i can be constructed in the same way: for any $\ell = j+1, \ldots, i$, the adversary $\mathtt{ACT} - \mathtt{STAT}_{i,j}$ first chooses an element $x'_{\ell+1}$ in D_I, then sets y'_ℓ to be equal to $f_I(x'_{\ell+1})$. It is easy to see that the reception of the last modified message $a'_{i-1} = (x'_{i-1}, y'_{i-1})$ allows the adversary $\mathtt{ACT} - \mathtt{STAT}_{i,j}$ to fool the verifier into accepting the subsequent fake authentication message $a'_i = (x'_i, y'_i)$, since $f_I(x'_i) = y'_{i-1}$.

5.6 Security of OWPF-EAC

Consider an active adversary which is able to modify the authentication message a_{i-1}, generated by the prover, by replacing it with a different message a'_{i-1}, generated by himself and sent to the verifier. The goal of the adversary is to fool the verifier into accepting the subsequent fake authentication message a'_i, which is randomly chosen in D_I and is related to the modified message a'_{i-1}, which is set to be equal to $f_I(a'_i)$. It is easy to see that such an attack is unsuccessful, since the verifier does not accept the modified message a'_{i-1}. Indeed the verification algorithm for the message a'_{i-1} outputs 0, since $f_I(a'_{i-1}) \neq a_{i-2}$. The last condition holds since a_{i-2} admits a unique inverse, being f a permutation.

In the following we formally prove that the OWPF-EAC is secure against active static attacks.

Theorem 5.6.1. *If $\Pi = (Gen, Samp, f)$ is a one-way permutation family, then the one-message unilateral entity authentication scheme $\hat{\Pi}$ obtained by the OWPF-EAC is secure in the sense of* IMP-A-ST.

Proof. Assume by contradiction that the one-message unilateral entity authentication scheme $\hat{\Pi} = (\hat{Gen}, \hat{Auth}, \hat{Ver})$ obtained by the OWPF-EAC is not secure in the sense of IMP-A-ST. Thus, for some pair of indices i and j such that $2 \leq i \leq t$ and $1 \leq j \leq i$, there exists a polynomial-time adversary $\mathtt{ACT-STAT}_{i,j}$ whose advantage $\mathbf{Adv}^{\mathbf{Imp}}_{\mathtt{ACT-STAT}_{i,j},\hat{\Pi}}(1^\tau)$ in the impersonation experiment $\mathbf{Imp}_{\mathtt{ACT-STAT}_{i,j},\hat{\Pi}}(1^\tau)$ is non-negligible. We will show how to construct a polynomial-time adversary A which, on input a set of parameters I generated by $Gen(1^\tau)$ and a uniformly distributed value $y \in D_I$, uses the adversary $\mathtt{ACT-STAT}_{i,j}$ to compute a value x' such that $f_I(x') = y$.
The adversary A, on inputs (I, y), generates the inputs for the adversary $\mathtt{ACT-STAT}_{i,j}$ as shown in the following:

- A sets *pub* to be equal to the pair $(I, f_I^{i-1}(y))$;

- A constructs the sequence $hist_{i-1} = (a_1, a_2, \ldots, a_{i-1})$ by setting a_{i-1} to be equal to the challenge value y, whereas, a_ℓ is set to be equal to $f_I^{i-\ell-1}(y)$, for any $1 \leq \ell \leq i-2$. Notice that the values $a_1, \ldots, a_{i-2}$ are easy to compute, since they require iterative computations of the permutation f_I.

The adversary $\texttt{ACT} - \texttt{STAT}_{i,j}$, on input the tuple $(i, j, pub, hist_{i-1})$, outputs the modified messages $a'_j, \ldots, a'_{i-1}$, along with the fake message a'_i. Such message satisfy $f_I(a'_j) = a_{j-1}$ and $f_I(a'_{\ell+1}) = a'_\ell$, for any $\ell = j, \ldots, i-1$. Since it also holds that $a_{j-1} = f_I^{i-j}(y)$, this implies that $a'_\ell = f_I^{i-\ell-1}(y)$, for any $\ell = j+1, \ldots, i$. Thus, the message a'_i satisfies $a'_i = f_I^{-1}(y)$. Therefore, adversary A outputs the value a'_i.

It is easy to see that the view of adversary $\texttt{ACT} - \texttt{STAT}_{i,j}$, when run as a subroutine by A, is distributed identically as in the impersonation experiment $\mathbf{Imp}_{\texttt{ACT}-\texttt{STAT}_{i,j},\hat{\Pi}}(1^\tau)$. Indeed the second element of the public information *pub* is a uniformly distributed element in D_I, since it is obtained by iterating the permutation f_I, starting from the value y, which is uniformly distributed in D_I. Moreover, for the same reasons, each element of the $hist_{i-1}$ sequence is also a uniformly distributed element in D_I. Since the advantage of adversary $\texttt{ACT} - \texttt{STAT}_{i,j}$ in the impersonation experiment $\mathbf{Imp}_{\texttt{ACT}-\texttt{STAT}_{i,j},\hat{\Pi}}(1^\tau)$ is non-negligible, it follows that also the advantage of A in the inversion experiment $\mathbf{Invert}_{A,\Pi}(1^\tau)$ is non-negligible. Contradiction. □

5.7 Security of TPF-EAC

In the following we will show that the TPF-EAC is secure against active static attacks. The proof of the next result follows the same lines as that of Theorem 5.6.1.

Theorem 5.7.1. *If $\Pi = (Gen, Samp, f, Inv)$ is a trapdoor permutation family, then the one-message unilateral entity authentication scheme $\hat{\Pi}$ obtained by the TPF-EAC is secure in the sense of* IMP-A-ST.

Chapter 6

Concluding Remarks and Future Work

In this thesis we have considered Entity Authentication.

First, we have proposed the *first theoretical characterization* for one-message unilateral entity authentication schemes, where the prover and the verifier do not share any secret information, such as a password, in advance.

Then, we have proposed three constructions for one-message unilateral entity authentication schemes, based on three different cryptographic primitives. The first construction uses as a building block a family of one-way functions, whereas, the second one is based on a family of one-way permutations. Finally, the third construction is based on a family of trapdoor permutations.

Afterwards, we have formalized the security requirements for one-message unilateral entity authentication schemes, by considering two different kinds of adversaries, *passive* and *active*, where anyone might be *static* or *adaptive*, respectively. The passive adversary is able to passively eavesdrop on multiple honest executions of the schemes and whose goal is to fool the verifier into accepting an authentication message which has not been generated by the prover. The active one is a more powerful kind of adversary because it is also allowed to perform other actions, such as modifying authentication messages generated by the prover before they reach

their destination.

We have shown that security against passive adaptive attacks is (polynomially) equivalent to security against passive static attacks. Further, we have shown that security against active adaptive attacks is (polynomially) equivalent to security against active static attacks. Moreover, we have proved that security against passive attacks is not (polynomially) equivalent to security against active ones.

So, we have formally proved that our first construction is secure against passive static adversaries and we have shown that it is insecure against active ones; indeed, we have shown a simple attack which can be carried out by them. At last, we have formally proved that the other two constructions we have proposed are secure against active static adversaries, the more powerful ones.

We plan to further investigate the power of active adversaries in future work. More in details, we would like to design one-message unilateral entity authentication schemes which provide security against active adversaries while relying on minimal computational assumptions.

Bibliography

[1] R. L. Rivest, "Cryptography," in *Handbook of Theoretical Computer Science - Algorithms and Complexity (vol. A)*, J. van Leeuwen, Ed. MIT Press, 1990, ch. 13, pp. 617–755.

[2] O. Goldreich, "Cryptography and cryptographic protocols," *Distributed Computing*, vol. 16, no. 2, pp. 177–199, 2003.

[3] C. E. Shannon, "A mathematical theory of communication," *The Bell System Technical Journal*, vol. 27, no. 3, pp. 379–423, 1948.

[4] W. Diffie and M. E. Hellman, "New directions in cryptography," *IEEE Transactions on Information Theory*, vol. 22, pp. 644–654, 1976.

[5] A. J. Menezes, S. A. Vanstone, and P. C. V. Oorschot, *Handbook of Applied Cryptography*, 1st ed. Boca Raton, FL, USA: CRC Press, Inc., 1996.

[6] J. Katz and Y. Lindell, *Introduction to Modern Cryptography*, 2nd ed. Chapman & Hall/CRC, 2014.

[7] D. R. Stinson and M. B. Paterson, *Cryptography: Theory and Practice*, 4th ed. Boca Raton, FL, USA: CRC Press, Inc., 2018.

[8] A. De Santis, M. Flores, and B. Masucci, "One-message unilateral entity authentication schemes," in *Proceedings of the 12th International Conference on Availability, Reliability and Security*, ser. ARES '17. ACM, 2017, pp. 25:1–25:6.

[9] A. De Santis, M. Flores, and B. Masucci, "Proving the security of entity authentication schemes," 2018, preprint.

[10] N. Smart, *Cryptography: An Introduction*, 3rd ed. McGraw-Hill College, 2004.

[11] M. Bellare and P. Rogaway, *Introduction to Modern Cryptography.* ©Mihir Bellare and Phillip Rogaway, 2005. [Online]. Available: http://web.cs.ucdavis.edu/~rogaway/classes/227/spring05/book/main.pdf

[12] R. A. Mollin, *An Introduction to Cryptography*, 2nd ed. Chapman & Hall/CRC, 2006.

[13] R. Oppliger, *Contemporary Cryptography*, 2nd ed. Artech House Publishers, 2011.

[14] H. Delfs and H. Knebl, *Introduction to Cryptography*, 1st ed. Springer, 2002.

[15] M. Bellare, "A note on negligible functions," *Journal of Cryptology*, vol. 15, no. 4, pp. 271–284, 2002.

[16] S. Goldwasser and S. Micali, "Probabilistic encryption," *Journal of Computer and System Sciences*, vol. 28, no. 2, pp. 270–299, 1984.

[17] A. W. Dent, "Fundamental problems in provable security and cryptography," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 364, no. 1849, pp. 3215–3230, 2006.

[18] C. E. Shannon, "Communication theory of secrecy systems," *Bell System Technical Journal*, vol. 28, no. 4, pp. 656–715, 1949.

[19] A. C. Yao, "Theory and applications of trapdoor functions," in *Proceedings of the 23rd Annual ACM Symposium on Foundations of Computer Science*, 1982, pp. 80–91.

[20] J. Rompel, "One-way functions are necessary and sufficient for secure signatures," in *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing*, 1990, pp. 387–394.

[21] J. Hastad, R. Impagliazzo, L. A. Levin, and M. Luby, "A pseudorandom generator from any one-way function," *SIAM Journal on Computing*, vol. 13, pp. 1364–1396, 1999.

[22] M. Blum and S. Micali, "How to generate cryptographically strong sequences of pseudorandom bits," *SIAM J. on Computing*, vol. 13, pp. 850–864, 1984.

[23] M. Naor and M. Yung, "Universal one-way hash functions and their cryptographic applications," in *Proceedings of the 21st Annual ACM Symposium on Theory of Computing*, 1989, pp. 33–43.

[24] R. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Communications of the ACM*, vol. 21, no. 2, pp. 120–126, 1978.

[25] M. O. Rabin, "Digitalized signatures as intractable as factorization," *Technical Report*, vol. TR-212, no. MIT/LCS, 1979.

[26] N. Bitansky, O. Paneth, and D. Wichs, "Perfect structure on the edge of chaos - Trapdoor permutations from indistinguishability obfuscation," in *Theory of Cryptography 2016, LNCS*, vol. 9562, 2016, pp. 474–502.

[27] M. Bellare and P. Rogaway, "Random oracles are practical: A paradigm for designing efficient protocols," in *CCS '93, Proceedings of the 1st ACM Conference on Computer and Communications Security*, 1993, pp. 62–73.

[28] L. Lamport, "Password authentication with insecure communication," *Communications of the ACM*, vol. 24, no. 11, pp. 770–772, 1981.

[29] N. Haller, "The s/key one-time password system," in *Proceedings of the Internet Society Symposium on Network and Distributed Systems*, 1994, pp. 151–157.

[30] N. Haller, C. Metz, P. Nesser, and M. Straw, "A one-time password system," *RFC*, no. 2289, 1998.

[31] D. L. McDonald, R. J. Atkinson, and C. Metz, "One time passwords in everything (opie): Experiences with building and using stronger authentication," in *Proceedings of the 5th Conference on USENIX UNIX Security Symposium - Volume 5*, ser. SSYM'95. Berkeley, CA, USA: USENIX Association, 1995, pp. 177–188.

[32] R. Joyce and G. Gupta, "Identity authentication based on keystroke latencies," *Communications of the ACM*, vol. 33, no. 2, pp. 168–176, 1990.

[33] M. Sandirigama and A. Shimizu, "Simple and secure password authentication protocol (sas)," *IEICE Transactions on Communications*, vol. 83, no. 6, pp. 1363–1365, 2000.

[34] H.-C. Kim, H.-W. Lee, K.-S. Lee, and M.-S. Jun, "A design of one-time password mechanism using public key infrastructure," in *Networked Computing and Advanced Information Management, 2008. NCM'08. Fourth International Conference on*, vol. 1. IEEE, 2008, pp. 18–24.

[35] D. M'Raïhi, S. Machani, M. Pei, and J. Rydell, "Totp: Time-based one-time password algorithm," *RFC*, no. 6238, 2011.

[36] C.-C. Chang and T.-C. Wu, "Remote password authentication with smart cards," *IEE Proceedings E-Computers and Digital Techniques*, vol. 138, no. 3, pp. 165–168, 1991.

[37] W.-H. Yang and S.-P. Shieh, "Password authentication schemes with smart cards," *Computers & Security*, vol. 18, no. 8, pp. 727–733, 1999.

[38] M.-S. Hwang and L.-H. Li, "A new remote user authentication scheme using smart cards," *IEEE Transactions on Consumer Electronics*, vol. 46, no. 1, pp. 28–30, 2000.

[39] Y. Tzu-Chang, S. Hsiao-Yun, and J.-J. Hwang, "A secure one-time password authentication scheme using smart cards," *IEICE Transactions on Communications*, vol. 85, no. 11, pp. 2515–2518, 2002.

[40] I.-E. Liao, C.-C. Lee, and M.-S. Hwang, "A password authentication scheme over insecure networks," *Journal of Computer and System Sciences*, vol. 72, no. 4, pp. 727–740, 2006.

[41] J. Xu, W.-T. Zhu, and D.-G. Feng, "An improved smart card based password authentication scheme with provable security," *Computer Standards & Interfaces*, vol. 31, no. 4, pp. 723–728, 2009.

[42] B. Vaidya, J. H. Park, S.-S. Yeo, and J. J. P. C. Rodrigues, "Robust one-time password authentication scheme using smart card for home network environment," *Computer Communications*, vol. 34, no. 3, pp. 326–336, 2011.

[43] S. Wu, Y. Zhu, and Q. Pu, "Robust smart-cards-based user authentication scheme with user anonymity," *Security and Communication Networks*, vol. 5, no. 2, pp. 236–248, 2012.

[44] B. Barak, O. Goldreich, R. Impagliazzo, S. Rudich, A. Sahai, S. P. Vadhan, and K. Yang, "On the (im)possibility of obfuscating programs," *Journal of the ACM*, vol. 59, no. 2, 2012.

[45] S. Garg, C. Gentry, S. Halevi, M. Raykova, A. Sahai, and B. Waters, "Candidate indistinguishability obfuscation and functional encryption for all circuits," in *Proceedings of the 54th Annual IEEE Symposium on Foundations of Computer Science*, 2013, pp. 40–49.