"RFID authentication and time-memory trade-offs"

Carpent, Xavier

Abstract

RFID is a technology that allows identification and authentication of objects or persons through the use of wireless communication between tags and readers. RFID Tags are small devices that are comprised of an antenna (for receiving/ transmitting data) and a chip. Although exceptions exist (e.g. passports, etc.), tags are generally inexpensive and moderately powerful in terms of computation. Due to the high requirements (secure authentication, respect of privacy, speed of authentication, etc.) and specific constraints (asymetric system, inexpensive tags, wireless communication, etc.), there are many challenges in RFID security. Mostly two have been studied in this thesis: ultralightweight authentication (authentication protocols dedicated to extremely low-end tags where classical crypto is deemed too expensive) and the complexity/privacy tradeoff (protecting privacy makes the task of readers very time-consuming). In the former, our results are mostly cryptanalytic (almost all ultralig...

RFID Authentication and Time-Memory Trade-offs

Xavier Carpent

*Thesis submitted in partial fulfillment of the requirements
for the Degree of Doctor in Engineering Sciences*

March 18, 2015

Institute of Information and Communication Technologies,
Electronics and Applied Mathematics (ICTEAM)

Louvain School of Engineering (EPL)

Université catholique de Louvain (UCL)

Louvain-la-Neuve
Belgium

Abstract

Two areas have been explored during this thesis: RFID authentication, and cryptanalytic time-memory tradeoffs.

Radio Frequency IDentification (or RFID) is a technology that allows identification and authentication of objects or persons through the use of wireless communication between tags and readers. RFID Tags are small devices that are comprised of an antenna (for receiving/transmitting data) and a chip. Although exceptions exist (e.g. passports, etc.), tags are generally inexpensive and moderately powerful in terms of computation, and most of the time rely on the electro-magnetic field provided by the reader for energy supply. Readers on the other hand are more expensive devices that have a computational power comparable to a PC. The RFID technology is very widespread nowadays, and it continues to grow rapidly. However, due to the high requirements (secure authentication, respect of privacy, speed of authentication, etc.) and specific constraints (asymmetric system, inexpensive tags, wireless communication, etc.), there are many challenges regarding its security. Mostly two have been studied in this thesis: ultralightweight authentication (authentication protocols dedicated to extremely low-end tags where classical crypto is deemed too expensive) and the complexity/privacy tradeoff (protecting privacy makes the task of readers very time-consuming). In the former, our results are mostly cryptanalytic, and in the latter where, it seems, no perfect solution exists, our results are mainly analytical and comparative (the OSK/AO protocol in particular achieves good privacy protection with reasonable complexity, and uses cryptanalytic time-memory tradeoffs that is the focus of the second part of the thesis).

A cryptanalytic time-memory tradeoff is a generic tool to carry out brute force search for the pre-image of a one-way function efficiently. It forms a compromise between the online exhaustive search (no precomputation, searching through all the possibilities) and the lookup table (all possibilities precomputed and stored, and then looking up when needed) approaches. In the former, no precomputation or memory is required, but the search is expensive, and in the latter, precomputation and storage are expensive, but the search is nearly instant. In time-memory tradeoffs, precomputation is expensive, but both storage and online search time are reasonable. In this thesis, ways to improve the performance of existing techniques have been explored, among which fingerprints (in which stored information is slightly different than in classical time-memory tradeoffs, which results in a speedup in the online phase), storage optimization (which reduce the storage of time-memory tradeoffs), and interleaving (which is a way to accommodate the technique to non-uniformly distributed input).

Contents

List of Publications

[20] Gildas Avoine and Xavier Carpent and Benjamin Martin. Strong Authentication and Strong Integrity (SASI) is not that Strong. In S.B. Ors Yalcin, editor, *Workshop on RFID Security – RFIDSec'10*, volume 6370 of *Lecture Notes in Computer Science*, pages 50–64, Istanbul, Turkey, June 2010. Springer.

[21] Gildas Avoine, Xavier Carpent, and Benjamin Martin. Privacy-friendly synchronized ultralightweight authentication protocols in the storm. *Journal of Network and Computer Applications*, 35(2):826–843, February 2012.

[16] Gildas Avoine and Xavier Carpent. Yet another ultralightweight authentication protocol that is broken. In *Workshop on RFID Security – RFIDSec'12*, Nijmegen, Netherlands, June 2012.

[12] Gildas Avoine, Muhammed Ali Bingol, Xavier Carpent, and Suleyman Kardas. Deploying OSK on low-resource mobile devices. In *Workshop on RFID Security – RFIDSec'13*, Graz, Austria, July 2013.

[13] Gildas Avoine, Muhammed Ali Bingol, Xavier Carpent, and Siddika Berna Ors Yalcin. Privacy-friendly authentication in RFID systems: On sub-linear protocols based on symmetric-key cryptography. *IEEE Transactions on Mobile Computing*, 12(10):2037–2049, October 2013.

[17] Gildas Avoine and Xavier Carpent. Optimal Storage for Rainbow Tables. In Sukwoo Kim and Seok-Yeol Kang, editors, *International Conference on Information Security and Cryptology – ICISC 2013*, Seoul, Korea, November 2013.

[19] Gildas Avoine and Xavier Carpent and Julio Hernandez-Castro. Pitfalls in Ultralightweight Authentication Protocol Designs. *IEEE Transactions on Mobile Computing.* (submitted)

[14] Gildas Avoine and Adrien Bourgeois and Xavier Carpent. Analysis of Rainbow Tables with Fingerprints. *Financial Cryptography and Data Security – FC'15.* (submitted)

[18] Gildas Avoine and Xavier Carpent. Interleaving Cryptanalytic Time-memory Trade-offs on Non-Uniform Distributions. *ACM Symposium on Information, Computer and Communication Security – ASIACCS'15.* (submitted)

Chapter 1

Introduction

This first chapter is an introduction to the two topics addressed in this thesis: RFID authentication and cryptanalytic time-memory trade-offs. Section 1.1 presents an introduction to Radio Frequency Identification (RFID), its characteristics, and its challenges. Section 1.3 presents a short introduction on Time-Memory Trade-offs (TMTO). Sections 1.2 and 1.4 present what was investigated and what was achieved in this thesis in the two subjects. Finally, Section 1.5 presents the plan of the rest of this thesis.

1.1 RFID in a Nutshell

RFID is a technology that allows small inexpensive devices, the RFID *tags*, to communicate wirelessly using radio waves with RFID *readers*, for the purpose of identification of persons and objects carrying them. The RFID tag consists of a chip for computation and storage of information, and an antenna used for communicating with a reader. Some high-end RFID tags have a battery and decent computational capabilities (*active* tags), but the vast majority of tags are very cheap and limited, and rely on the electromagnetic field of the reader for power (*passive* tags). The field is modulated by the tag's chip and reflected to the reader for identification.

The RFID technology has been around since the 70's [61], but has started to be truly ubiquitous in the early 2000's (with the creation of the Auto-ID center and the publication of the EPCGlobal [77] standard). Nowadays, RFID is used in a variety of industrial and every-day applications (See Figure 1.1): goods tracking, animal identification, access control, public transportation, wireless payment, passports, etc. All things considered and despite its numerous benefits, the deployment of RFID has been relatively slow. Some issues include practicability (although not impossible, it is hard for RFID to work with water and metal), cost (to put it bluntly, it is impractical to rely on tags that cost as much or nearly as much as the items they are attached to), and security. This last point in particular, along with other considerations, is the focus of the first part of this thesis (Chapters 2 and 3).

1.1.1 System Model

In the rest of this manuscript, the following model is used for describing identification and authentication protocols. An RFID system consists of three main entities [187]: a tag, a reader, and a back-end system.

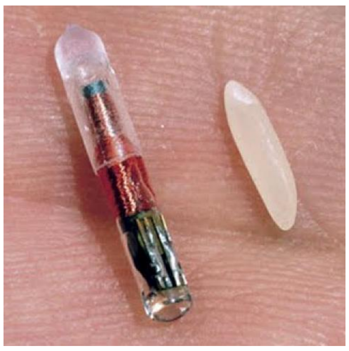
The *tag* (or transponder) is attached to an object or person, and is uniquely identifiable in the RFID system. Characteristics such as computational power, storage capabilities, or communication distance, strongly vary with price and usage, but are usually very limited.

A *reader* (or transceiver) communicates with RFID tags in order to perform identification and authentication. As stated earlier, they provide most of the power used by a tag, and usually have much larger computational capabilities. Depending on the protocol used, they may perform heavy computation, such as cryptographic calculation, on behalf of the tag.

The *back-end system* (or database) stores records associated with the content of the tags. In the physical world, it is usually connected to many readers in an RFID system. However most

(a) RFID tag used for animal identification [credits: Sandstein/Creative Commons].

(b) RFID tag used for product tracking [public domain].


(c) A very small RFID tag [public domain].

Université catholique de Louvain
UCL
CARTE D'ACCÈS
Catherine Defrance
N° carte 1234567 | N° FGS 12345678
(d) RFID tag inside an ID-1 card for access control at the UCL [credits: `http://www.uclouvain.be/`].

(e) An RFID reader for Oyster cards, used in public transportation in London [credits: Tom Page/Creative Commons].

(f) A Belgian biometric passport with an RFID tag inside [public domain].

Figure 1.1: Pictures of RFID tags and readers in different settings.

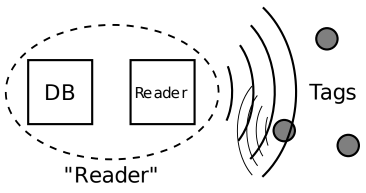


Figure 1.2: Architecture of an RFID system, as modeled in this thesis.

analyses assume that the communication channel between a reader and the back-end system is *secure*, to the point that the reader and the database are considered only one entity.

Figure 1.2 is a schematized depiction of an RFID system, as modeled in this thesis.

In the following, a protocol is defined as being a set of rules to exchange data between a reader and a tag. In *identification protocols*, a reader merely determines the identity of a tag (and its carrier), whereas in *authentication protocols*, a tag proves its identity to a reader (and vice-versa in *mutual* authentication protocols) using cryptographic tools. The former are used in applications where identity theft is irrelevant such as supply-chain tracking and animal identification. The latter (which usually require more potent tags) are used in applications requiring security such as passports, public transportation, wireless payments or access control.

1.1.2 Threats in RFID

In the context of the analysis of authentication protocols, the attacker may be of two different types: an active or a passive attacker. A passive attacker, or *eavesdropper*, may only witness the data transmitted between a tag and a reader. An active one may, in addition, tamper with the data, or interrupt its transmission. In some situations, it is also relevant to consider adversaries able to "compromise" a tag in order to reveal its secrets. This is however considered relatively difficult to achieve, and only applicable when the benefit is worth the effort (see Chapter 3).

Attacks against RFID systems can be divided roughly into four categories: denial of service, impersonation, information leakage, and malicious traceability. Other threat classifications exist [62, 175, 139], but these attack types constitute a solid summary of the capabilities and goals of an attacker.

Denial of Service (DoS) is an attack that occurs when an adversary attempts to prevent the application to function properly. In the framework of RFID, besides obvious physical harm to readers, tags, cables and other assets, this may be achieved using various techniques such as using blocker tags [111], using RFID jammers and zappers, introducing electromagnetic noise on the channel, etc. Electronic DoS attacks are extremely difficult to avoid, but are usually not taken into account in the security analysis of authentication protocols. This is due to the fact that they are often applicable regardless of protocol details. Some types of DoS attacks are due to weaknesses in the protocol design, though. For instance, desynchronization attacks in stateful protocols make further authentication of a tag-reader pair impossible (see Section 2.3.6). In this specific case, other attacks are sometimes possible after a tag-reader pair has been desynchronized. This kind of DoS attacks must be taken into account in the analysis of authentication protocols.

Impersonation consists in being authenticated as someone else without being authorized to do so. This can be achieved by replay attacks, for instance, or any other weakness in the protocol, including those that allow an attacker to acquire knowledge of the secret of a tag (also known as *key recovery*). The attacker can then disguise an expensive product into a cheap one, or gain access to restricted areas for instance. Relay attacks may also make an attacker impersonate a tag, but in security analyses done in this thesis, we are not concerned with such attacks. These require very specific protocols named *distance bounding* protocols and are the focus of other works (Hancke and Kuhn's solution [94] is an early and famous example).

Information leakage is a scenario in which an attacker gains information deemed private on the product or the tag holder. For instance, an attacker could get sensitive data of a user, such as his social security number, his address, etc. The attacker could for instance acquire the identity of a user's personal belongings, in order to spot a potential robbery victim. Other possible reasons are political, industrial, or personal espionage, blackmailing, etc. Information leakage is usually prevented by mapping a real product or person ID in the database to an anonymous ID in the tag, which only the database can pair.

Malicious traceability has somewhat less dangerous consequences, but it is also the hardest problem to deal with. It consists in tracing a tag (and its holder) and therefore violating the user's location privacy (in space and time). This can be performed if the attacker is able to find a correlation between authentication sessions of a tag. This is especially hard to prevent, because the response of the tag must change with each session and have negligible correlation with previous (and future) responses. Many privacy models have been proposed [113, 184, 181, 99, 24], but two characteristics are usually required: *indistinguishability* [149] (or *untraceability* [9]), and *forward security* [149] (or *forward untraceability* [9]). Untraceability is the fact that an adversary is not able to tell two tags apart, given a set of authentication sessions of these tags. Forward untraceability is the fact that an adversary acquiring the secret of a tag is not able to trace past authentication sessions of that tag.

1.1.3 Challenges in RFID Authentication

Tag Cost

As mentioned earlier, tag cost plays a critical role in the massive deployment of RFID. Although some sensitive applications use relatively expensive tags (e.g. passports), some others require tags to be extremely cheap (e.g. tracking of relatively cheap products, disposable tags, etc.) but still provide some acceptable level of security. The main drivers of cost in tags is the presence of a battery, power requirements, and the size of a chip (measured in gate equivalents). Providing security and ensuring privacy for such constrained devices is a real challenge, and is the focus of a significant part of the research community.

Privacy

The recent advent of ubiquitous technologies has raised an important concern for citizens: the need to protect their privacy. So far, this wish was mostly ignored by industry, but national and international regulation authorities, as the European Commission recently published some guidelines to enforce customers' privacy in RFID systems: "Privacy by design" is the way to be followed as stated in EC Recommendation of 12.5.2009. Research on privacy is an active domain but there is still a wide gap between theory and everyday life's applications. Filling this gap requires academia to design protocols and algorithms that fit the real life constraints.

In order to analyze the privacy of authentication protocols, many models are available in the literature [113, 184, 181, 99, 24]. In this thesis, most of the discussions regarding malicious traceability attacks fit in the model of Juels and Weis [113], which is based on Avoine's seminal work [9]. This model is relatively simple and intuitive, and although it lacks the power of expressivity of more evolved ones (such as Vaudenay's [184] or Avoine, Coisel and Martin's [24]), it is sufficient to understand the attacks and protocol characteristics discussed in this manuscript[1].

The model of Juels and Weis defines privacy with the following game. First, an adversary interacts with the system (querying tags, readers, eavesdropping communications, etc.). She then chooses two tags $\mathcal{T}_0$ and $\mathcal{T}_1$. A bit $b \in \{0, 1\}$ is chosen randomly and unknowingly to her, and she interacts with the system again except $\mathcal{T}_b$. After a while, she must guess whether $b = 0$ or 1. The system is deemed private if there exists no adversary capable of winning with a probability non-negligibly higher than 1/2. The more powerful property of *forward privacy* (defined in [113] as well, and intuitively introduced in [149]) is ensured if an adversary, having acquired the secret of a tag, is not able to trace past authentication sessions of that tag.

Speed of Authentication

Most RFID applications require the authentication of a tag to be done relatively quickly. A fluid and seamless flow of customers in mass public transportation is a typical example of this necessity. This requires the computation on both sides to be relatively quick, and the communication in both directions to be relatively short. Although the readers are much more powerful than tags, depending on the specific protocol readers may need to perform much more computation. Such applications also require authentication protocols that scale well when there is a large number of tags registered in the system.

RFID Authentication protocols usually start with an identification phase, where the reader acquires the identity of a tag. It then looks up secret data linked with it, and uses that data to verify the claim of the prover[2], or use the data provided by the prover to both identify and authenticate it at the same time. If privacy is a requirement, and if only symmetric-key encryption is used, the identification phase may be very time-consuming, especially in large systems.

1.2 Results on RFID Authentication

Work on RFID authentication described in this thesis essentially focused on two problematics.

The first is the analysis of so-called "ultralightweight protocols." As mentioned in Section 1.1.3, there is a need for protocols that provide secure and privacy-friendly authentication in systems where tags have extremely little computational power. While symmetric-key cryptography is usually considered acceptable for tags in the intermediate price range, and the more expensive public-key cryptography is only usable on high-end tags such as passports, the lower tier of tags have to rely on something simpler. In response to that, many protocols have been proposed to provide authentication using only a handful of very basic operations (such as XORs, modular additions, rotations, etc.). These protocols, dubbed "ultralightweight," are however relatively weak, and none of them today can be regarded as sufficiently secure to be used in practice.

[1]In particular, the model of Juels and Weis is not capable of pinpointing precise levels of privacy. However, no formal proof of privacy is done in this thesis.

[2]In the context of RFID, the terminology used is "tags" and "readers", but authentication protocols are not specific to the technology, and the terminology "prover" (the one that provides proof, the tag) and "verifier' (the one that verifies and authenticates, the reader) is used in broader contexts. In this thesis, the two terminologies are used interchangeably.

Many of these protocols have been analyzed in this thesis, and some of them broken. The article [20] contains the cryptanalysis of SASI, an early ultralightweight protocol notable for being the first of its kind to use rotations. A survey [21] has then been done on ultralightweight authentication protocols of the time as well as attacks on them, and an application of the attack in [20] to the Yeh-Lo-Winata protocol. [16] presents the cryptanalysis of several ultralightweight authentication protocols. Finally, [19] is a discussion on the situation in the field and typical design weaknesses of ultralightweight protocols, and it advocates a change in the way these protocols are designed. The important results are presented in Chapter 2.

Many of these protocols have been analyzed in this thesis, and some of them broken. A first article [20] contains the cryptanalysis of SASI, an early ultralightweight protocol notable for being the first of its kind to use rotations. A survey [21] has then been done on ultralightweight authentication protocols of the time as well as attacks on them, and it presents an application of the attack in [20] to the Yeh-Lo-Winata protocol. Several ultralightweight authentication protocols were then cryptanalyzed in [16]. Finally, a discussion on the situation in the field and typical design weaknesses of ultralightweight protocols was done in [19], advocating a change in the way these protocols are designed. The important results are presented in Chapter 2.

The second problematic is the scalability of privacy-friendly authentication protocols in systems with many tags. As mentioned in Section 1.1.3, respecting privacy and ensuring a quick identification of tags in the reader are two somewhat contradictory goals. Indeed, if privacy is a concern, no information sent by the tag should be correlated with its identity from an attacker's point of view. On the other hand, such a restriction hinders the identification of the tag on the reader side. Designing a protocol that supports both aspects is quite challenging and many interesting partial solutions have been proposed.

Some of these solutions are analyzed in this work. A survey and comparison of existing solutions was first done in [13], along with attacks on some of them. The best solution according to this study, OSK/AO, has then been further analyzed and implemented in practice in [12]. This protocol uses cryptanalytic time-memory trade-offs to accelerate the authentication, a topic that is the focus of the second part of this thesis. The problematic of the complexity of the identification phase of private authentication protocols is studied in Chapter 3, and the OSK/AO protocol in Chapter 4

1.3 Introduction to Cryptanalytic Time-Memory Trade-offs

Cryptanalytic time-memory trade-offs are a tool to make brute-force attacks on hash functions or ciphers more practical. As their name suggest, they consist in a trade-off between online time and required memory to invert a one-way function. They were first introduced by Hellman in 1980 [97], and were later refined and improved. Time-memory trade-offs have been used in many practical attacks such as against A5/1 (used for GSM communications) in 2000 [48], or other stream ciphers like LILI-128 in 2002 [169].

1.3.1 Motivation

A fundamental problem in cryptanalysis is finding the preimage of a given output of a one-way function. A simple method is applying the function to all possible inputs until finding the expected value. Such an exhaustive search requires N operations in the worst case to find a preimage, where N is the size of the input space. This becomes impractical when N is large.

The other extreme is to first construct a look-up table including all the preimage values. Afterwards, finding a preimage is done via a table look-up operation which requires a negligible amount of time. The precomputation process however requires an effort equal to an exhaustive search, but is to be performed only once. Although this method is quite fast during the online search phase, it may require prohibitively large amounts of memory for large problems. The comparison of exhaustive search and exhaustive storage methods is depicted in Table 1.1.

Time-memory trade-offs are an intermediate solution to this problem. They consist in an offline precomputation phase, and an online search phase, and require some memory. The more memory is dedicated to the trade-off, the faster the search phase. The memory required is typically much smaller than for exhaustive storage, and the online phase is on average typically much faster than

Table 1.1: Comparison of exhaustive search and table look-up methods (average case, cryptographic operations only).

	Exhaustive search	Exhaustive storage
Precomputation	0	N
Average Online computation	$N/2$	0
Memory (storage)	0	N

for exhaustive search. The precomputation phase however is more expensive than for the exhaustive storage solution. A TMTO is relevant when the online phase is performed many times, when the precomputation phase is carried out by a more powerful entity than the online phase, or when the window of opportunity for the attack is short, but there is a sizeable preparation time ("Lunchtime attack").

1.3.2 Variants and Adaptations

Arguably, the most important of the algorithmic improvements to the Hellman method [97] is the rainbow tables, introduced by Oechslin in [147]. Rainbow tables have been illustrated by the very efficient cracking of Windows LM Hash passwords in 2003 [147] or Unix passwords (using FPGA) in 2005 [138]. Although comparing different trade-off algorithms is not trivial, and the results strongly depend on the parameters, rainbow tables have been shown to be superior to other trade-off algorithms in most situations [121]. Distinguished points (attributed to Rivest in 1982 [70]) is another improvement over the Hellman method, but evidence shows they are slower in practice than rainbow tables [147, 121].

Although they were not investigated in this thesis, these variations are also relevant to the study of time-memory trade-offs.

Hellman's technique is designed to invert random functions. Fiat and Naor provide in [82] a construction for inverting any function, at the price of a less efficient trade-off. De, Trevisan and Tulsiani propose in [68] a similar construction for inverting any function on a fraction of their input. They also suggest using time-memory trade-offs for distinguishing the output of pseudorandom generators from random.

Time-memory trade-offs have also been applied to stream cipher independently by Babbage in [31] and Golić in [86]. These trade-offs are call time-memory-data trade-offs, because the number of bits of data (the keystream) is another parameter. Biryukov and Shamir improve in [47] the efficiency of this attack on stream ciphers by combining the Babbage/Golić and the Hellman techniques. Finally, Biryukov, Mukhopadhyay and Sarkar generalize these different approaches in [46], and propose a way to use Hellman's technique on block ciphers with multiple data.

In [38], Barkan, Biham, and Shamir showed that the performance of existing time-memory trade-offs can not be improved by more than a logarithmic factor.

1.4 Results on Time-Memory Trade-offs

Cryptanalytic time-memory trade-offs (TMTO) can be improved in many ways. Improving rainbow tables specifically was the focus of this thesis.

The first improvement is the use of *fingerprints*. It is a generalization of two former improvements on rainbow tables, the checkpoints, and the truncated endpoints. This generalization allows to analyze these two improvements jointly, which makes it possible to find optimal configurations in a systematic way. Rainbow tables with fingerprint in optimal configuration are about twice as fast (in the online phase) as classical rainbow tables on typical problem sizes. This technique is described and analyzed in Chapter 5 of this thesis.

The second improvement regards *storage*. Storage is extremely important in time-memory trade-offs, as more efficient storage translates directly into a faster search. This aspect was studied [17] by providing a lower bound on rainbow tables storage, and a new technique called compressed delta encoding was introduced to reach this bound. These results are presented in Chapter 6.

Finally, the third improvement is called *interleaving*, and aims at providing a solution when deadling with non-uniformly distributed input. Passwords (which are a typical target of TMTO) are for instance not chosen randomly by users. Interleaving is described and analyzed in Chapter 7.

1.5 Plan of the Manuscript

Chapter 2 presents the discussion on the ultralightweight protocols, the situation in the field, their weaknesses, and attacks on some of them. Chapter 3 presents an overview of the problematic of scalability for privacy-friendly authentication protocols. Chapter 4 studies one of them, OSK/AO in detail, and its implementation in practice. Chapter 5 is a description and analysis of the fingerprints for the rainbow table method. Chapter 6 presents the discussion on storage in rainbow tables, and introduces and analyses compressed delta encoding. Chapter 7 discusses the interleaving technique. Finally, Chapter 8 concludes this manuscript by summarizing the work considered in this thesis, along with perspectives of future works in the areas that were investigated.

Chapter 2

Ultralightweight Authentication

Articles related to this chapter

[20] Gildas Avoine and Xavier Carpent and Benjamin Martin. Strong Authentication and Strong Integrity (SASI) is not that Strong. In S.B. Ors Yalcin, editor, *Workshop on RFID Security – RFIDSec'10*, volume 6370 of *Lecture Notes in Computer Science*, pages 50–64, Istanbul, Turkey, June 2010. Springer.

[21] Gildas Avoine, Xavier Carpent, and Benjamin Martin. Privacy-friendly synchronized ultralightweight authentication protocols in the storm. *Journal of Network and Computer Applications*, 35(2):826–843, February 2012.

[16] Gildas Avoine and Xavier Carpent. Yet another ultralightweight authentication protocol that is broken. In *Workshop on RFID Security – RFIDSec'12*, Nijmegen, Netherlands, June 2012.

[19] Gildas Avoine and Xavier Carpent and Julio Hernandez-Castro. Pitfalls in Ultralightweight Authentication Protocol Designs. *IEEE Transactions on Mobile Computing.* (submitted)

RFID authentication is a very active and challenging research topic. It has strong requirements such as security of course, but also privacy, and yet very strong constraints such as powerful adversaries and very constrained devices. This last point in particular is the focus of a significant part of the research community, and has spawned hundreds of papers [11]. Indeed, there is a strong incentive to build very cheap RFID tags that still provide some level of security. Such tags can not contain a battery, must make do with very few logic gates, and must be able to complete an authentication in very little time. In response to that, a number of protocols that rely on extremely simple operations have been created. Such protocols have been named "ultralightweight," a term coined by Chien in [64]. They can be broadly defined as protocols that are aimed at requiring about 1.5K GE (gate equivalents) or less, relying on only very simple operations on the tag's side, and willing to accept compromises to offer some security level. This includes, in particular, most of the authentication protocols aimed at passive and inexpensive RFID tags. As one might expect, most of the ultralightweight protocols are usually very weak, and are typically broken very quickly.

In this thesis, work done on ultralightweight protocols was mostly cryptanalytical. This chapter presents a few attacks on these protocols[1], and an analysis of their general weaknesses. Its structure is the following. In Section 2.1, a general description of ultralightweight protocols is given, and a typical protocol, SASI, is presented (along with an attack on it). In Section 2.2, a brief state of the art is presented in the form of statistics. The most typical mistakes found in ultralightweight

[1]Not all the attacks that were found as part of this thesis are presented in this chapter for the sake of brevity. However, the general weaknesses exploited by these attacks are part of the discussions in Sections 2.3 and 2.4

protocols are presented in Section 2.3. In Section 2.4, the problematic of dubious security proofs is mentioned along with reasons why they are usually flawed. In Section 2.5, some recent proposals are discussed, with attacks and comments on them based on the discussions from Sections 2.3 and 2.4. Section 2.6 presents a few facts on general-purpose lightweight building blocks. The chapter is concluded in Section 2.7.

2.1 Ultralightweight Protocols

2.1.1 Tag Capabilities

The cost of tags plays a critical role in the ubiquitous deployment of RFID systems. In order to keep their price low, tags must be very simple and use very little energy. Typical figures are a price of 5 to 10 cents, and about 5–10 K logical gate equivalents (GE's) per tag, among which only 250–3000 are devoted to security [64]. By comparison, best implementations of AES require roughly 3400 gate equivalents [79], and hash functions MD5 and SHA-256 respectively require roughly 8K and 11K logical gate equivalents [52]. It is possible to use classical cryptographic primitives such as block ciphers and hash functions on higher-end tags, but the ultra low-cost class is the focus of this chapter.

These constraints have prompted the community to create authentication protocols that require very little computation on the tag side, often relying on extremely lightweight primitives such as logical AND/OR/XOR, modular addition, rotations, etc.

A typical counter-argument for the need of such protocols is that, with the improvement of technology and the large-scale deployment of RFID, the price of tags (even those capable of solid cryptography) will be driven down. However, the problematic remains whole because there is always going to be a market for cheaper devices.

2.1.2 Attack Model

Ultralightweight authentication protocols must primarily ensure that neither an impersonation attack, nor a key-recovery attack (even partial-recovery) is feasible. Protocol analyses usually consider the Dolev-Yao model [73], namely a model where the adversary is powerful but not almighty: the adversary cannot guess random numbers chosen in a sufficiently large space, she cannot decrypt or create valid ciphertexts without the correct secret, and she cannot retrieve private keys from public information. However, she can easily communicate with both a tag or a reader, eavesdrop communications in both directions, or perform relay attacks (possibly modifying the messages). This last point is often a source of confusion in ultralightweight proposals, which treat active attackers as being more powerful than eavesdroppers, despite some active attacks being in fact not harder than eavesdropping. Skimming (communicating directly with a tag or a reader) and relay attacks are examples of active attacks that are relatively easy to do, possibly more so than eavesdropping (see for instance the `libnfc` library [130]).

Ultralightweight authentication protocols usually also attempt to ensure privacy. The privacy property is commonly defined in authentication protocols as the inability for an adversary to track a prover. Privacy is consequently also called *untraceability* and is represented by an experiment where two interactions prover–verifier are provided to the adversary, and the latter must recognize which one was produced by his target (previously queried – possibly with restrictions – during a learning phase). Several untraceability models exist, with different experiments. The most widely used models are due to Juels and Weis [113], Vaudenay [184], Deursen, Mauw, and Radomirovic [181], Hermans, Pashalidis, Vercauteren, and Preneel [99], and Avoine, Coisel, and Martin [24]. It is today strongly advised to use such a well-established model instead of ad hoc ones. A comparison of existing models was done by Coisel and Martin [65].

Designing a privacy-friendly authentication protocol is a difficult question. While a classical challenge-response protocol (where the identity of the prover is not sent in the clear on the channel) seems to be privacy-friendly at first sight, it suffers from a linear complexity search (see Chapter 3) and does not ensure forward-privacy: if the prover is compromised, the privacy of the past interactions is no longer ensured. The latter issue can be mitigated by updating the key, but desynchronization attacks must be prevented in such a case. Vaudenay actually demonstrated that only public-key cryptography can ensure the highest attainable privacy level [183]. Ultralightweight authentication protocols do not necessarily target the highest privacy level, though.

2.1.3 SASI, a Typical Ultralightweight Protocol

The SASI (Strong Authentication and Strong Integrity) protocol was introduced by Chien in [64]. It is one of the early proposals in the field, and arguably one of the most famous ones[2]. It is also quite representative of the general pattern followed by most other ultralightweight protocols. This section presents a description of the SASI protocol along with the details of an attack on it.

Description and Attacks

SASI, like many in the ultralightweight family, is a stateful mutual authentication protocol. Each tag keeps a state that changes whenever it is successfully authenticated. Reader authentication is therefore necessary for the tag to know when to update its state.

Each tag has a secret static identifier ID, two secret keys K_1 and K_2, and a pseudonym IDS[3]. The pseudonym is used to identify tags efficiently while keeping a certain degree of privacy (see Section 3.2 for a discussion on the subject). The keys K_1 and K_2 as well as the pseudonym IDS are all updated when the tag is authenticated.

The SASI protocol relies on logical OR ($\vee$), logical XOR ($\oplus$), modular addition ($+$), and the rotation $Rot(x, y)$. The latter operation is defined as a circular left-shift of x of $r(y)$ bits, where:

$$r : [0, 2^L - 1] \to [0, L - 1].$$

Several types of rotations can be used with SASI, especially the *modular* rotation, that is $r(y) = y \bmod L$, and *Hamming weight* rotation, with $r(y) = \mathcal{H}(y) \bmod L$, where $\mathcal{H}$ is the Hamming weight function. In the latter, the modulus is there to fold the case where $\mathcal{H}(y) = L$ back to a number in $[0, L - 1]$ (a rotation of L bits is the same as a rotation of 0 bits). Note that r was not precisely defined in [64], and it was pointed out in [173] that the rotation intended to be used in the original version of the protocol is the Hamming-weight one. The modular version was analyzed and attacked in [100].

The protocol definition is as follows. The reader initiates the authentication by sending a *hello* message to the tag, which answers its current index-pseudonym IDS. At that point, the reader uses the index-pseudonym to find an entry in its internal database with ID, K_1 and K_2. It then produces two nonces n_1 and n_2, and computes A, B, and C, and sends these three values to the tag. The values A, B and C are defined as:

$$A = IDS \oplus K_1 \oplus n_1 \tag{S-A}$$
$$B = (IDS \vee K_2) + n_2 \tag{S-B}$$
$$C = (K_1 \oplus \bar{K_2}) + (\bar{K_1} \oplus K_2) \tag{S-C}$$

where $\bar{K_1}$ and $\bar{K_2}$ are intermediate secret values defined as:

$$\bar{K_1} = Rot(K_1 \oplus n_2, K_1)$$
$$\bar{K_2} = Rot(K_2 \oplus n_1, K_2)$$

When the tag receives the message $A||B||C$, it extracts the nonces n_1 and n_2 using:

$$n_1 = A \oplus IDS \oplus K_1$$
$$n_2 = B - (IDS \vee K_2)$$

It then computes $\bar{K_1}$, $\bar{K_2}$, and $\tilde{C} = (K_1 \oplus \bar{K_2}) + (\bar{K_1} \oplus K_2)$. If C and $\tilde{C}$ are equal, the tag authenticates the reader, and computes the message D to send it to the reader:

$$D = (\bar{K_2} + ID) \oplus ((K_1 \oplus K_2) \vee \bar{K_1}). \tag{S-D}$$

[2]As of August 2014, Chien's paper [64] is the most cited one among the list given in Table 2.3

[3]In fact, it also stores backup versions of these values as a defense mechanism against desynchronization attack. The protocol is simplified in that regard in this section for clarity reasons. See 2.3.6 for a discussion on the subject.

Upon reception of D, the reader computes its local version $\tilde{D}$, compares it with D, and if they match, it authenticates the tag.

In the updating stage, the tag updates its key and its index-pseudonym using:

$$\begin{aligned} IDS &= (IDS + ID) \oplus n_2 \oplus \bar{K_1} \\ K_1 &= \bar{K_1} \\ K_2 &= \bar{K_2} \end{aligned} \tag{S-IDS}$$

The update process of the reader is the same.

The protocol is depicted in Figure 2.1.

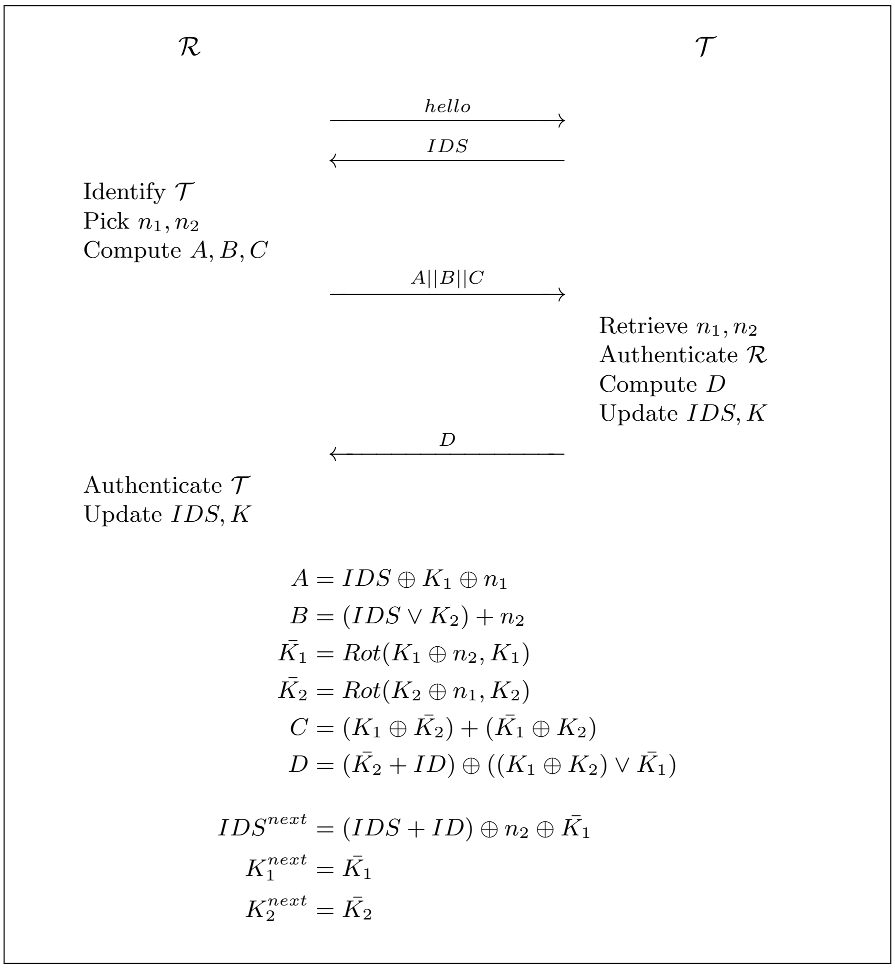

$\mathcal{R}$
$\mathcal{T}$
$hello$
IDS
Identify $\mathcal{T}$
Pick n_1, n_2
Compute A, B, C
$A||B||C$
Retrieve n_1, n_2
Authenticate $\mathcal{R}$
Compute D
Update IDS, K
D
Authenticate $\mathcal{T}$
Update IDS, K
$A = IDS \oplus K_1 \oplus n_1$
$B = (IDS \vee K_2) + n_2$
$\bar{K_1} = Rot(K_1 \oplus n_2, K_1)$
$\bar{K_2} = Rot(K_2 \oplus n_1, K_2)$
$C = (K_1 \oplus \bar{K_2}) + (\bar{K_1} \oplus K_2)$
$D = (\bar{K_2} + ID) \oplus ((K_1 \oplus K_2) \vee \bar{K_1})$
$IDS^{next} = (IDS + ID) \oplus n_2 \oplus \bar{K_1}$
$K_1^{next} = \bar{K_1}$
$K_2^{next} = \bar{K_2}$

Figure 2.1: The SASI protocol

SASI has been the subject of several attacks. The first to be published was a traceability attack by Phan [161]. An elaborate key-recovery attack has also been conceived as part of this thesis (see [20]). The latter is described below.

Preliminary Tools

This section analyzes in detail the mechanics of the addition, in order to have a framework for equations mixing additions and bitwise operations like logical OR and XOR. Two subproblems are then presented that are useful in the attack of SASI, but that might be used for other purposes as well.

Notations and Definitions

In the following, $[x]_i$ denotes the bit at position i in x. In particular, $[x]_0$ is the least significant bit (LSB) of x, and $[x]_{L-1}$ its most significant bit (MSB). By convention, $[x]_i = 0$ when $i > \lceil \log_2(x) \rceil$. The "$i$-th bit of x," refers to $[x]_i$. The following notation is used for the *carry* of the addition, and the *borrow* of the subtraction, respectively:

$\mathcal{C}(a, b, i)$	denotes the carry at bit i of the sum of a and b, and
$\mathcal{B}(a, b, i)$	denotes the borrow at bit i of the difference of a and b.

Using this notation, the result of the addition of numbers a and b at bit i is written as:

$$[a + b]_i = [a]_i \oplus [b]_i \oplus \mathcal{C}(a, b, i - 1). \tag{2.1}$$

Likewise, the difference of two numbers a and b at bit i is written as:

$$[a - b]_i = [a]_i \oplus [b]_i \oplus \mathcal{B}(a, b, i - 1). \tag{2.2}$$

The *carry* of two numbers a and b at bit index i is computed as:

$$\mathcal{C}(a, b, i) = ([a]_i \wedge [b]_i) \vee \big[([a]_i \vee [b]_i) \wedge \mathcal{C}(a, b, i - 1)\big], \tag{2.3}$$

with the convention that $\mathcal{C}(x, y, i) = 0$ if $i < 0$, for any x and y. Indeed, there is a carry at bit i either if the bits i of both operands are 1, or if at least one of them is 1 while there is a carry at bit $i - 1$. This is no different of the way a computer performs addition.

Similarly, $\mathcal{B}(a, b, i)$ is the *borrow* of the subtraction of b to a at bit i, and is computed as:

$$\mathcal{B}(a, b, i) = ([\overline{a}]_i \wedge [b]_i) \vee \big[([\overline{a}]_i \oplus [b]_i) \wedge \mathcal{B}(a, b, i - 1)\big], \tag{2.4}$$

with the same convention, that is, $\mathcal{B}(x, y, i) = 0$ if $i < 0$, for any x and y. Again, there is a borrow at bit i if either a is 0 and b is 1, or if they are equal but there is a borrow at bit $i - 1$.

Modular Addition

If the $+$ operator is defined as *modular* addition, as often in cryptography, extra care is needed. Namely, when $a \equiv b \pmod{N}$, that does not necessarily mean that $[a]_i = [b]_i$. Indeed, even though $4 \equiv 1 \pmod{3}$, $[4]_0 = 0 \neq [1]_0 = 1$. Recall that $[4]_0$ denotes the bit at index 0 in its base two representation (its LSB). What is true, however, is that if $a \equiv b \pmod{N}$, then $a \bmod N = b \bmod N$, hence $[a \bmod N]_i = [b \bmod N]_i \; \forall\, i \geq 0$.

In the particular case of $N = 2^L$, if $a \equiv b \pmod{N}$, then $[a]_i = [b]_i$ if $i < L$. Indeed, when N is a power of two, computing the remainder is similar to dropping all the bits above L.

In the practical subproblems discussed below, addition modulo 2^L is used, and bit indices are smaller than L, so we do not refer to this issue later on.

First Subproblem

This first subproblem is stated as follows.

Problem 1. *Given a and $[a + x]_i$, guess $[x]_i$.*

Equation (2.1), which yields:

$$[x]_i = [a]_i \oplus [a + x]_i \oplus \mathcal{C}(a, x, i - 1).$$

Both $[a]_i$ and $[a + x]_i$ are known, but the carry is unknown. Using Equation (2.3) yields the two following cases. When $[a]_k = 1$:

$$\begin{aligned}
\mathcal{C}(a, x, k) &= ([a]_k \wedge [x]_k) \vee \big[([a]_k \oplus [x]_k) \wedge \mathcal{C}(a, x, k - 1)\big] \\
&= [x]_k \vee \big([\overline{x}]_k \wedge \mathcal{C}(a, x, k - 1)\big) \\
&= [x]_k \vee \mathcal{C}(a, x, k - 1)
\end{aligned}$$

Table 2.1: Possible outputs of $\mathcal{C}(a,x,k)$ and their probability, given a.

$[a]_k$	$\mathcal{C}(a,x,k)$	$\Pr(\mathcal{C}(a,x,k)=1)$
0	$[x]_k \wedge \mathcal{C}(a,x,k-1)$	$\frac{1}{2}\cdot\Pr(\mathcal{C}(a,x,k-1)=1)$
1	$[x]_k \vee \mathcal{C}(a,x,k-1)$	$1-\frac{1}{2}\cdot\Pr(\mathcal{C}(a,x,k-1)=0)$

Table 2.2: Possible outputs of $\mathcal{B}(a,b\vee u,k)$ and their probability, given a, b.

$[a]_k$	$[b]_k$	$\mathcal{B}(a,b\vee u,k)$	$\Pr(\mathcal{B}(a,b\vee u,k)=1)$
0	0	$[u]_k \vee \mathcal{B}(a,b\vee u,k-1)$	$1-\frac{1}{2}\cdot\Pr(\mathcal{B}(a,b\vee u,k-1)=0)$
0	1	1	1
1	0	$[u]_k \wedge \mathcal{B}(a,b\vee u,k-1)$	$\frac{1}{2}\cdot\Pr(\mathcal{B}(a,b\vee u,k-1)=1)$
1	1	$\mathcal{B}(a,b\vee u,k-1)$	$\Pr(\mathcal{B}(a,b\vee u,k-1)=1)$

Likewise, when $[a]_k = 0$:

$$\begin{aligned}\mathcal{C}(a,x,k) &= ([a]_k \wedge [x]_k) \vee \big[([a]_k \oplus [x]_k) \wedge \mathcal{C}(a,x,k-1)\big]\\ &= [x]_k \wedge \mathcal{C}(a,x,k-1)\end{aligned}$$

If the distribution of x is known (in the case of nonces it is uniform), the probability of $[x]_k$ being 0 or 1 is computable, and thus the probability of $\mathcal{C}(a,x,i-1)$ being 0 or 1 as well.

In the general case, the actual values taken by $[x]_i$ are unknown, although some information may be gained. That information can be used to guess a possible value for it, which will be correct with a given computable probability. Assuming uniform distribution for x, the possible outputs and their probability of occurring are depicted in Table 2.1.

The output is $[x]_i = [a]_i \oplus [a+x]_i$ if $\Pr(\mathcal{C}(a,x,i-1)=1) < \frac{1}{2}$, and $[x]_i = [a]_i \oplus [a+x]_i \oplus 1$ otherwise. The probability of guessing right is computable given the distribution of x.

Second Subproblem

This second subproblem is stated as follows.

Problem 2. *Given a, b, and the relation $a = (b \vee u) + x$, find $[x]_i$ for a given i (u is unknown).*

From Equation (2.2) yields:

$$[x]_i = [a-(b\vee u)]_i = [a]_i \oplus [b\vee u]_i \oplus \mathcal{B}(a,b\vee u,i-1),$$

with $[a]_i$ known. Furthermore, if $[b]_i = 1$, then $[b\vee u]_i = 1$, and if not, there is a 50% chance of guessing the right bit (assuming uniform distribution for u). As for the borrow $\mathcal{B}(a,b\vee u,i-1)$, Equation (2.4) may be used in the same fashion as in the previous subproblem. If $[b]_k = 1$,

$$\begin{aligned}\mathcal{B}(a,b\vee u,k) &= ([\overline{a}]_k \wedge ([b]_k \vee [u]_k)) \vee \big[([\overline{a}]_k \oplus ([b]_k \vee [u]_k)) \wedge \mathcal{B}(a,b\vee u,k-1)\big]\\ &= [\overline{a}]_k \vee \big([a]_k \wedge \mathcal{B}(a,b\vee u,k-1)\big)\\ &= [\overline{a}]_k \vee \mathcal{B}(a,b\vee u,k-1)\end{aligned}$$

Otherwise, if $[b]_k = 0$:

$$\begin{aligned}\mathcal{B}(a,b\vee u,k) &= ([\overline{a}]_k \wedge ([b]_k \vee [u]_k)) \vee \big[([\overline{a}]_k \oplus ([b]_k \vee [u]_k)) \wedge \mathcal{B}(a,b\vee u,k-1)\big]\\ &= ([\overline{a}]_k \wedge [u]_k) \vee \big[([\overline{a}]_k \oplus [u]_k) \wedge \mathcal{B}(a,b\vee u,k-1)\big]\end{aligned}$$

Again, the actual values taken by $[x]_i$ are unknown, but it is possible to make a good guess with computable probability. Assuming uniform distribution for u, the possible outputs and their probability of occurring are depicted in Table 2.2.

Full-disclosure Attack

Attack Outline In this attack scenario, the adversary considered is passive and can therefore only eavesdrop the communications between a reader and a tag, i.e. the messages A, B, C and D, and IDS. It is also assumed that the channel between the reader and its database is secure.

The attack is a full-disclosure of the tag's secret ID. It is *probabilistic* in the sense that the adversary is never 100% sure of the ID recovered. However she can be as close as she wants to this certainty, as long as she has more protocol runs to listen to. It is not dependent of the definition of the rotation, though its efficiency is.

The idea is to build a progressive knowledge on ID with the information computed from public quantities. Each information gain requires three consecutive successful authentications, but other successful authentications can exist between each information gain.

In order to carry out the attack, it is first needed to compute the least significant bit (LSB) of ID, as described in the "Attack Initialization" section. Once the LSB is retrieved, the attack described in the "Attack Details" section reveals the remaining bits of ID.

Attack Initialization The aim here is to recover the LSB of ID. Therefore, only the case $i = 0$ matters, where the modular addition ($+$) and bitwise XOR ($\oplus$) are the same LSB-wise. Recall that $[x]_i$ denotes the i-th bit of x.

Lemma 1. *If* $[IDS]_0 = 1$ *and* $[B]_0 \oplus [C]_0 \oplus [D]_0 \oplus [IDS^{next}]_0 = 1$*, then*

$$[ID]_0 = [B]_0 \oplus [IDS^{next}]_0 \oplus 1.$$

Proof. The message (S-B) at the LSB is:

$$[B]_0 = ([IDS]_0 \vee [K_2]_0) \oplus [n_2]_0.$$

Since $[IDS]_0 = 1$, we have that $[n_2]_0 = [B]_0 \oplus 1$, no matter $[K_2]_0$. Moreover, from message definitions (S-B), (S-C), (S-D), and (S-IDS), the following relations are true at the LSB:

$$\begin{aligned} [B]_0 &= 1 \oplus [n_2]_0 \\ [C]_0 &= [K_1]_0 \oplus [K_2]_0 \oplus [\bar{K_1}]_0 \oplus [\bar{K_2}]_0 \quad (2.5) \\ [D]_0 &= [\bar{K_2}]_0 \oplus [ID]_0 \oplus (([K_1]_0 \oplus [K_2]_0) \vee [\bar{K_1}]_0) \quad (2.6) \\ [IDS^{next}]_0 &= [IDS]_0 \oplus [ID]_0 \oplus [n_2]_0 \oplus [\bar{K_1}]_0 \end{aligned}$$

Hence,

$$[B]_0 \oplus [C]_0 \oplus [D]_0 \oplus [IDS^{next}]_0 = [K_1]_0 \oplus [K_2]_0 \oplus (([K_1]_0 \oplus [K_2]_0) \vee [\bar{K_1}]_0).$$

Since $[B]_0 \oplus [C]_0 \oplus [D]_0 \oplus [IDS^{next}]_0 = 1$, it is impossible that $[\bar{K_1}]_0 = 0$. Therefore, $[\bar{K_1}]_0 = 1$ and:

$$[B]_0 \oplus [C]_0 \oplus [D]_0 \oplus [IDS^{next}]_0 = [K_1]_0 \oplus [K_2]_0 \oplus 1.$$

Now that those two quantities are known, $[\bar{K_2}]_0$ may be computed using (2.5):

$$[\bar{K_2}]_0 = [B]_0 \oplus [D]_0 \oplus [IDS^{next}]_0.$$

The latter equation is then used in (2.6):

$$\begin{aligned} [D]_0 &= [\bar{K_2}]_0 \oplus [ID]_0 \oplus (([K_1]_0 \oplus [K_2]_0) \vee [\bar{K_1}]_0) \\ &= [B]_0 \oplus [D]_0 \oplus [IDS^{next}]_0 \oplus [ID]_0 \oplus 1. \end{aligned}$$

□

A quantity is said to have a uniform distribution if every element of its domain is equally likely to be instantiated. It is quite easy to see that public quantities IDS, A, B, C, and D have uniform distribution, since their computation involves either a bitwise XOR, or a modular addition with a nonce or with a key (keys also have uniform distributions because they are also updated using a bitwise XOR with a nonce). The domain of these quantities is $[0, 2^L - 1]$, and hence they also have "bitwise" uniform distribution in the sense that every bit has an equal probability to be a zero or a one.

Getting the LSB of ID requires two bits to be equal to 1. Since quantities taken into account for this observation have bitwise uniform distribution, the probability of occurrence is $\frac{1}{4}$. The number of runs needed for this observation has a geometric distribution of average four. The result is quite similar to the one in [161], except that here the adversary knows for sure when conditions are met, because they only involve public quantities.

In fact, the attack could be generalized to an arbitrary bit index, but this would need conditions of which probabilities of occurrence decrease exponentially with the position of that bit index. Instead, a much more efficient attack to retrieve $[ID]_i$ when $i > 0$ is described in the next section.

Attack Details It is assumed in the following that the adversary has executed the initial part of the attack, and that she knows $[ID]_0$.

At the LSB, (S-IDS) becomes:

$$[IDS^{(n+1)}]_0 = [IDS^{(n)}]_0 \oplus [ID]_0 \oplus [n_2^{(n)}]_0 \oplus [\bar{K_1}^{(n)}]_0. \tag{2.7}$$

When $[IDS^{(n)}]_0 = 1$, $[n_2^{(n)}]_0$ is known by computing $[B^{(n)}]_0 \oplus [IDS^{(n)}]_0$. Thus, Equation (2.7) yields:

$$[\bar{K_1}^{(n)}]_0 = [ID]_0 \oplus [B^{(n)}]_0 \oplus [IDS^{(n+1)}]_0.$$

The next round, according to the key updating process $K_1^{(n+1)} = \bar{K_1}^{(n)}$, $[K_1^{(n+1)}]_0$ is known. Furthermore, if again $[IDS^{(n+1)}]_0 = 1$, then $[n_2^{(n+1)}]_0 = [B^{(n+1)}]_0 \oplus [IDS^{(n+1)}]_0$. Therefore, $[K_1^{(n+1)} \oplus n_2^{(n+1)}]_0$ is known. So, since $\bar{K_1}^{(n+1)} = Rot(K_1^{(n+1)} \oplus n_2^{(n+1)}, K_1^{(n+1)})$,

$$[K_1^{(n+1)} \oplus n_2^{(n+1)}]_0 = [\bar{K_1}^{(n+1)}]_{r(K_1^{(n+1)})}.$$

Recall from the protocol description that r denotes the function used in the rotation operation. In most cases, $r(K_1^{(n+1)})$ is not known (only its LSB is). We can write, assuming K_1 has a uniform statistical distribution,

$$\Pr(r(K_1^{(n+1)}) = i) = p_r(i) \qquad \forall\, i \in [0, L-1],$$

where p_r is the probability distribution function of r. For instance, in the case of modular rotation, $r(x) = x \bmod L$, and $p_r(x) = \frac{1}{L}$, and in the case of Hamming weight rotation, $r(x) = \mathcal{H}(x)$, and $p_r(x) = \frac{\binom{L}{x}}{2^L}$. So, with probability $p_r(i)$, $[\bar{K_1}^{(n+1)}]_i$ is known. This result is used in (S-IDS):

$$[ID + IDS^{(n+1)}]_i = [IDS^{(n+2)}]_i \oplus [n_2^{(n+1)}]_i \oplus \underbrace{[\bar{K_1}^{(n+1)}]_i}_{[K_1^{(n+1)} \oplus n_2^{(n+1)}]_0}.$$

The computation of $[n_2^{(n+1)}]_i$ has already been discussed in the "Second Subproblem" section. Finally, $[ID + IDS^{(n+1)}]_i$ and $IDS^{(n+1)}$ are known (with a certain probability), but this does not necessarily mean that $[ID]_i$ is. However, some information on $[ID]_i$ can be recovered using results from the "First Subproblem" Section. Finally, a value for $[ID]_i$ is obtained with a given probability, which is quantified in the next section.

An outline of the attack can be seen in Algorithm 1.

In order to provide a more intuitive description of the attack, consider the following game. Having a slightly biased coin, the goal is to know what side of this coin has the highest probability of appearing. It may be tossed as many times as wanted, but the idea is to have a good probability of guessing the right side, while minimizing the number of tosses. The smaller the bias is, the harder it is to tell whether it is heads or tails that is the most favorable side. Indeed if the bias is, for instance, 75% for heads and 25% for tails, as little as 20 tosses are sufficient to win the game

Algorithm 1 Outline of the attack. (1) refers to the "First Subproblem" section, and (2) to the "Second Subproblem" section. Note that the "computed" $[K_1^{next}]_0$ is on the attacker side, and that the actual $[K_1^{next}]_0$ is unknown. For clarity reasons, there is no difference of notation between computed (or guessed) values and real ones.

```
Execute the traceability attack to get $[ID]_0$
repeat
  if $[IDS]_0 = 1$ then
    if the previous $[K_1]_0$ was known (i.e. previous $[IDS]_0$ was 1) then
      Compute $n_2$ probabilistically (2)
      Compute $ID$ probabilistically given $IDS$ and $IDS + ID$ (1)
      for all $i \in [1, L-1]$ do
        Update the knowledge of $[ID]_i$ with the advantages of (1) and (2) and $p_r(i)$
      end for
    end if
    Compute $[K_1^{next}]_0 = [ID]_0 \oplus [B]_0 \oplus [IDS^{next}]_0$
  end if
until all the bits of $ID$ are found with satisfactory probability
```

with overwhelming probability. However, if the bias turns out to be 50.01% for heads, and 49.99% for tails, a lot more of them are needed.

This is exactly the idea of convergence of the attack, except that the biased side of $L-1$ independent coins are the objective, and that each toss has a different "weight". Indeed only a little bit of information is gained, and only when $[IDS]_0 = 1$ for two consecutive runs. Moreover, each information gain has to be weighted with $p_r(i)$, the information on n_2, and the information on ID given $ID + IDS$.

Theoretical Analysis of the Attack

The optimal strategy for the "update of knowledge" in the attack is analyzed hereafter. It corresponds to giving what conclusion may be drawn given the set of observations and by linking these observations with the probability of guessing the right ID. It is an application of Bayesian inference.

Suppose that an oracle is available, which has a secret bit b and a set of $0 \leq q_k \leq 1$ that, upon query, outputs a bit b_k and a value q_k such that $\Pr(b_k = b) = q_k$. It is assumed that $\Pr(b = 0) = \Pr(b = 1) = \frac{1}{2}$, that all the outputs are independent, and that $q_k \geq \frac{1}{2}$, without loss of generality. Indeed, if one guess is such that $q_k < \frac{1}{2}$, then it would be equivalent to output the opposite bit with complementary probability $1 - q_k > \frac{1}{2}$.

For convenience, O denotes the observations, that is the event corresponding to observing the set of bits b_k. the following sets are also defined:

$$S_0 = \{k \in [1, N] \mid b_k = 0\}, \text{ and}$$
$$S_1 = \{k \in [1, N] \mid b_k = 1\}.$$

If $N = |S_0| + |S_1|$ outputs of the oracle are observed, then:

$$\Pr(O|b = 0) = \prod_{k \in S_0} q_k \prod_{k \in S_1} (1 - q_k) \tag{2.8}$$

$$\Pr(O|b = 1) = \prod_{k \in S_0} (1 - q_k) \prod_{k \in S_1} q_k \tag{2.9}$$

Using Bayes' rule :

$$\begin{aligned}\Pr(b=0|O) &= \frac{\Pr(b=0\cap O)}{\Pr(O)}\\ &= \frac{\Pr(O|b=0)\cdot\Pr(b=0)}{\Pr(O|b=0)\cdot\Pr(b=0)+\Pr(O|b=1)\cdot\Pr(b=1)}\\ &= \frac{\Pr(O|b=0)}{\Pr(O|b=0)+\Pr(O|b=1)}\end{aligned}$$

since events $b=0$ and $b=1$ are equiprobable. Equations (2.8) and (2.9) yield:

$$\begin{aligned}\Pr(b=0|O) &= \frac{\prod_{k\in S_0} q_k \prod_{k\in S_1}(1-q_k)}{\prod_{k\in S_0} q_k \prod_{k\in S_1}(1-q_k)+\prod_{k\in S_0}(1-q_k)\prod_{k\in S_1} q_k}\\ &= \frac{1}{1+\prod_{k\in S_0}\frac{1-q_k}{q_k}\prod_{k\in S_1}\frac{q_k}{1-q_k}}\end{aligned} \tag{2.10}$$

An equivalent but more convenient way of seeing this is the following. Instead of outputting probabilities q_k, the oracle can output *advantages* a_k such that $|\Pr(b_k=b)-\Pr(b_k\neq b)|=a_k$. Put differently,$a_k=|2q_k-1|$. In this scenario, Equation (2.10) becomes:

$$\Pr(b=0|O)=\frac{1}{1+\prod_{k\in S_0}\frac{1-a_k}{1+a_k}\prod_{k\in S_1}\frac{1+a_k}{1-a_k}} \tag{2.11}$$

Recall from Algorithm 1 that the information on each bit of ID is weighted using:

- $p_r(i)$,
- the trust level on $[ID]_i$ given $[ID+IDS]_i$ (subproblem 1),
- the trust level on $[n_2]_i$ (subproblem 2).

Indeed, each guess on the i-th bit of ID is correct if:

- the guessed rotation is the correct one,
- the guess of $[ID]_i$ given $[ID+IDS]_i$ is correct,
- the guess at $[n_2]_i$ is correct.

The probability of correctness for the rotation is simply $p_r(i)$, and the probability of correctness for the two subproblems (the level of trust or *advantage* on the quantities n_2 and ID) is computable, given public messages, as seen in Sections 2.1.3 and 2.1.3. If independence between these advantages is assumed, then one just has to multiply them to obtain an advantage a_k related to the i-th bit on the k-th run. Using Equation (2.11), a total probability on the value of a bit of ID can be computed, given a certain amount of information materialized by the guesses and advantages on these guesses on that bit. Hence, for every iteration of the attack, a little bit of information is gained, that is quantified by one term of one of the products in Equation 2.11. Once satisfactory, the success probability for each individual bit can be computed using Equation 2.11.

Both theoretically and experimentally (see original paper [20]) that the average advantages for the first and second subproblems are respectively roughly $\frac{1}{2}$ and $\frac{1}{3}$. This is particularly important for the second subproblem, where simply knowing IDS and B yields roughly one third of n_2.

Recall from Section 2.1.3 that to execute the inner part of the attack and thus bring information, it is required that $[IDS]_0=1$ for two consecutive runs. Since IDS has a uniform distribution, this will occurs with probability $\frac{1}{4}$. The average number of runs needed is thus multiplied by four.

Optimizations and Experimentations

Some optimizations that improve in practice the efficiency of the attack presented in Section 2.1.3 are presented in this section.

The idea is that it is somewhat wasteful to only use $[ID]_0$ while more and more knowledge on ID is revealed along the attack. Progressive knowledge on ID can not only help solving the first subproblem ($[ID]_i$ from IDS and $[ID+IDS]_i$), but it can also be used as a base, instead of $[ID]_0$ only.

This especially helps with the Hamming-weight rotations where the most and least significant bits are hard to guess using $[ID]_0$ only (because $p_r(i)$ is very small for small or big i).

Experiments have been carried out with the two definitions of the rotation and the number of bits correctly guessed on average have been observed. When this number is close to L, it means that the whole ID can be recovered with good probability, and when it is close to $\frac{L}{2} = 48$, it means that the outputted guessed ID is not better than if guessed at random. Figure 2.2 shows the evolution of the quality of the ID recovered (when applying the optimization) with respect to N, the number of observed runs[4].

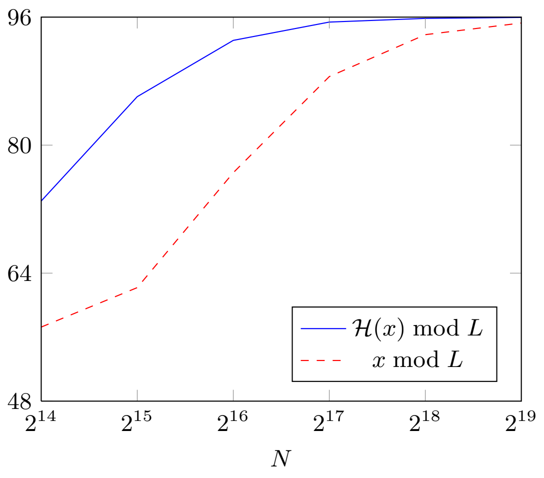


Figure 2.2: Average number of bits correctly guessed in ID when the adversary observes N runs, for the two usual rotations (optimization applied).

Other optimizations may also be applied, such as the following. The adversary does not know K_1 in whole, but she does know $[K_1]_0$, and for instance in the case of modular rotation, she knows the parity of it, and thus she can reduce the possibilities from L down to $\frac{L}{2}$, which improves the advantage per sample quite a bit. She could also reuse old authentication sessions when she has better knowledge of ID. These methods were not taken into account in the experiments for Figure 2.2. The point is that the number of runs needed to achieve overwhelming probability could be further reduced, but the first optimization alone is enough to make the attack practical.

2.2 Statistics on the State of the Art

This section presents a few facts on the state of the art of ultralightweight protocols.

Table 2.3 presents a list of most prominent ultralightweight authentication protocols, and the first attack on each of them, when applicable. It also shows the date of publication of each article and highlights the time in months elapsed between the publication of a protocol and the publication of an attack on it. What is considered to be a “first attack” is the chronologically first traceability or disclosure attack that targets the specific construction of a protocol (desynchronization attacks were discarded because, although important, they do not target specific protocol weaknesses but

[4]These results were obtained on an average of roughly 500 experiments conducted with a simulated SASI initiated with random secrets.

rather general issues in their structure). The dates, both for protocols and attacks, are taken to be the earliest after becoming public (e.g., eprint or date of conference over proceedings).

Table 2.3: List of ultralightweight protocols, the first attack on them, and the relevant dates.

Protocol	Paper	Pub. Date	First Attack	First Attack Date	Months
LMAP	[155]	07-2006	[129]	05-2007	10
M2AP	[156]	09-2006	[129]	05-2007	8
EMAP	[154]	11-2006	[127]	04-2007	5
SLMAP	[128]	10-2007	[103]	05-2009	19
SASI	[64]	12-2007	[161]	06-2008	6
Qingling-Yiju-Yonghua	[163]	8-2008	[153]	7-2009	11
Gossamer	[157]	09-2008	N/A	N/A	N/A
LMAP++	[126]	09-2008	[33]	04-2011	31
David-Prasad	[67]	06-2009	[101]	06-2010	12
Lee-Hsieh-You-Chen	[122]	08-2009	[158]	02-2010	6
Yeh-Lo-Winata	[191]	02-2010	[159]	10-2010	8
Eghdamian-Samsudin	[76]	11-2011	[16]	06-2012	7
RPAP	[143]	10-2011	[16]	06-2012	8
NRS	[81]	11-2011	[3]	12-2013	25
LPP	[78]	12-2011	[3]	12-2013	24
PUMAP	[40]	3-2012	[16]	06-2012	3
DIDRFID/SIDRFID	[124]	5-2012	[16]	06-2012	1
RAPP	[176]	5-2012	[16]	06-2012	1
Improved LMAP+	[92]	5-2012	[16]	06-2012	1
RIPTA-DA	[84]	7-2012	[32]	7-2013	12
Pang-Li-He-Alramadhan-Wang	[152]	3-2013	[2]	12-2013	9
DT	[75]	5-2013	[34]	06-2013	1
RAPLT	[109]	10-2013	[19]	11-2013*	1*
UMAP	[43]	9-2013	[19]	11-2013*	2*
Ling-Shen	[133]	11-2013	[19]	02-2014*	3*
LPCP	[85]	11-2013	[2]	12-2013	1

* Date when the attack was found, not published. Numbers omitted in the discussions below for fairness.

Figure 2.3 shows the distribution of the time required for the publication of an attack, based on data from Table 2.3. Although there are many factors influencing the data, it is clear that the time for publication of a serious attack is extremely short. The average number of months required is about 9.5[5]. Half of the protocols are broken within 8 months, and most of them within a year.

Note that depending on the journal or conference, it takes significant time to publish an attack on a protocol. A minimal estimation is that at least three to four months are required from finished research to publication. Generally, it might also take time for the proceedings to become available. Note finally that some of these protocols have been published in relatively low-visibility venues, which has definitely helped increase their life expectancy. It is probably safe to assume that on average, a typical ultralightweight protocol is broken in under four months following its publication.

In most cases, these protocols repeat time and again the same typical mistakes, and no progress is achieved. Many proposals present striking similarities with existing protocols, and often have the exact same flaws, or even worse ones. Sometimes, hash functions or ciphers are used alongside usual ultralightweight operations such as XOR, additions, rotations, etc. (an example is presented in Section 2.5). This makes the protocol basically not less expensive than a classical challenge-response, which is no longer ultralightweight, and sometimes weaker. Moreover, there are typically one to three papers presenting attacks on each protocol, and again they often exploit the same

[5]Gossamer was discarded in these calculations since no attack has been published on it yet to the best of our knowledge. There is, however, an existing desynchronization attack (see [174] and [192]), but it is related to the structure of Gossamer, not its operations, and the two papers propose different fixes to the problem.

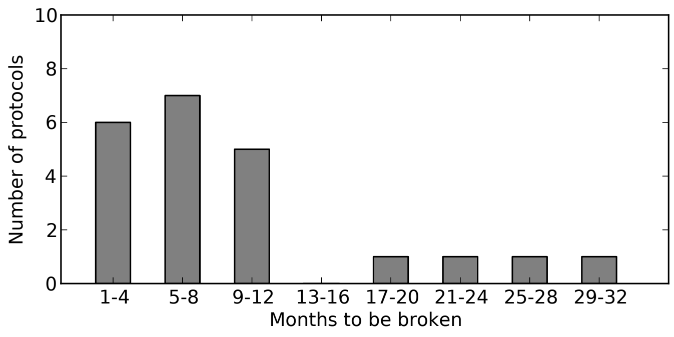


Figure 2.3: Histogram of the time for ultralightweight protocols to be broken.

typical weaknesses.

2.3 Common Flaws

2.3.1 Linearity and T-functions

The same definition of T-functions as proposed by Klimov and Shamir in [118] are used here: A T-function is a mapping from n-bit words to n-bit words in which for each $0 \leq i < n$, the bit i in any output word depends only on bits $0, 1, ..., i$ of any input word. This concept is very useful because all the boolean operations and most of the numeric operations in modern processors are T-functions. Additionally, the composition of T-functions results also in a T-function.

These have many nice properties, but almost by definition also a quite undesirable one: they achieve very poor levels of diffusion. By definition, in a T-function it is not possible that all output bits depend on all input bits, which is the ideal scenario for security purposes. This is particularly dangerous in cryptographic applications, lightweight or otherwise. The only reasonable way to address this shortcoming is by combining these operations with other which do not exhibit this characteristic. But unfortunately many designers do not follow this simple combination rule, and have proposed schemes entirely based on T-functions which are doomed to fail.

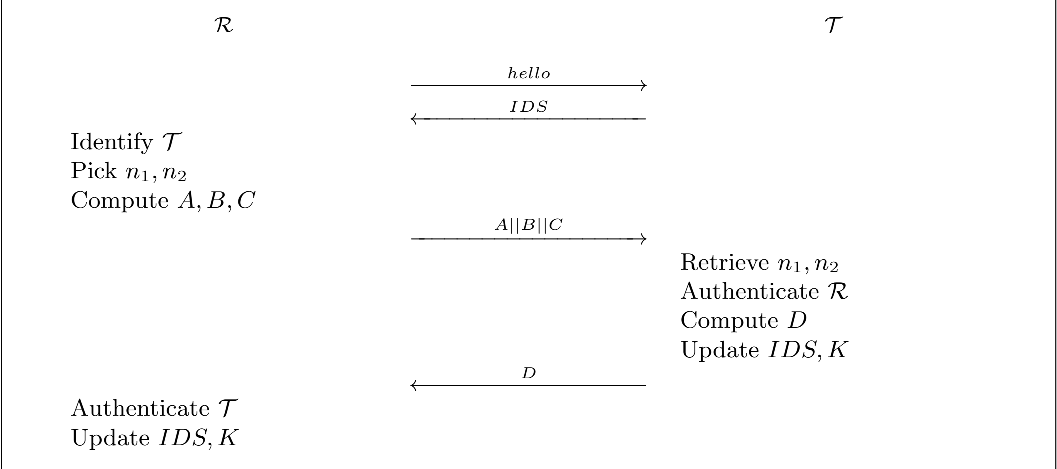
Some common mistakes found in the UMAP family of protocols are discussed hereafter. These were pioneer proposals in many ways but they incurred in what, with the hindsight gained after more than 8 years of research in the area, now looks like quite elementary mistakes. As an example, the LMAP protocol (presented in [155]) is pictured in Figure 2.4 and discussed below.

As can be seen in Figure 2.4, this early protocol is composed exclusively of T-functions (modular addition, $\oplus$, $\vee$). This seriously limits its security due to their poor diffusion characteristics. Not only the message generation is marred by this, but it also affects the key update phase. This same pitfall is present in all members of the UMAP protocol family. The quick appearance of multiple attacks like those of [36, 35, 129] is, in retrospect, harldy surprising. Despite this, recent proposals continue to rely heavily and carelessly on T-functions.

Not only T-functions but also linearity[6] should be avoided, or at least dealt with carefully in cryptographic protocols. If a protocol or primitive is linear, it always accepts a better (and in most cases much better) attack than exhaustive key search. Not to mention linear cryptanalysis attacks. This, of course, flagrantly violates one of the basic design principles of every cryptographic design, hence the reasoning behind avoiding linearity at any price, as it constitutes a vulnerability in itself.

This may be clear to most members of the cryptography community, but seems less obvious for some of the designers of lightweight protocols. Most people know that some of the common bitwise operations are linear, but many seem to forget that permutations in general and rotations (for more security implications of the rotation operation, please see Section 2.3.3) in particular are

[6]Here the term linearity is used to abstract the characteristic that small input changes produce small, bounded output changes, and should be seen as opposite to non-linearity or to the avalanche effect property, where minimal input changes produce an avalanche of changes in the output. In that sense, a hash function is highly non-linear and $\oplus$ is linear. Modular addition, on the other hand, is not considered linear in Z_2 because of the carry bit, though it admits good linear approximations.



$$\begin{aligned}
A &= IDS \oplus K_1 \oplus n_1 \\
B &= (IDS \vee K_2) + n_1 \\
C &= IDS + K_3 + n_2 \\
D &= (IDS + ID) \oplus n_1 \oplus n_2
\end{aligned}$$

$$\begin{aligned}
IDS^{(n+1)} &= (IDS^{(n)} + (n_2^{(n)} \oplus K_4^{(n)})) \oplus ID \\
K_1^{(n+1)} &= K_1^{(n)} \oplus n_2^{(n)} \oplus (K_3^{(n)} + ID) \\
K_2^{(n+1)} &= K_2^{(n)} \oplus n_2^{(n)} \oplus (K_4^{(n)} + ID) \\
K_3^{(n+1)} &= (K_3^{(n)} \oplus n_1^{(n)}) + (K_1^{(n)} \oplus ID) \\
K_4^{(n+1)} &= (K_4^{(n)} \oplus n_1^{(n)}) + (K_2^{(n)} \oplus ID)
\end{aligned}$$

Figure 2.4: The LMAP protocol.

also linear operations, so one should not base the security of a protocol on those operations alone, because the composition of linear operations is also linear.

Nonlinearity needs to be injected at some stage of the design, or otherwise the protocol remains linear overall and, consequently, very insecure. Despite this, many examples of designers disregarding this basic policy may be found in the literature. Particularly notable and common examples are the many proposals whose security seems to be based almost exclusively in the use of Cyclic Redundancy Codes (CRCs) that many authors seem to forget are also linear and, as such, offer very little security if at all.

This is excusable to a certain extend in proposals that want to be compliant with the EPC-C1-G2 standard recommendations, but then the security claims of the proposed protocols should be adjusted down accordingly. CRCs are linear and they should under no circumstances be employed in cryptography. For a further example of this, see Section 2.4.2.

As an additional example of how pervasive this mistake is, the papers [85] and [152] are good very recent illustrations of this common failure. Not surprisingly they were both quickly dismantled in [2].

2.3.2 Biased Output

Another important weakness of many lightweight schemes proposed in the last years is that some of the operations used have a biased output, a feature that in many cases lead to additional security vulnerabilities. This is typical of boolean functions such as OR ($\vee$) and AND ($\wedge$), where $x \vee y$ and $x \wedge y$ have, for unbiased random values of x and y, heavily (75%) biased outputs, respectively, towards $\overline{1}$ and $\overline{0}$. This can constitute a security weaknesses because these two boolean functions leak information for both of their arguments. For example, if $x_i \vee y_i = 0$, then $x_i = y_i = 0$, which

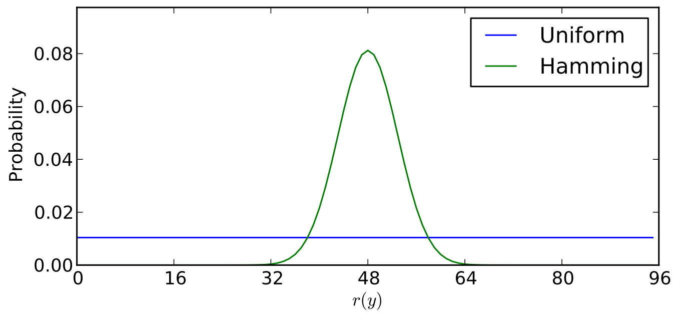


Figure 2.5: Distribution of probability of $r(y)$ for the modular and Hamming weight rotations, with y a uniform random variable and $L = 96$.

discloses both the inputs. This happens roughly 25% of the times (similarly with AND, of course). This could be more than enough to, after seeing some exchanges, be able to completely recover all the inputs. An additional undesirable property derived from these highly biased outputs is the possibility of message insertion. For instance, imagine that one of the exchanged messages, used for synchronization or authentication purposes, is the result of one of this biased operators. Then, an attacker can easily impersonate it with a high probability, so it is a very bad idea to use it for authentication purposes, as proposed in many protocols. LMAP, for instance, has the message $B = (IDS \vee K_2) + n_1$, a typical construction that has been seen in this or similar forms many times later. The issue here is that B is very biased and thus can be impersonated with high probability even for an attacker who does not know any of the secrets K_2 or n_1. Even more worryingly, as the value of B is public (it is exchanged in the open), the attacker can use $B - \overline{1}$ as a very good approximation to the secret n_1 (on average, 75% of the bits are correct), and this approximation can be used later in other parts of the protocol to approximate the secret.

2.3.3 Rotations

Rotations have been used for a long time in cryptography. Modern block ciphers and hash functions still mostly rely on ARX (addition, rotation, XOR) designs. Rotations are extremely cheap to implement, and they bring diffusion, which complements nicely the addition and the XOR (which exhibit poor diffusion properties, as shown in Section 2.3.1). Fixed-amount rotations are typically used in ARX designs, but data-dependent rotations, as first featured in the RC5 block cipher [165], also exist.

The SASI [64] protocol was the first ultralightweight authentication protocol to feature data-dependent rotations as far as we know. Since then, most ultralightweight protocols have used them, and in many cases they are the weak spot for ad-hoc attacks. Two types of rotations are typically used in these protocols: modular rotations (as used in RC5) and Hamming weight rotations. In the following, $Rot(x, y)$ denotes the rotation operation, in accordance with the literature. The rotation consists in a circular left shift of x by $r(y)$ positions, where $r(y) = y \bmod L$ for modular rotations, and $r(y) = \mathcal{H}(y)$ for Hamming weight rotations.

One thing to bear in mind is that a rotation is a permutation and is therefore linear, inheriting all the pitfalls commented in Section 2.3.1.

The most important shortcoming with data-dependent rotations is that there are only L possible outputs (where L is the size in bits of the input). Moreover, the distribution of these outputs is known *a priori*. The modulo operation has a uniform distribution over uniform input, and the Hamming weight has binomial distribution $B(L, 1/2)$ over uniform input. Figure 2.5 shows the distribution of probability of $r(y)$ for the two rotations, with y a uniform random variable and with $L = 96$[7].

Modular rotations have a maximal entropy ($\log_2 L = 6.58$ bits), since each shift is equiprobable. This minimizes the advantage of the adversary for guessing the output. Nevertheless, they are quite probable to behave like the identity operation (probability of $1/L$). This has led to many attacks,

[7]This is the value that is typically used in the literature for key lengths.

whether when the attacker assumes this blindly (as the desynchronization attack on PUMAP in [16]) or when she is able to verify it (as the traceability attack on modular SASI in [100]).

Hamming weight rotations have worse entropy (e.g. 4.34 bits in the case of $L = 96$) than modular rotations. In particular, the three-sigma rule implies that, for instance in the case of $L = 96$, the number of bits rotated is between 33 and 63 in 99.7% of cases. It is therefore even easier for an attacker to guess the output of a rotation. However, Hamming weight rotations virtually never behave like the identity.

Another promising tool to attack schemes using rotations and additions is "Mod n cryptanalysis" [116]. Although it has never been applied in the cryptanalysis of an ultralightweight protocol, it has, on the other hand, been used to successfully attack block ciphers such as RC5P [114] (a variant of RC5 where xors are replaced with modular additions) and M6 [104] (a family of ciphers proposed for use with the he IEEE1394 FireWire stand), which use the same kind of operations as ultralightweight protocols.

2.3.4 Message Composition

Securely designing the messages exchanged over an ultralightweight protocol is a difficult open problem. Keeping the secrets exchanged as secure as possible against any leakage is indeed a big challenge, particularly in such constrained environments. As a general rule of thumb, it seems that the deeper these secrets are into the message, the better.

There is a light parallelism here between being deep inside the message expression and the operation of repeated rounds in a block cipher.

In general, it seems that the outer the secret you want to protect, the easier it is for the attacker to recover it. For instance, in LMAP, the key update phase is defined by

$$IDS^{(n+1)} = (IDS^{(n)} + (n_2^{(n)} \oplus K_4^{(n)})) \oplus ID. \tag{2.12}$$

The ID, the secret that the whole protocol is designed to protect, is in the outermost part of the message or, alternatively at the root of the operations tree. This quite frequent feature heuristically leads to major leakage of secret bits, as the rest of the message the ID is combined with is far from being perfectly random, particularly with the usual constraints in GE, time and power consumption imposed unto these ultralightweight protocols.

There are many more examples that exhibit transgressions to this rule of thumb for minimizing secret leakage, even within the most recent proposals. Consider for instance this message of the RAPP protocol [176]:

$$C = Per(n_1 \oplus K_1, n_1 \oplus K_3) \oplus ID. \tag{2.13}$$

The ID is again at the out most position of the message construction. This would have no serious consequences if the rest of the message was perfectly random (as in a One Time Pad), but this is not the case here. Any reasonable alternative, like putting the secret ID in an inner position, would be at least slightly better, as for example in $C = Per(n_1 \oplus K_1, ID \oplus n_1 \oplus K_3)$.

2.3.5 Knowledge Accumulation

Knowledge Accumulation of a Static Secret

If partial leakage of a static secret occurs in a round of a protocol, there is an obvious traceability issue. Indeed, it becomes possible for an attacker to correlate two leaked traces of an eavesdropped exchange. A typical example is recovering the least significant bit of the static identifier (see for instance [161], the first traceability attack on SASI).

More importantly, an attacker is sometimes able to recover the full static secret after a few rounds. Indeed, she can combine the different observations using Bayesian inference. An example of such an attack was the full cryptanalysis of SASI [20].

Key Updating

One of the initial goals of synchronized protocols is to provide forward privacy. Forward privacy is a stronger notion than privacy. Simply put, a protocol is said to be forward private if an attacker, having recovered the internal state of a tag, is not able to recognize the tag in past interaction traces. For a more formal definition, see [149]. Forward privacy cannot be achieved in a protocol if the secrets used in the exchange are all static. Indeed, if the attacker knows the secrets of a tag at some point, it also knows them in the past, since the secret does not change in the tag's lifetime. Therefore, she can recompute the messages sent by a tag in previous interactions, and recognize it easily. Note that a changing secret is required for forward privacy, but it does not guarantee it (indeed, there are many synchronized protocols that are not private, and therefore not forward private).

A positive side effect, and possibly the main advantage of changing the secrets is that it is conceivably harder to obtain the full secret at any given time, if only a partial leakage is obtained at every authentication round. It seems to be a good feature as it is intuitively harder to hit a "moving target" that a static one, akin to the internal state of a stream cipher.

Synchronized protocols however have the issue of being susceptible to desynchronization attacks.

2.3.6 Desynchronization

Desynchronization attacks are active attacks. They are studied here despite some ultralightweight protocols not claiming resistance against active attackers, only against passive ones. Nevertheless, as pointed out in Section 2.1.2, active attacks are particularly relevant in the considered context. In any case, most authors aim to achieve resilience against active attacks, including some defense against desynchronization.

Desynchronization generally occurs when an active attacker is able to stop the prover and/or the verifier to engage in further successful executions of the protocol. It is generally achieved by tricking the prover, the verifier, or both, into believing that a successful authentication session has taken place, thus updating internal counters and variables in an asynchronous way. Common implementations of this attack involve sending specially crafted messages to make only one of the involved parties reach an irreversible state from which it will not be capable of communicating in the future with other parties.

Many ultralightweight authentication protocols have been shown to be vulnerable to attacks of this kind. A widespread defense mechanism consists in storing the last pseudonym used, together with the current one, and use the former if the latter fails (as first suggested in LMAP [155], by using a flag that is activated when an authentication session terminates abnormally). This approach however modifies the behavior of the protocol, which opens the door to other attacks such as timing attacks [22]. Moreover, it comes with the cost of storing an extra copy of the state, which is relatively expensive in terms of area. Although not exactly flawed, this solution is not perfect.

Desynchronization protection in ultralightweight authentication protocols is relatively poorly studied on its own, and in any case caution is recommended when analyzing the security against such attacks.

An example of two classical desynchronization attacks can be found in [172]. Desynchronization can be considered as a type of Denial of Service attack, and as most DoS attacks, it does not admit an easy, ideal solution.

2.3.7 Vulnerability to Systematic Black-box Attacks

Tango Attack

The Tango Attack was first introduced in [101] and [159], and later employed in [39]. Its concept is extremely simple, but it is notable for being one of the very few systematic black-box attacks that can be applied to many different ultralightweight authentication protocols.

It could also be seen as a new tool to analyze lightweight protocols, and thus helpful in the design of more secure future proposals. The Tango attack is successful against other lightweight protocols, apart from those it was shown to break in the literature. This is a direct consequence of designers not having strict constraints on the resources needed for an adequate (i.e. highly nonlinear) mixture of the internal secret values in the message composition. This makes it hard to avoid leaking some secret bits in every session. The Tango is an attack that aims to obtain a

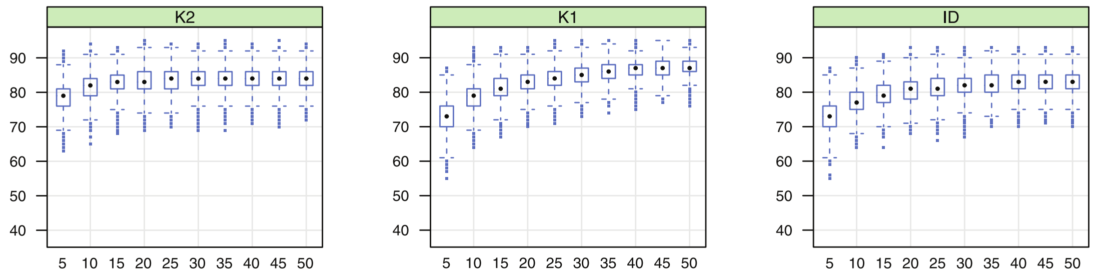


Figure 2.6: A genetic tango attack against the DavidPrasad RFID ultralightweight authentication protocol, from [39]. Plots represent the number of bits recovered with respect to the number of eavesdropped sessions.

full disclosure of all secrets that the protocol is designed to conceal. It is a passive attack that generally only needs to eavesdrop a small number of (possibly consecutive) authentication sessions, as shown in Figure 2.6. This is a very realistic attack scenario. The Tango attack is a simple, yet powerful technique of cryptanalysis which is based on the computation and full exploitation of multiple approximations to said secret values, using Hamming distances and the representation of variables in an n-dimensional space.

Its first use was against the David and Prasad protocol [67], and it emerged as a passive (i.e. completely realistic in the underlying security model) and extremely efficient attack to fully recover the secret key values K_1 and K_2 and the static identifier of the tag ID.

The attack is divided into two main phases: 1) Selection of good approximations; and 2) Combination of these good approximations for disclosing K_i or ID. The two phases are briefly described hereafter.

Phase 1: The attack exploits the leakage of secret information over the insecure radio channel due to fact that exchanged messages are derived from secret values by using relatively simple (i.e. not highly nonlinear) operations only. This is why the attacker can find and succeed in using multiple simple combinations of the exchanged public messages $f(A, B, D, E, F)$ as good approximations for the secrets K_i or ID. In weak protocols, public exchanged messages do not hide these secret values well enough. From all the set of approximations, the adversary is interested on those that are systematically closer (on average) to the target secret value. That is, those for which the Hamming distance between an approximation Z and the value X deviates from the expected value 48, so either $\mathcal{H}(Z, X) < 48$ or $\mathcal{H}(Z, X) > 48$ significantly.

Phase 2: The basic idea at this stage of the attack is to combine multiple approximations obtained in different sessions, to construct a global one which is highly correlated with the secret values (i.e. keys K_i and static identifier ID). This can be done in a number of different ways and forms, but for instance in the case of the David-Prasad protocol, even a simplistic approach works quite nicely.

The procedure is the following: for each authentication session eavesdropped, a number of good approximations to the secret values is computed, and then stored as rows of three different matrices. After eavesdropping a given number of sessions, the global values are computed simply by repeatedly adding each of the columns of the matrices, and returning a 0 if the total number of ones in the said column is below a given threshold, or a 1 otherwise. The simplest way to obtain a final value is to select the majority value in each column of this matrix. One can quickly sum all the rows to obtain a final vector. Then, if the value in a column of this vector is greater than half of the number of approximations times the number of eavesdropped sessions, a 1 is conjectured in that column, or a 0 otherwise.

In the case of the David-Prasad protocol, this easy and efficient way of combining approximations works surprisingly well for producing accurate global approximations to all three secret values after eavesdropping a relatively small number of authentication sessions. Results are powerful enough to consider the protocol completely broken, as after observing only 5 or 10 sessions, an attacker can guess about 80 out of 96 bits of the keys and static identifier, as shown in Figure 2.6. The remaining ones can be easily identified by an offline brute force search, or by eavesdropping more sessions. These results contradict the security properties claimed in [67].

After conducting the attack, the adversary is able to retrieve all the secret information shared between the tag and the server, so she can trivially bypass any authentication mechanisms (i.e. tag and reader authentication) and impersonate the tag in the future, or just clone it. Confidential information is put at risk and tag's answers can be tracked even though two random numbers are used in each session. A desynchronization attack against the tag (or the sever) is also quite straightforward, since the adversary can generate any desired valid synchronization messages.

In [39], a technique to automatically find new good approximations based on the use of genetic programming is described. This further automates the whole Tango attacks, and makes it suitable for testing new proposals in a black-box manner.

Heuristic Simulation Attacks

The concept behind heuristic simulation attacks, again one of the few general-purpose published attacks against ultralightweight protocols, is very simple: To infer all or part of the secrets used in a protocol by extracting information from the proximity between the exchanged messages in a real protocol run and those generated by an approximation to the secrets.

These attacks generally start from a randomly generated set of secret values. Then, by applying some heuristic techniques (like simulated annealing) it tries to approximate the true value of the secrets. For that it employs a fitness function that is generally a function of the distance between generated messages and the eavesdropped ones.

Different degrees of success in this approximation could lead to a key recovery attack (when this works well and recovers multiple secret bits) or to a traceability attack (when only some parts of the secrets can be recovered). Of course, a number of different heuristic techniques can be used ranging from hill climbing or simulated annealing to genetic algorithms. The fitness function is generally a measure of the distance between two sets of messages, but can be quite more complex, like weighing some messages more than others, etc.

The first application of this approach was in [102, 103], which presented a traceability attack against SLMAP [128]. Authors showed a new metaheuristic-based traceability attack on SLMAP and analyzed its implications. The main interest of their approach is that it is a complete black-box technique that does not make any assumptions on the components of the underlying protocol and can thus be easily generalized to analyze many other proposals.

The main idea behind this approach is to transform the cryptanalysis of a security protocol into a search problem, where a large number of different search metaheuristics can be applied. In general, during this search one tries to find which are the secret state values (keys, nonces, etc.) of some subset of the parties involved in the protocol.

This could be done in various ways, but the most natural approach is to measure the cost of the tentative set of secret values by the proximity of the messages produced by these tentative solutions to the real public messages generated and exchanged during the actual protocol execution.

Most cryptographic protocols should exchange one or more messages to accomplish their intended objective(s) (authentication, key exchange, key agreement, etc.), and in the vast majority of cases these messages are sent via an insecure or public channel that can be easily snooped.

In this attack model, the cryptanalyst generally tries to infer the secret values that the two parties intend to hide by exploiting the knowledge of the exchanged messages. In a robust, secure, well-designed cryptographic protocol, even states that are very close to the real state should not produce messages that are very close (for any useful distance definition) of the real public messages.

This should be done, typically, by means of a careful design and message construction based on the use of some highly-nonlinear cryptographic primitives such as block ciphers or hash functions.

Unfortunately, new proposals in the field of lightweight cryptography, which are intended towards very computationally constrained environments (such as low-cost RFID systems) cannot use classical cryptographic primitives.

Then, for this search problem, a number of metaheuristic techniques can be used to minimize the distance between the candidate and the real exchanged messages. In [103], the authors used a Simulated Annealing technique, which is a metaheuristic technique that is extremely efficient and has some ability to avoid becoming quickly trapped in local minima.

With this, the next IDS can be predicted with sufficient accuracy despite not knowing the protocol secrets, and hence to mount a successful traceability attack. One important characteristic of their attack is that it was successful after eavesdropping only one authentication session, which is a very economic requirement compared with those of other passive attacks.

The general traceability attack algorithm is described in Algorithm 2.

Algorithm 2 Metaheuristic traceability attack against SLMAP.

```
Snoop an SLMAP run
$Known \leftarrow IDS_n, A, B, C, D$
repeat $k$ times
    Start a Simulated Annealing to minimize $f_S$
    Run SLMAP over the $Known$ values
    Compute approximation for $IDS_{n+1}$ and store in $ListIDS$
end
$MajIDS \leftarrow$ majority vector of $ListIDS$
Get $IDS_0$ and $IDS_1$, candidate values for $IDS_{n+1}$
$\rho_0 \leftarrow$ correlation between $MajIDS$ and $IDS_0$
$\rho_1 \leftarrow$ correlation between $MajIDS$ and $IDS_1$
if $\rho_0 > \rho_1$ then
    output 0
else
    output 1
end if
```

This attack is efficient and effective, with a quite high success probability (around 95%) for a number of trials $k = 50$ as shown in Table 2.4.

Both attacks are quite simple, powerful, and attractive due to their generality and black-box potential.

New ultralightweight protocol proposals could benefit strongly to be tested against both attacks before publication, in order to avoid major pitfalls.

Resilience against both attacks does of course not prove that the security of the new protocol is foolproof, but at least will offer guarantees that the scheme has avoided some of the trivial security weaknesses that plague most of current proposals.

2.4 Dubious Proofs of Security

This section discusses some of the many dubious security proofs that different authors have used over the years in an attempt to prove the security of their proposals.

2.4.1 Randomness Tests

As shown in Figure 2.5, one of the first proposals in this area, LMAP, tried to prove some degree of security by verifying that the exchanged messages looked random enough. For that, multiple sessions of the protocol were run and the exchanged messages recorded and later analyzed with different randomness test batteries such as the well-known ENT [185], Diehard [135] and NIST [167].

The results were indeed impressive, with the messages used for running the protocol passing all tests with flying colors. However, this does not prove any security level, as LMAP (depicted in Figure 2.4) was broken shortly afterwards. This simple methodological approach presents a number of limitations. First, the most obvious one is that two fresh random numbers generated by the reader, n_1 and n_2, are injected into the messages, and this makes it hardly surprising that some of the exchanged messages, that are combined with these two random messages, look random. Randomness might appear, then, not as a consequence of a well designed protocol but just as a result of employing these sources of randomness repeatedly during message composition. Consider this artificial, very simple, and Trivially Weak authentication protocol, depicted in Figure 2.7.

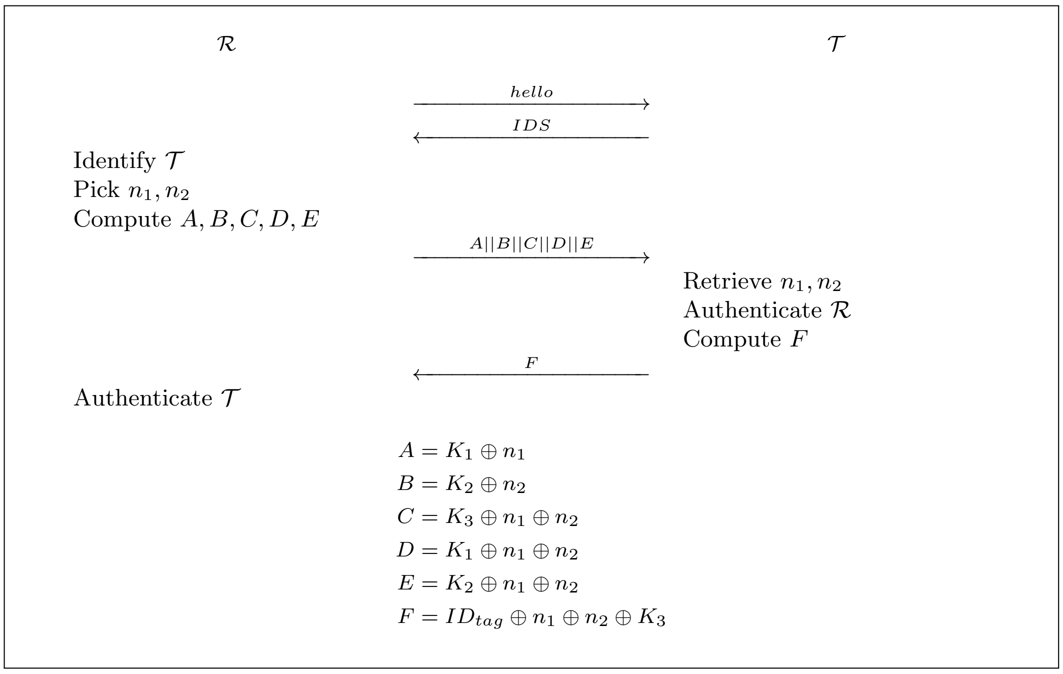
It is clear that, in this case, an attacker can trivially recover the values of all secrets involved

Table 2.4: Attacks results for 20 runs, after observing 1 auth. session, from [103].

Exp.	**Good Approx.**	**Bad Approx.**	**Correct bits**	**Corr.**	**Rand. Corr.**	**Exp. Result**
1	46	4	60	0.23591	-0.00532	Success
2	37	13	60	0.24760	-0.08020	Success
3	36	14	53	0.10418	0.02094	Success
4	39	11	56	0.13681	-0.01610	Success
5	44	6	59	0.22518	-0.03496	Success
6	34	16	53	0.10919	-0.12903	Success
7	41	9	56	0.17157	0.12330	Success
8	43	7	62	0.29247	0.08456	Success
9	45	5	60	0.24967	-0.06447	Success
10	42	8	58	0.19593	0.19375	Success
11	44	6	58	0.21009	0.09940	Success
12	35	15	53	0.10707	-0.11736	Success
13	38	12	53	0.09923	-0.02172	Success
14	47	3	61	0.026238	-0.04895	Success
15	35	15	53	0.10629	0.14811	Fail
16	47	3	66	0.36751	-0.16724	Success
17	44	6	59	0.21093	0.03115	Success
18	39	11	55	0.14885	0.08340	Success
19	31	19	52	0.10013	-0.01543	Success
20	39	11	58	0.20798	0.03845	Success
Mean	**40.3**	**9.7**	**57.25**	**0.19**	**0.01**	**95% Success**

Table 2.5: Randomness tests over the set of LMAP exchanged messages.

	A	*B*	*C*	*D*
Entropy (bits/byte)	7.999999	7.999999	7.999999	7.999999
Compression Rate	0%	0%	0%	0%
χ^2 Statistic	250.98 (50%)	255.71 (50%)	244.46 (50%)	255.18 (50%)
Arithmetic Mean	127.5062	127.4977	127.4946	127.5030
Monte Carlo π Estimation	3.1413 (0.01%)	3.1417 (0.0%)	3.1417 (0.0%)	3.1413 (0.01%)
Serial Correlation Coeff.	-0.000040	0.000010	-0.000077	-0.000036
Diehard Battery (p-value)	0.227765	0.775516	0.641906	0.410066
Nist Battery	✓	✓	✓	✓



$\mathcal{R}$
$\mathcal{T}$
$hello$
IDS
Identify $\mathcal{T}$
Pick n_1, n_2
Compute A, B, C, D, E
$A||B||C||D||E$
Retrieve n_1, n_2
Authenticate $\mathcal{R}$
Compute F
F
Authenticate $\mathcal{T}$
$A = K_1 \oplus n_1$
$B = K_2 \oplus n_2$
$C = K_3 \oplus n_1 \oplus n_2$
$D = K_1 \oplus n_1 \oplus n_2$
$E = K_2 \oplus n_1 \oplus n_2$
$F = ID_{tag} \oplus n_1 \oplus n_2 \oplus K_3$

Figure 2.7: The Trivially Weak protocol, for demonstration purposes.

simply by solving the trivial equations involved. Indeed,

$$\begin{aligned}
n_2 &= A \oplus D \\
n_1 &= B \oplus E \\
K_1 &= A \oplus B \oplus E \\
K_2 &= B \oplus A \oplus D \\
K_3 &= A \oplus B \oplus C \oplus D \oplus E \\
ID_{tag} &= F \oplus C
\end{aligned}$$

Nevertheless, even in this very simple and weak protocol, exchanged messages pass all randomness statistical tests[8]. At the very least, to prevent the indirect measure of the randomness of the source of n_1 and n_2 one has also to examine in detail all the correlations between these exchanged messages.

So if by now it seems clear that randomness of the exchanged messages is not a sufficient condition, it should also be clear that it is neither a necessary one. Another trivial way of showing this is by thinking about highly formatted messages and how, even if a protocol is secure, due to formatting and padding of some or all of its messages these may not pass some randomness test.

Intuitively, this shows that randomness of the exchanged messages is neither a sufficient nor a necessary condition for protocol robustness, and as such it should no longer be used to prove any kind of security level in future proposals.

2.4.2 BAN and GNY Logic

Some authors have tried to use BAN logic to prove the security of their proposals. A notable example is [163]. Needless to say, this approach did not work out as intended and, despite being accompanied by a formal security proof in BAN logic the proposal was broken shortly afterwards in [153]. In the particular case of [163], the authors mistakenly employed CRC (Cyclic Redundancy Codes) as recommended by the EPC-C1-G2 standard, but instead of using them as simple error detection tool, they employed them for encryption. In their idealized model, they identified their

[8]Provided, of course, that the (P)RNG used for creating n_1 and n_2 is good enough.

CRC usage as equivalent to encryption, so some of the BAN logic rules (for example *R1: Message-meaning rule*) do not hold any more. This is a particularly flagrant mistake because CRCs are linear and should never be used for encryption, hashing or any other cryptographic purposes. But it is also a very common error, as never an idealized scenario like the one modelled by BAN logic (with perfect, unbreakable and zero-leaking ciphers) accurately models reality. The level of abstraction needed in the modelling phase basically makes it impractical for most realistic situations. This is, unfortunately, not only a limitation of BAN logic but, to different extents, is also in most formal models (GNY, etc.) which makes them not very relevant for our purposes.

A recent example of the current and continuous use of these quite limited approaches is [166], where again BAN logic was used again to proof the security of the proposal, despite its severe limitations and unrealistic assumptions. Another recent case can be found in Section 2.5.3, where a protocol published in Nov. 2013 uses GNY logic to prove its security. Needless to say, this proposal can be attacked quite straightforwardly as mentioned in 2.5.3.

2.4.3 Other Approaches

Recently, some authors are taking a different approach to this problem, by using automatic analysis tools previously employed on the formal analysis of classical protocols. Two interesting examples are [108], and [177].

In the first, the author uses the AVISPA [5] tool to analyze the security of the LMAP protocol. This protocol, being the first of its kind has many vulnerabilities (some of them having been discussed in Section 2.3), so the conclusion of this work are to be taken lightly, as apparently only two attacks have been discovered by AVISPA and the author proposed an easy patch.

Although these results have to be examined with more caution, it seems that the general approach is promising and that better proposals will appear if using the AVISPA and similar tools becomes commonplace in the area.

In [177], the authors propose a key establishment and derivation protocol for EPC Gen2 tags, and employ model checking techniques to verify whether the security properties needed hold in a finite state machine. For that, they use automated reasoning, specifically the Constraint-Logic based Attack Searcher (CL-AtSe), and the HLPSL input language.

This should be an approach to encourage in the future, with any new proposal having been tested by automated tools prior to acceptance. Of course attacks could still exist after this check, but this check prevents falling flat on well-known mistakes.

2.5 Weaknesses in Recent Protocols

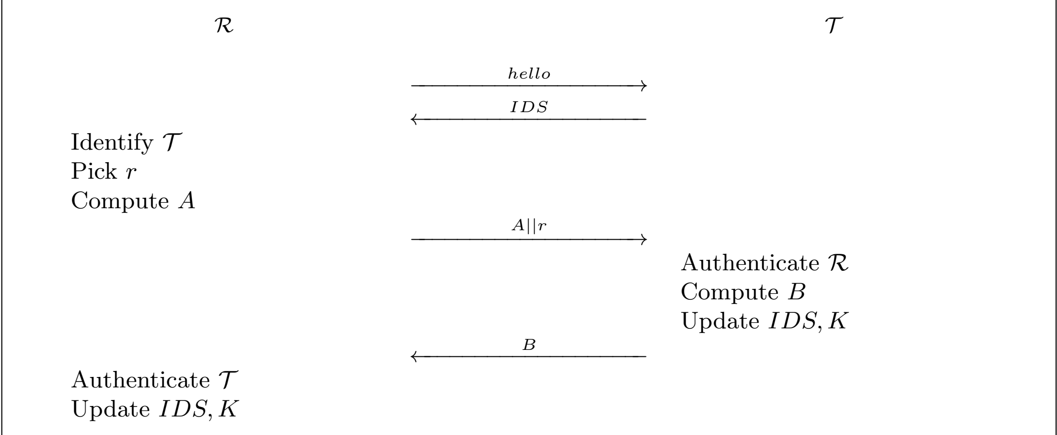
In this section, recently published protocols are presented along with weaknesses in their design, illustrating the discussions of Sections 2.3 and 2.4.

2.5.1 Bilal and Martin

Bilal and Martin proposed in 2013 a protocol called UMAP [43]. The protocol definition is depicted in Figure 2.8. It uses modular rotations, so a natural first attempt at analyzing its security is to look at what happens when they behave like the identity. This happens with a non-negligible probability of L^{-2}. The relevant equations of the protocol become:

$$\begin{aligned} A &= n_2 + IDS + K + ID + n_1 \\ B &= n_1 + IDS + K + ID + n_2 \\ IDS^{\text{next}} &= n_1 + IDS + n_2 \\ K^{\text{next}} &= n_2 + K + n_1 \end{aligned}$$

Note that in particular, we have that $A = B$. Therefore, with probability L^{-2}, an attacker can authenticate in place of a tag, just by knowing the IDS of some target tag (this may be obtained beforehand simply by sending hello to the tag), and by replicating A as sent by the reader. This attack vector is addressed in Section 2.3.3.



$$\begin{aligned}
n_1 &= f(K, r) \\
n_2 &= f(r, K) \\
A &= rot(rot(n_2 + IDS + K + ID, n_1) + n_1, n_2) \\
B &= rot(rot(n_1 + IDS + K + ID, n_2) + n_2, n_1) \\
IDS^{\text{next}} &= rot(rot(n_1 + IDS, n_1) + n_2, n_2) \\
K^{\text{next}} &= rot(rot(n_2 + K, n_1) + n_1, n_2)
\end{aligned}$$

Figure 2.8: Bilal and Martin's protocol.

Another point regards the function f. It is described in [43] as a "lightweight pseudo random function." If such a function is readily available on the tag, and if it has good security properties, then why bother with the other constructions ? As mentioned in Section 2.2, a very basic challenge-response protocol, such as the following, works just as well:

$$\begin{aligned}
R &\to T : \text{hello} \\
T &\to R : IDS \\
R &\to T : r, f_K(r) \\
T &\to R : f'_K(r)
\end{aligned}$$

2.5.2 Jeon and Yoon

Jeon and Yoon presented in 2013 their protocol RAPLT [109]. This protocol (shown in Figure 2.9) introduces two new lightweight operations: $Mer(A, B, K, C)$ and $Sep(C, K, A, B)$. Their definition is the following. $Mer(A, B, K, C)$ merges the values $A \in \{0,1\}^L$ and $B \in \{0,1\}^L$ into $C \in \{0,1\}^{2L}$ using the bits of $K \in \{0,1\}^{2L}$: if a bit in K is 0, then a bit of A is moved to C, else a bit of B is used. $Sep(C, K, A, B)$ does the opposite: if a bit in K is 0, the next bit in C is moved to A, else it is moved to B.

The first thing to note is that these operations are ill-defined as they both require that $\mathcal{H}(K) = L$. The authors make no mention of any way to guarantee that, and it would anyway result in a smaller entropy. Regardless, it is assumed that this property is satisfied in what follows.

The usage of Sep in the protocol does not follow the definition. In the protocol, it is specified that $Sep(M_1, M_2, K_2||K_1, B_1||B_2)$ is used. Instead, the authors presumably rather meant $Sep(M_1||M_2, K_2||K_1, B_1, B_2)$, since otherwise the variables would not have the right sizes.

Another point is that these operations, that consist in rearrangements of bits of some of their inputs, have linear properties. The only other operation used is the XOR, which makes the protocol entirely linear. This problem is addressed in Section 2.3.1.

Finally, a traceability attack may easily be mounted against the protocol by using the fact that the two new operations are Hamming-weight invariant. A very similar attack was previously done

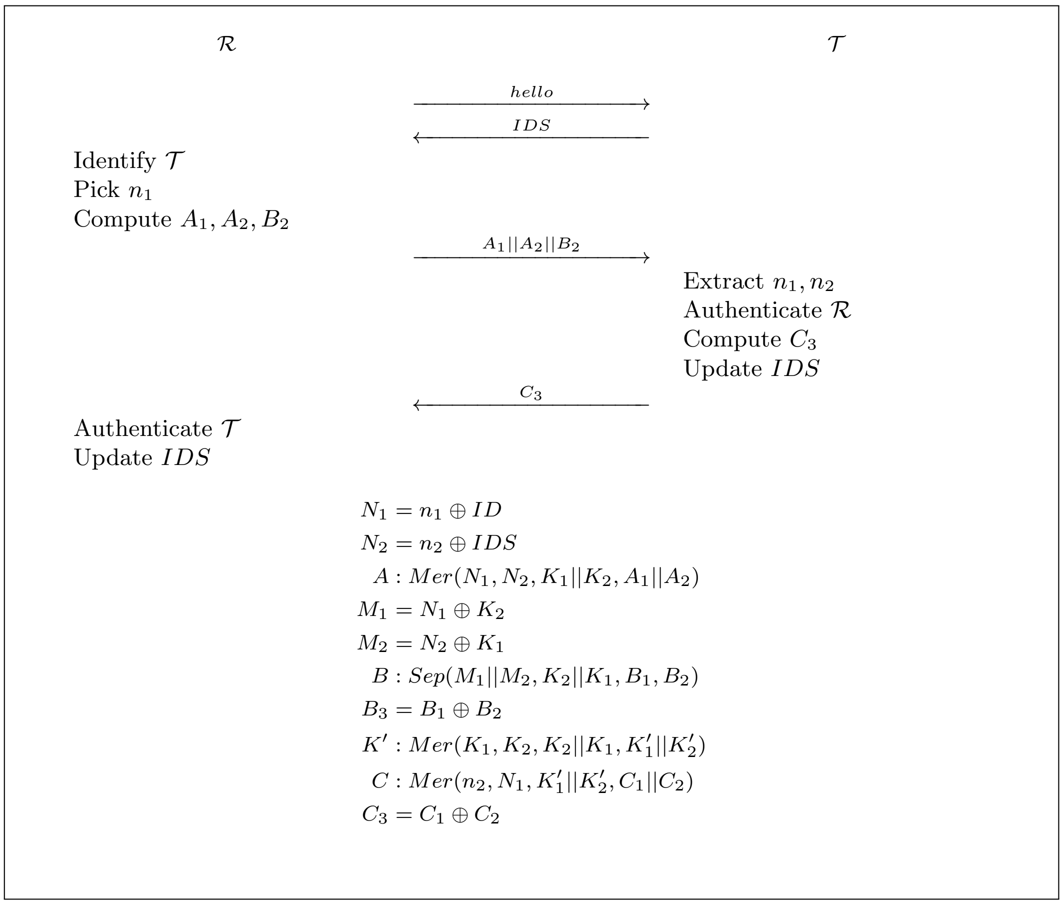


Figure 2.9: Jeon and Yoon's protocol.

on RAPP [176] in [16]. We have that, after $Mer(A, B, K, C)$, or after $Sep(C, K, A, B)$,

$$\mathcal{H}(A) + \mathcal{H}(B) = \mathcal{H}(C) = \mathcal{H}(C_1) + \mathcal{H}(C_2).$$

A property of the Hamming weight is that

$$\mathcal{H}(A \oplus B) \equiv \mathcal{H}(A) + \mathcal{H}(B) \pmod 2,$$

for any A, B. Therefore,

$$\begin{aligned}\mathcal{H}(B_3) &\equiv \mathcal{H}(B_1) + \mathcal{H}(B_2)\\ &\equiv \mathcal{H}(M_1) + \mathcal{H}(M_2)\\ &\equiv \mathcal{H}(N_1) + \mathcal{H}(N_2) + \mathcal{H}(K_2) + \mathcal{H}(K_1) \pmod 2.\end{aligned}$$

Moreover,

$$\mathcal{H}(A_1) + \mathcal{H}(A_2) \equiv \mathcal{H}(N_1) + \mathcal{H}(N_2) \pmod 2.$$

Therefore,

$$\mathcal{H}(K) \bmod 2 = (\mathcal{H}(B_3) + \mathcal{H}(A_1) + \mathcal{H}(A_2)) \bmod 2$$

Since A_1, A_2 and B_3 are public, and since K is static, it is easy for an attacker to trace a tag by observing its communications with a genuine reader.

2.5.3 Ling and Shen

Ling and Shen present in [133] a protocol that is strongly inspired by the SASI protocol [64]. A formal proof using GNY logic is made, which should indicate that the protocol is secure.

However, since messages C and D are exactly the same ones as in SASI, Phan's simple traceability attack [161], which allows a passive attacker to recover one bit of the secret, works here too. Without going into the details, it essentially relies on the fact that the OR ($\vee$) operation has biased output, an issue that has been covered in Section 2.3.2.

This attack proves that the formal security proof made in [133] is not valid, a point that was already raised in Section 2.4.2.

2.6 Ultralightweight Building Blocks

Rather than designing an authentication protocol from scratch using ultralightweight operations, another approach to designing an ultralightweight authentication protocol is to use a classical challenge-response authentication protocol, and using ultralightweight constructions as cryptographic primitives.

In the category of ultralightweight primitives, that goes up to around 1.5K GE, there is generally no place for highly optimized versions of standard ciphers like AES and IDEA, or slight modifications of classical block ciphers like DES (DESL, DESXL) [120].

Only newly and purposefully designed low-cost primitives can be ascribed to it. Fortunately for the area, more and more of the new proposals fall easily within this limit, as there seems to be growing consensus this is about the maximum GE that many devices would be able to afford to devote to security.

Within this limit, there was hardly any valid proposal before the publication of PRESENT [51] in 2007, as both DES (2300-3000 GE) or even DESXL (2168 GE) are too large. AES needs even more gates with around 3400 GE (recent results [79] put this at 2400GE, still well above the 1.5K limit). Not even well-known extremely efficient and easy to memorize proposals such as TEA [188] (2100) and XTEA (2000) were light enough.

Other more recent proposals like MCRYPTON [131] (2949 GE), HIGHT [105] (3000) and SEA [171] (2280) were also not adequate for our purposes. After 2007, on the other hand, there has been a blossoming of new primitives, including most notably PRESENT (1570 GE), based on the AES finalist Serpent, which can fairly be considered a pioneer in this area.

The European eSTREAM project was also relevant, as it looked for stream ciphers with efficient hardware implementation, and it contributed to the development of Trivium (2599 GE) and Grain (1294 GE).

Other ultralightweight block cipher are LBlock (1320 GE), TWINE (1116 GE), MIBS (1400 GE), KLEIN [87] (1478 GE), KATAN/KTANTAN [69] (from 688 GE), Piccolo (683 to 1043 GE), LED [91] (1040 GE) and the PRINTcipher (503 GE) or PRINCE.

From this long list, specially notable are the KATAN, LED and KLEIN proposals which have gained respect and popularity after resisting attacks for a time. Although all have received some minor attacks, these look at the moment like a particularly good trade-off between security and efficiency.

It is important to note that the gate count needs to be taken carefully into consideration, as not all of these algorithms operate over the same key size, block size or produce the same throughput.

Lastly, even the NSA came with two proposals in this area, called SIMON (763 GE) and SPECK (884 GE), meant to provide high performance across a range of devices. In the light of recent NSA scandals these two proposals, independently of their respective merits, will probably never prove very popular within the cryptography community. On top of that, there are some attacks [180] that present a not particularly rosy view on them.

Not only block and stream ciphers designers have been attracted to the challenge of devising ultralightweight primitives, as some interesting though less numerous construction in other areas of cryptography have been proposed. Particularly interesting are the SPONGENT [50] (from 738 GE to 1950) and QUARK [8] (from 1379 to 4460 GE) families of lightweight hash functions. Despite being by far more popular, they are complemented by the PHOTON [90] family (1122 to 2768 GE).

Note that all these proposals, despite being analyzed much more deeply than protocols discussed in this chapter, are still under scrutiny and are continuously improved. In particular, they are not considered safe under fault attacks or side-channel analysis [37, 189].

2.7 Conclusion

The need for inexpensive tags capable of secure, efficient, and privacy-friendly authentication has prompted a significant part of the community to develop lightweight protocols. This endeavor has been the focus of many researchers since the early days of RFID security in 2005, and is still a recurring topic. The data provided in Section 2.2 speaks for itself regarding the existing proposals (virtually all of them are broken, with half of the protocols being broken within 8 months of their publication, and most of them within a year), and it was shown in Section 2.5 that little effort is often sufficient to break new schemes as well.

The problems discussed in this chapter indicate clearly that the current approach for designing these protocols is wrong.

One possible alternative approach is using block ciphers or hash functions issued from the lightweight cryptography community, such as PRESENT, on top of a classical challenge-response authentication protocol. This approach is arguably more reliable, and is probably the best compromise today. There are a few security issues with existing lightweight building blocks, but it is an active area of research, and it is scrutinized by experienced cryptographers. There are also privacy and efficiency considerations regarding classical challenge-response protocols (as discussed in Chapters 3 and 4). One may finally argue that, although it is of course possible to achieve authentication using general-purpose block ciphers or hash functions, they may be excessive and in particular simply too expensive.

There has been other attempts to design an ad-hoc authentication protocol for RFID, such as the notorious HB$^+$ (introduced in 2005 by Juels and Weis [112] and based on HB, a secure identification scheme for Human Beings designed by Hopper and Blum in 2001 [106]). HB$^+$ requires only lightweight calculations from the prover and the protocol benefits from a security proof based on a reduction to the *Learning Parity with Noise* (LPN) problem. In spite of attractive properties, HB$^+$ and its family (the protocol spawned numerous fixes and variants) are not secure, and moreover considered not so lightweight by some experts, in particular not fitting within the 1500 GE margin (see [6] for a thorough discussion on the subject).

Chapter 3

Complexity and Privacy

Articles related to this chapter

[13] Gildas Avoine, Muhammed Ali Bingol, Xavier Carpent, and Siddika Berna Ors Yalcin. Privacy-friendly authentication in RFID systems: On sub-linear protocols based on symmetric-key cryptography. *IEEE Transactions on Mobile Computing*, 12(10):2037–2049, October 2013.

Privacy is a major concern for RFID protocols, as was highlighted in Chapter 1. The speed of authentication is also very important for RFID applications. The example that embodies this problematic is public transportation. Authentication needs to be fast, with a simple swipe of a tag being sufficient to identify and authenticate a user. It is generally agreed upon that about 200 milliseconds can be devoted to grant or deny access to a customer [66].

Combining privacy protection with an efficient identification procedure is a challenging task, and is the subject of a significant part of the literature in RFID authentication.

This chapter reviews the associated literature, categorizes protocols according to common features, analyzes them, compares their properties and discusses about which can be considered as the best ones to date. Many new attacks on several of these protocols are provided, as well as some patches. Note that low-level criteria such as gate count or power consumption of tags are not considered in this chapter, as the cost of building blocks was the focus of Chapter 2.

Section 3.1 presents some preliminary solutions that contextualize the problematic. Section 3.2 presents some protocols trading away privacy in order to achieve faster authentication. Sections 3.3, 3.4, and 3.5 present three families of protocols (respectively with shared secrets, based on hash-chains, and based on counters) dedicated to the scalability issue, along with discussions and attacks on them. These protocols are then compared in Section 3.6, and the chapter is concluded in Section 3.7.

3.1 Preliminaries

This Section looks at the naive solutions for the scalability problematic. In what follows, the variable N refers to the number of tags in a system.

Challenge-response Protocols

The ISO-9798 (Figure 3.1) defines challenge-response authentication protocols, which are commonly used in RFID. These are used in the MIFARE Classic for instance[1]. Other standards are also in

[1]The authentication protocol of the MIFARE Classic is based on the ISO-9798-2.3, a mutual authentication protocol using a stream cipher.

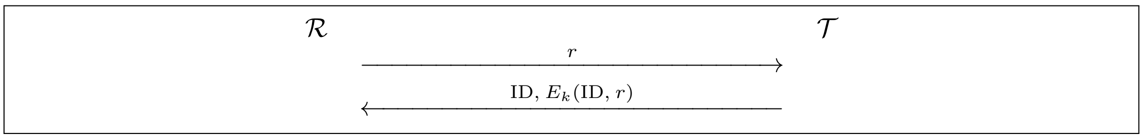


Figure 3.1: The ISO 9798-2.2 challenge-response protocol.

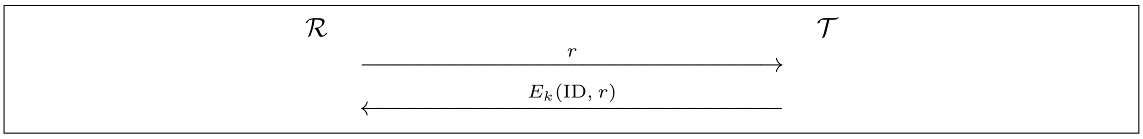


Figure 3.2: Variant to the ISO 9798-2.2 protocol that has good privacy but is inefficient.

application, such as the ISO-11770 (a protocol with key agreement), used for example in the Basic Access Control of e-passports.

In the example of Figure 3.1, the ID of the tag is sent in the clear. This allows the reader to identify the tag, find the corresponding key k and authenticate the tag. This solution is very efficient and secure, but there is no privacy since the identifier is sent in the clear.

On the other hand, when the tag does not directly reveal its identifier (as depicted in Figure 3.2), the reader has to try all the possible keys while decrypting the message, and check if it corresponds to the nonce he sent to authenticate the tag (there is no identification prior to authentication). This solution has good privacy and security, but takes $O(N)$ cryptographic operations, which is inefficient in large systems. In the following, this is referred to as a "linear protocol."

Finally, if the identifier is not sent in the clear, but there is only one key in the system (common to all tags), identification is $O(1)$ and the privacy is respected. However, an adversary compromising a single tag and retrieving its key breaks the whole system (she can mount not only traceability attacks, but above all impersonation attacks, targeting any tag in the system). This approach, dubbed the "master key" solution, is another extreme case of the privacy-security-efficiency trade-off.

The rest of this chapter focuses on protocols designed to reduce the complexity of the identification, while trying to preserve privacy, and which are distinct from these three extreme cases. Protocols such as the HB family ([54, 74, 93, 106, 112, 141]), or protocols such as [44, 60] are therefore not considered below.

Public-key Cryptography for RFID

Public-key cryptography (PKC) seems to be a solution to the identification problem stated above. The randomized Schnorr protocol [55], for instance, uses public-key encryption to provide both strong privacy and constant-time identification.

However, PKC is expensive, being in terms of gates required on the tag, or of time and especially energy necessary to perform the computations on a tag. Although some recent studies point otherwise (see, e.g., [95, 107, 123]), it is generally acknowledged that PKC is not affordable on low-cost tags, and most of the proposals for authentication in RFID use symmetric-key building blocks. Hopefully, further research in that area will improve the feasibility of PKC for low-cost RFID, but there will always be a market for symmetric-key solutions. For these reasons, only symmetric-key schemes are considered in what follows.

Note on Protocol Names

Some of the protocols analyzed in this chapter were named differently by their authors in different publications. To avoid confusion, Table 3.1 presents the matches between the protocols proposed with different names. The papers and publication years are given in the first row. Each other row

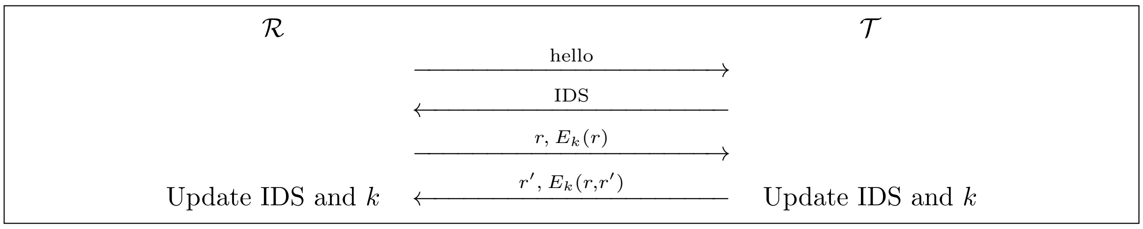


Figure 3.3: Hypothetical pseudonym-based challenge-response protocol.

represents one protocol, showing names given in each paper. In what follows the name used is the one that appeared most recently (shown in bold in Table 3.1).

Table 3.1: Matching of the names of some protocols

[178]	[57]	[179]	[58]
2006	2006	2007	2009
YA-TRAP	-	YA-TRIP	**RIP**
-	-	YA-TRAP	**RIP+**
-	YA-TRAP+	-	**RAP**
-	O-TRAP	-	**O-RAP**
-	-	-	**O-RAKE**
-	-	**YA-TRAP***	-
-	-	**YA-TRAP*& fwd**	-

3.2 Protocols with Limited Privacy

This section briefly discusses protocols trying to achieve some practical privacy, although not being private in a strict sense.

By trying to lower the identification procedure complexity, some solutions also lower the privacy or the security considerably. For instance, one could imagine a very simple scheme where each tag has a limited amount of ephemeral pseudonyms (or "coupons"), using one each time a reader wants to authenticate it. This solution is both private and efficient, but has a limited lifetime and an adversary could perform denial-of-service attacks very easily. Juels proposes in [110] a similar protocol in which each tag loops through a sequence of secrets to authenticate itself to a reader, again providing efficiency, but limited privacy. Henrici and Müller proposes in [98] that tags communicate to the reader the number of failed authentication attempts since the last legitimate authentication. While this allows the reader to efficiently identify the tags, it also allows an adversary to trace them, as pointed in [9].

Ultralightweight protocols, such as the ones discussed in Chapter 2, share a pattern on exchanged messages. As illustrated in Chapter 2, virtually all these protocols are broken due to the nature of the building blocks used. One could however conceive a protocol following the same structure, but using regular cryptographic building blocks, such as the one illustrated in Figure 3.3. This protocol provides $O(1)$ identification, security[2], and some level of privacy. However, this protocol can not be completely private strictly speaking. Indeed, tag pseudonyms (IDS) change after each successful authentication, but an adversary is able to trace a tag between two genuine authentications by simply querying the tag and then terminating the authentication session. Since no update is performed, the pseudonym remains the same and the tag can be traced. This is captured by the notion of existential and universal untraceability in more detailed privacy models such as Avoine, Coisel and Martin's [23].

[2]It could be the target of desynchronization attacks such as described in Section 2.3.6, although countermeasures would work here too.

Note that despite the fact that these solutions are not private strictly speaking, there might be scenarios where they can be applied, since *some* privacy is better than none at all. The rest of this chapter however focuses on protocols aiming to achieve “perfect” privacy to some extent.

3.3 Protocols with Shared Secrets

Some recent protocols have the common feature that several tags in the system share their secrets (at least partially). They manage to lower the online complexity of the reader by storing tag secrets in a particular structure (a tree, a grid, etc.). While these protocols provide that very desirable property, and also bring new and interesting ideas, they all have traceability issues.

This section introduces Molnar and Wagner’s tree-based protocol [140], Alomair, Clark, Cuellar, and Poovendran’s protocol [4], Avoine, Buttyán, Holczer, and Vajda’s group-based protocol [15], and Cheon, Hong, and Tsudik’s meet-in-the-middle protocol [63]. Some attacks on these protocols are also discussed, especially new attacks against [63] and [4].

3.3.1 Tree-based and Group-based Protocols

As stated previously, privacy-friendly challenge-response protocols do not scale well: the reader must check $O(N)$ keys to authenticate a tag, where N is the total number of tags in the system.

Molnar and Wagner propose in [140] an approach that reduces the complexity from $O(N)$ to $O(\log N)$. The fundamental idea is to manage the tags’ keys in a tree structure instead of using a flat structure. More precisely, the tags are assigned to the leaves of a balanced tree with branching factor b at each level of the tree. Each edge of the tree carries a random key. Each tag stores the keys along the path from the root to the leaf corresponding to the given tag, while the reader stores the whole tree. During the authentication process, the reader performs one challenge-response per tree level in order to identify the sub-tree the tag belongs to. Each challenge-response requires from the reader an exhaustive search in a set containing b keys only. The overall reader’s complexity of the authentication is $b \log_b N$ in the worst case.

The significant complexity improvement due to Molnar and Wagner’s technique (MW) has however an unacceptable drawback: the level of privacy provided by the scheme is quickly decreasing when an adversary tampers with tags. Giving the adversary the ability to tamper with some tags makes sense because MW is useless without this assumption: in such a case, the same key can be stored in all the tags and the complexity problem no longer occurs. On the other side, giving to the adversary the ability to tamper with tags significantly degrades the privacy in MW.

Avoine, Dysli, and Oechslin raise this attack in [25] and evaluate the trade-off between complexity and privacy according to the branching factor. Buttyán, Holczer, and Vajda in [59] also identified weaknesses of MW and introduce an improvement with variable branching factors. Nohl and Evans in [146] provided another approach to analyze MW. Later on, Halevi, Saxena, and Halevi [93] present a lightweight privacy-friendly authentication protocol that combines Hopper and Blum’s HB protocol [106] and the tree-based key infrastructure suggested by Molnar and Wagner [140]. However, [106] inherits from the weaknesses of MW as demonstrated by Avoine, Martin, and Martin in [29]. Finally, Beye and Veugen further analyze the improvement of Buttyán *et al.* in [42].

One may also cite some other attempts to design tree-based protocols, e.g., [190] or the saga [1, 72, 134, 186]. However, as said previously out, tree-based secret sharing is definitely not suited when the adversary is capable of tampering with tags, and the tree structure is even not the best solution in that case. Indeed, Avoine, Buttyán, Holczer, and Vajda demonstrate in [15] that a simpler structure than the tree, namely when tags are grouped and each group share a same key, achieves a higher level of privacy and a better efficiency. Finding a better structure, that does not avoid the traceability problem but that mitigates it is still an open problem.

	K_1^1	K_1^2	K_1^3	$\dots$	K_1^i	$\dots$	K_1^n
K_2^1	$\mathcal{T}_{1,1}$	$\mathcal{T}_{2,1}$	$\mathcal{T}_{3,1}$	$\dots$	$\mathcal{T}_{i,1}$	$\dots$	$\mathcal{T}_{n,1}$
K_2^2	$\mathcal{T}_{1,2}$	$\mathcal{T}_{2,2}$	$\mathcal{T}_{3,2}$	$\dots$	$\mathcal{T}_{i,2}$	$\dots$	$\mathcal{T}_{n,2}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\ddots$	$\vdots$
K_2^j	$\mathcal{T}_{1,j}$	$\mathcal{T}_{2,j}$	$\mathcal{T}_{3,j}$	$\dots$	$\mathcal{T}_{i,j}$	$\dots$	$\mathcal{T}_{n,j}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\ddots$	$\vdots$
K_2^n	$\mathcal{T}_{1,n}$	$\mathcal{T}_{2,n}$	$\mathcal{T}_{3,n}$	$\dots$	$\mathcal{T}_{i,n}$	$\dots$	$\mathcal{T}_{n,n}$

Figure 3.4: Tags' secrets organized in a grid in CHT.

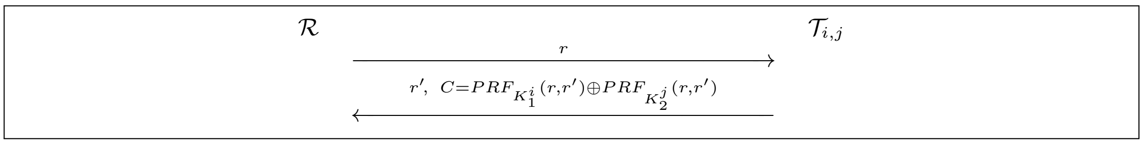


Figure 3.5: Cheon-Hong-Tsudik plain protocol.

3.3.2 Cheon, Hong, and Tsudik's Protocol

Description

The protocol proposed by Cheon, Hong, and Tsudik in [63] is an innovative proposal to reduce the reader complexity. It uses a *meet-in-the-middle* strategy, similar to the one used in several famous attacks on double-encryption schemes [71]. The idea is the following. During the initialization, the system chooses two sets of keys $\mathcal{K}_1$ and $\mathcal{K}_2$ such that $|\mathcal{K}_1| = |\mathcal{K}_2| = n$, where $N = n^2$ is the number of tags in the system, and $\mathcal{K}_1 \cap \mathcal{K}_2 = \emptyset$. It then initializes each tag $\mathcal{T}_{i,j}$ with a unique pair of keys $\langle K_1^i, K_2^j\rangle$, where $K_1^i \in \mathcal{K}_1$ and $K_2^j \in \mathcal{K}_2$, yielding an $n \times n$ grid in which each cell represents a tag, as depicted in Figure 3.4.

The identification procedure, represented in Figure 3.5, is as follows. The reader $\mathcal{R}$ first picks a nonce r and sends it to a tag $\mathcal{T}_{i,j}$ entering its field. The latter then picks another nonce r', and computes $C = PRF_{K_1^i}(r, r') \oplus PRF_{K_2^j}(r, r')$, where PRF is a pseudo-random function. The tag $\mathcal{T}_{i,j}$ then sends the pair $\langle C, r'\rangle$ to $\mathcal{R}$. In order to identify the tag, $\mathcal{R}$ computes $PRF_{K_1^x}(r, r')$ for $x \in [1, n]$, and then computes $C \oplus PRF_{K_2^y}(r, r')$ for $y \in [1, n]$, and tries to find a match between two values. This search requires $2n = 2\sqrt{N}$ PRF evaluations at worst, rather than N for a standard linear search[3]. An adversary eavesdropping r, r', and C however would have to search the entire key space, since she does not know the key sets $\mathcal{K}_1$ and $\mathcal{K}_2$.

The protocol presents an efficient search procedure, but is not synchronized (i.e., the tag has no *state* that changes over time). This implies that it does not provide any forward-privacy, because an adversary having compromised a tag gets its two keys, and can thus recompute messages previously produced by the tag, in this way "tracing" the tag in the past.

Moreover, the authors themselves identify an important issue. Indeed, when a tag is compromised, its two sub-keys are disclosed, but this does not leak any information on other tags' keys as the combination of subkeys is unique. However, when the adversary compromises several tags, she gains knowledge of key-pairs of legitimate tags. For instance, if the adversary compromises the tags $\mathcal{T}_{a,b}$ and $\mathcal{T}_{c,d}$, she also discovers the keys of the tags $\mathcal{T}_{a,d}$ and $\mathcal{T}_{b,c}$. These will respectively be referred as *directly compromised tags* and *indirectly compromised tags* hereafter (a *compromised tag* refers to either situation). Additionally, *partially compromised* refers to the tags for which only one key is known.

The authors describe an extension to mitigate this problem by introducing proper authentication in the protocol. In this extension, each tag has a third, unique subkey K_3. The key sets $\mathcal{K}_1$ and $\mathcal{K}_2$ have a size of N^α, with $0 \le \alpha \le \frac{1}{2}$ being a system parameter and $\mathcal{K}_3$ has a size of N, such that $N^{1-2\alpha}$ tags have the same $\langle K_1, K_2\rangle$ key-pair. The tag further computes $C' = PRF_{K_3}(r, r')$,

[3]Note that in [63], the authors state that the search is $O(\sqrt{N}\log N)$. Only cryptographic operations are considered in the online time, so $O(\sqrt{N})$ is used instead.

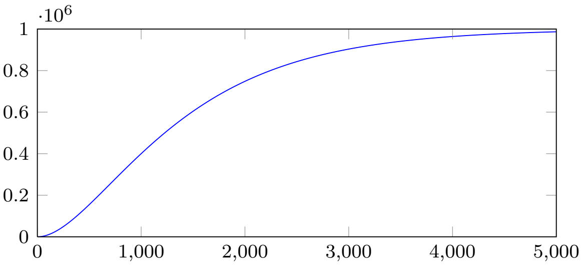


Figure 3.6: Average number of indirectly compromised tags in a system of $N = 10^6$ tags, with respect to an increasing number of directly compromised tags.

and sends it to the reader. After the usual search procedure, $\mathcal{R}$ checks the value C' to authenticate the tag. Since K_3 is unique to each tag, the impersonation attack is prevented, but there is still a traceability issue, as detailed in Section 3.3.2.

Impersonation Attack on the Plain Protocol

After having compromised some tags, an adversary can perform the following impersonation attack. She listens to a legitimate authentication session between $\mathcal{R}$ and $\mathcal{T}_{i,j}$. When $\mathcal{T}_{i,j}$ outputs (r', C), she blocks the message. She can now change C in order to authenticate another tag than $\mathcal{T}_{i,j}$. Because the protocol is stateless, $\tilde{C}$, the modified C, will be accepted (provided it is valid), and the corresponding tag will be identified. Two situations may occur for an adversary:

1. She wants to let a tag that is compromised be authenticated instead of $\mathcal{T}_{i,j}$.
2. She wants to let a tag that is partially compromised be authenticated.

In case 1, the adversary can replace the authenticating tag with another compromised tag, say, $\mathcal{T}_{a,b}$, by simply replacing C by $\tilde{C} = PRF_{K_1^a}(r,r') \oplus PRF_{K_2^b}(r,r')$. This problem was already highlighted in [63].

In case 2, the adversary must at least know one of the keys of $\mathcal{T}_{i,j}$ to succeed (i.e. $\mathcal{T}_{i,j}$ must be partially compromised). Suppose that the adversary knows K_1^i but not K_2^j, and that she also knows another key K_1^k. She can then replace C by $\tilde{C} = PRF_{K_1^k}(r,r') \oplus PRF_{K_2^j}(r,r')$ by computing $\tilde{C} = C \oplus PRF_{K_1^i}(r,r') \oplus PRF_{K_1^k}(r,r')$, and by doing so, authenticate $\mathcal{T}_{k,j}$, which is only partially compromised. Of course, she does not know the keys of the victim in advance, so the attack is probabilistic. She can thus iterate on all the tags for which she knows the secrets partially. A side-effect of this is that when $\mathcal{R}$ accepts the authentication, the adversary gets $PRF_{K_2^j}(r,r')$, which can lead to a traceability attack.

In [63], the authors state that, when compromising t tags, the number of indirectly compromised tags is $t^2 - t$. This is actually rather optimistic (from an attacker viewpoint) and only accurate when t is small. A more precise result is provided in Lemma 2. An example is presented in Figure 3.6.

Lemma 2. *Let T denote the number of* directly compromised *tags and S the total number of* compromised *tags (both directly and indirectly), that is the ones for which both keys are known. Then, the expected number of compromised tags given that t tags were directly compromised is:*

$$\mathbb{E}[S|T=t] = N\left[1 - \frac{2\binom{N-n}{t} - \binom{N-2n+1}{t}}{\binom{N}{t}}\right],$$

where $n = \sqrt{N}$. A similar result applies for the authentication extension, and S here denotes the number of compromised cells*:*

$$\mathbb{E}[S|T=t] = n^2\left[1 - \frac{2\binom{N-N/n}{t} - \binom{N-2N/n+N/n^2}{t}}{\binom{N}{t}}\right],$$

with $n = N^\alpha$.

Proof. Consider the plain protocol first. We have $\mathbb{E}\left[\#\text{ indirectly compr. tags}\right] = \mathbb{E}\left[\#\text{ compr. tags}\right] - t$, where a compromised tag refers to a tag that is either directly or indirectly compromised. R_i denotes the following random variable:

$$R_i = \begin{cases} 1 & \text{if at least one compromised tag has } K_1^i \\ 0 & \text{otherwise.} \end{cases}$$

Likewise,

$$C_i = \begin{cases} 1 & \text{if at least one compromised tag has } K_2^i \\ 0 & \text{otherwise.} \end{cases}$$

Note that the total number of compromised tags can be expressed as:

$$S = \left(\sum_{i=1}^{n} R_i\right)\left(\sum_{i=1}^{n} C_i\right).$$

Indeed, this number corresponds to the number of tags ("cells" in the grid) for which K_1 and K_2 are known. Therefore,

$$\begin{aligned} \mathbb{E}[S] = \mathbb{E}\left[\sum_{i=1}^{n}\sum_{j=1}^{n} R_i C_j\right] &= \sum_{i=1}^{n}\sum_{j=1}^{n} \mathbb{E}[R_i C_j] \\ &= \sum_{i=1}^{n}\sum_{j=1}^{n} \Pr(R_i = 1 \wedge C_j = 1). \end{aligned} \tag{3.1}$$

Note that R_i and C_j are not independent. However,

$$\begin{aligned} \Pr(R_i = 1 \wedge C_j = 1) &= 1 - \Pr(R_i = 0 \vee C_j = 0) \\ &= 1 - [\Pr(R_i = 0) + \Pr(C_j = 0) - \Pr(R_i = 0 \wedge C_j = 0)]. \end{aligned} \tag{3.2}$$

$\Pr(R_i = 0)$ is the probability that, after compromising t tags, none belong to the row i. That is, $\Pr(R_i = 0) = \binom{N-n}{t}/\binom{N}{t}$. Moreover, $\forall\, 1 \le i, j \le n$ we have $\Pr(C_j = 0) = \Pr(R_i = 0)$ since the grid is symmetric. Likewise, $\Pr(R_i = 0 \wedge C_j = 0) = \binom{N-2n+1}{t}/\binom{N}{t}$. Using (3.2) and the above in (3.1) gives:

$$\begin{aligned} \mathbb{E}[S|T = t] &= \sum_{i=1}^{n}\sum_{j=1}^{n} \Pr(R_i = 1 \wedge C_j = 1 | T = t) \\ &= \sum_{i=1}^{n}\sum_{j=1}^{n}\left[1 - 2\frac{\binom{N-n}{t}}{\binom{N}{t}} + \frac{\binom{N-2n+1}{t}}{\binom{N}{t}}\right] \\ &= N\left[1 - \frac{2\binom{N-n}{t} - \binom{N-2n+1}{t}}{\binom{N}{t}}\right]. \end{aligned}$$

The demonstration for the authentication extension is very similar to the above, except that cells contain $N^{1-2\alpha}$ tags. □

This result allows to quantify the probability of success of our attacks and confirms their feasibility.

Traceability Attack on Authentication Extension

Recall that in the authentication extension, the grid can now be seen as $N^\alpha \times N^\alpha$ "cells" of $N^{1-2\alpha}$ tags secrets. No two tags share the K_3 key, but each $\langle K_1, K_2\rangle$ is shared among $N^{1-2\alpha}$ tags. As the authors mentioned, this leads to a traceability issue because if an attacker knows a $\langle K_1^i, K_2^j\rangle$

pair, she can track $\mathcal{T}_{i,j}$ with probability $1/N^{1-2\alpha}$ by using the fact that there are $N^{1-2\alpha}$ tags with the same pair.

In this section, a more dangerous issue is given. It is assumed that the adversary has obtained the keys related to s cells. For the sake of simplicity, it is assumed that the compromised tags are put back into circulation. Since this number is supposedly small compared to N, the number of tags in the system, this is a reasonable assumption.

Let X denote the set of tags which secrets belong to one of the s cells known by the adversary. In a Juels and Weis game [113], when two tags $\mathcal{T}_0$ and $\mathcal{T}_1$ are presented to her, the adversary is asked to answer which of these tags is her target. Several cases occur:

- $E_1 = \mathcal{T}_0 \in X \wedge \mathcal{T}_1 \notin X$
- $E_2 = \mathcal{T}_0 \notin X \wedge \mathcal{T}_1 \in X$
- $E_3 = \mathcal{T}_0 \in X \wedge \mathcal{T}_1 \in X \wedge \langle K_1, K_2\rangle_{\mathcal{T}_0} \neq \langle K_1, K_2\rangle_{\mathcal{T}_1}$
- $E_4 = \mathcal{T}_0 \in X \wedge \mathcal{T}_1 \in X \wedge \langle K_1, K_2\rangle_{\mathcal{T}_0} = \langle K_1, K_2\rangle_{\mathcal{T}_1}$
- $E_5 = \mathcal{T}_0 \notin X \wedge \mathcal{T}_1 \notin X$

The obvious strategy for an adversary is, after choosing r, to query $\mathcal{T}_0$ and $\mathcal{T}_1$, and compare their answer with what would have answered the tags of which she knows the keys. If there is a match, then she identifies the tag and deduces its keys. In E_1 and E_2, only either of $\mathcal{T}_0$ and $\mathcal{T}_1$ is identified, and the adversary is able to determine correctly whether it is her target or not in all cases. If both tags are identified, the adversary succeeds only when they have a different key-pair (E_3, but not E_4). Finally, if neither is identified, the adversary is unable to tell her target apart in any better way than at random. Therefore, in the first three events, the adversary succeeds in the attack, and in the other two she fails. It is clear that the first two cases are symmetric:

$$\Pr(E_1) = \Pr(E_2) = \frac{NM - M^2}{N^2}, \tag{3.3}$$

where $M = sN^{1-2\alpha}$, that is the number of tags for which the adversary knows the secrets. Likewise,

$$\Pr(E_3) = \frac{M^2}{N^2}(1 - 1/s). \tag{3.4}$$

The overall probability that the adversary succeeds after corrupting s cells is thus

$$\begin{aligned}\Pr(E_1 \vee E_2 \vee E_3) &= \Pr(E_1) + \Pr(E_2) + \Pr(E_3)\\ &= 2\frac{M}{N} - \frac{M^2}{N^2}(1 + 1/s),\end{aligned}$$

because these events are mutually exclusive. This probability can become much higher than the one presented in [63]. For instance, in a system with $N = 10^6$ tags, configured with $\alpha = \frac{1}{3}$ (as suggested by the authors), an adversary having compromised $t = 300$ tags has roughly $s = 8750$ compromised cells (Lemma 2), and a probability of tracing a tag of roughly 0.984.

Discussion

Two important attacks on CHT were introduced. The first one regards the plain protocol and allows an adversary to change the tag being authenticated. The targeted tag need not be completely indirectly compromised, as a probabilistic approach can be carried out. The second attack regards the authentication extension, and allows an adversary to trace a tag.

The second attack is similar to the one [25] against MW. Although quite different technically, MW and CHT have in common the fact that tags share parts of their secrets. This property yields efficient tag identification, but tag compromising is dangerous since it jeopardizes the privacy of the whole system. Figure 3.7 is a comparison of the probability of tracing in CHT and MW protocols (with different values for the branching factor), in a system with $N = 10^6$ tags. Figure 3.8 is a comparison of their efficiency as a function of the number of tags in the system.

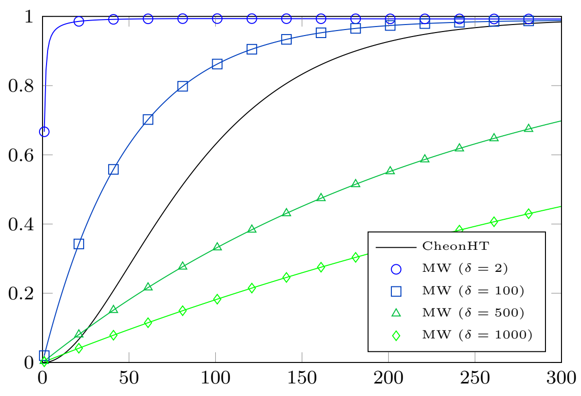


Figure 3.7: Probability of tracing a tag in CHT and in MW with respect to the number of compromised tags.

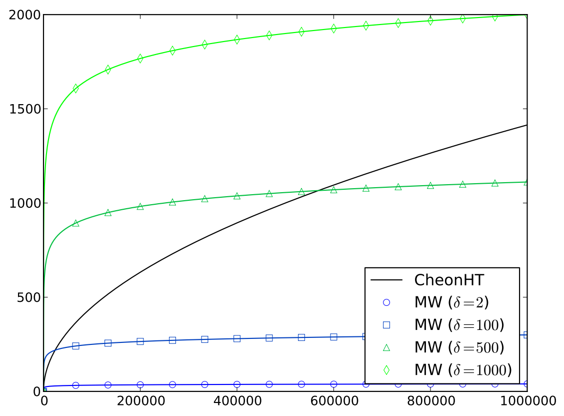


Figure 3.8: Average number of PRF execution on the reader side to authenticate a tag in CHT and MW, as a function of the number of tags in the system.

3.3.3 Alomair, Clark, Cuellar, and Poovendran's Protocol

Description

The protocol introduced by Alomair, Clark, Cuellar, and Poovendran in [4] provides Constant-Time Identification (CTI). This protocol is classified here in the shared-secret family in the sense that the system manages a pool of shared secret pseudonyms such that each tag is paired with a pseudonym for a while, and is reassigned to another one each time it is legitimately authenticated. Consequently, different tags may use the same pseudonym, at different times. Using re-usable pseudonyms was first introduced by Juels in [110] where each tag manages its own pool of pseudonyms and uses linear combination of them once all the pseudonyms have been used. However, tags do not exchange their pseudonym in [110], contrarily to [4].

During the set up phase, each of the N_T tags is assigned with a secret key k, a cycling counter c that is incremented modulo C each time the tag is queried (initially $c = 0$), and an initial pseudonym ψ drawn from a pool $\mathcal{E}$ of size $N > N_T$. A sketch of CTI is depicted in Figure 3.9 and we refer the reader to [4] for a detailed description.

The key-point of CTI is that each time a tag is legitimately authenticated, it releases its current pseudonym in order to get a new one from the reader randomly drawn from $\mathcal{E}$; it also updates its secret key k with the value $h(k)$ where h is a hash function. CTI provides constant time identification but this property is obtained after pre-calculation of all the NC possible answers from the tags. In that sense, CTI is not far from OSK [149], a protocol based on hash-chains that is analyzed in Section 3.4.1. The table of pairs (pseudonym, counter) in [4] is in some way similar to the table of pairs (identifier, counter) in [149]. A few differences can nevertheless be raised: (1) a denial of Service (DoS) occurs with OSK after M illegitimate authentications, while CTI is DoS-resistant; (2) OSK with authentication requires to compute between 3 (if there is no attack) and $2m+1$ hash calculations per identification, while CTI requires 4 hash calculations in any case; (3) CTI needs a larger memory than OSK and provides a lower privacy-resistance, as explained below. Note that both of them are resistant to timing attacks as stated in [22].

Intra-legitimate authentication Attack

The main drawback of CTI, already mentioned in [4] is the cycling counter because a tag can be easily tracked between two legitimate authentications if an adversary is able to query it C times. Indeed, recording each of the answers $h(0, \psi, c, k, r)$ $(0 \leq c < C)$, the adversary can definitely track the tag till the next legitimate authentication. This attack is especially meaningful when considering tags that are not frequently used, e.g., passports or tickets used for ephemeral event and kept by the customer as souvenir... Increasing C makes the attack harder, but this also significantly increases the memory consumption (and the reader's workload during the setup). This attack makes CTI not traceability-resistant in the Juels and Weis model [113].

Inter-legitimate authentication Attack

The pseudonyms used in the system are originally secret and can only be revealed in case of tampering attack. In such a case the current pseudonym of the compromised tag is revealed (and the secret key as well) but the adversary can also obtain additional pseudonyms by impersonating the tag in the system. This attack is mentioned in [4] but its analysis is refined here. Its impact should not be underestimated. First of all, the number of pseudonyms obtained by the adversary after tampering with only one tag is [4] $N\left(1-(1-1/N)^q\right)$, where q is the number of protocol executions[4]. Let $\mathcal{E}^q \subset \mathcal{E}$ the set of pseudonyms so revealed, the adversary can track a tag (even after legitimate authentications) as follows: in the learning phase as defined in the model of Juels and Weis [113], the adversary queries the targeted tag $\mathcal{T}_{\text{target}}$ once and so obtains a value $h(\psi_{\text{target}}, c_{\text{target}})$. Trying an exhaustive search on all values in $\mathcal{E}^q$ and all counter values, she obtains c_{target} if and only if $\psi_{\text{target}} \in \mathcal{E}^q$, which occurs with probability $|\mathcal{E}^q|/N$. In the challenge phase, given $\mathcal{T}_0$ and $\mathcal{T}_1$, the adversary must decide which one is $\mathcal{T}_{\text{target}}$. To do so, she applies the same technique and so possibly obtains c_0 and c_1. From c_0 and c_1, she could be able to decide which of $\mathcal{T}_0$ and $\mathcal{T}_1$ is $\mathcal{T}_{\text{target}}$. For example, if the adversary knows that her target is rather new while c_i ($i = 0$ or 1) is large, it may be safe to conclude that $\mathcal{T}_{\text{target}}$ is $\mathcal{T}_{1-i}$. To illustrate this attack, consider the following practical parameters: $N = 2N_T$, $N_T = 10^6$, $C = 10^3$, and $q = 10^3$. The probability to track a given tag is therefore 0.1%, assuming that one of the two tags only is rather new.

[4]Note that [4] suggests to limit the number of requests to a reader per tag, but bounding q to a value less than 1000 does not seem realistic in most applications as the adversary can avoid being detected, using a slow attack.

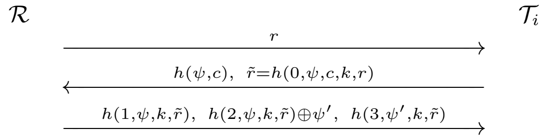

$\mathcal{R}$
$\mathcal{T}_i$
r
$h(\psi,c),\ \tilde{r}=h(0,\psi,c,k,r)$
$h(1,\psi,k,\tilde{r}),\ h(2,\psi,k,\tilde{r})\oplus\psi',\ h(3,\psi',k,\tilde{r})$

Figure 3.9: The CTI Protocol.

3.3.4 Discussion

While protocols using shared secrets all aim mainly to decrease the identification time on the reader, they all have issues when facing adversaries capable of compromising tags. One could argue that a protocol using only one "master key" is the extreme case in that direction: it has constant-time identification, but no privacy/security as soon as one tag is compromised.

All of the proposals analyzed in this section have important problems, mostly due to the fact that compromising one tag reveals information on other tags too. However, no element shows that sharing secrets between tags is a definitely flawed way of reducing identification time. It remains an open question whether it is possible to design such a protocol without any loss of security or privacy.

3.4 Protocols Based on Hash-Chains

An early family of sub-linear protocols uses hash-chains to update the internal state of the tags. In this section, the protocol of Ohkubo, Suzuki, and Kinoshita's (OSK) [149] is described along with two of its improvements, OSK/AO [25, 30] and OSK/BF [145]. Another protocol based on hash-chains, O-RAP [58], is then discussed.

A traceability attack is pointed out on the mutual authentication extension of OSK/AO protocol, and a solution to overcome this problem is suggested. New weaknesses of O-RAP and OSK/BF are also given.

3.4.1 OSK Protocol

OSK [149] is a well-known synchronized identification protocol[5], and was one of the earliest of its kind. However, beside its traceability issue, and although the protocol is very efficient when all tags are synchronized, the worst-case complexity of the search makes the protocol unsuitable for most practical systems.

OSK works as follows. Each tag $\mathcal{T}_i$ of the system is initialized with a randomly chosen secret s_i^0. When queried by a reader, a tag answers with the hash of its *current* secret, that is $\sigma = G(s_i^j)$, and immediately updates it using another hash function: $s_i^{j+1} = H(s_i^j)$. When receiving an answer, the reader looks in its database for an initial secret s_i^0 that leads to σ, in other words, it checks whether there exists i and j such that $G(H^j(s_i^0)) = \sigma$. To do that, from each of the N initial secrets s_i^0, the reader computes the hash chains as shown in Figure 4.2 until it finds a value matching σ, or until it reaches a given maximum limit L on the chain length. An overview of the protocol is shown in Figure 3.11.

The value $\sigma = G(s_i^j)$ does not allow an eavesdropper to learn the identity of $\mathcal{T}_i$. However, since a tag updates its secret regardless of the success of the identification, a rogue reader initiating the protocol with $\mathcal{T}_i$ will make it update its secret. An adversary initiating lots of instances of the protocol with $\mathcal{T}_i$ will perform a *desynchronization* denial-of-service attack. Indeed, the reader would then need to compute a lot of hashes to identify $\mathcal{T}_i$. To prevent this, the length of the hash chains have to be bounded, i.e., the reader stops its search after L hashes per tag. This protection has the following drawback: an adversary skimming a tag L times makes it unable to be identified by the system, and therefore traceable (see the model of privacy of Juels and Weis [113] for details).

Another isue is the timing attacks [22]. Timing attacks are side-channel privacy attacks, and their objective is to distinguish two tags based on the amount of time taken during authentication. In the case of OSK, it is possible for an adversary to distinguish two tags if one takes significantly longer to be authenticated than the other one (for instance if the former has previously been skimmed many times by the adversary).

The authors later introduced in [150] some ideas to improve the efficiency of the search at the cost of lowering privacy. They are not considered here since these modifications purposely reduce the privacy of the protocol. Instead, variants that make the search more efficient without hurting the privacy are presented thereafter.

[5]PFP, introduced by Berbain, Billet, Etrog, and Gilbert in [41] is strongly inspired by OSK. The building blocks in PFP are different than the ones in OSK, and they are used in a different way, but the global scheme is the same, and the security and privacy properties of the two protocols are equivalent. Hence, PFP is not detailed further.

$$\begin{array}{cccccccc}
s_1^0 & \longrightarrow & r_1^0 & r_1^1 & r_1^2 & \dots & r_1^{L-1} & r_1^L \\
\dots & \longrightarrow & \dots & \dots & \dots & \dots & \dots & \dots \\
s_i^0 & \longrightarrow & \dots & \dots & \dots & \boxed{r_i^j = G(H^j(s_i^0))} & \dots & r_i^L \\
\dots & \longrightarrow & \dots & \dots & \dots & \dots & \dots & \dots \\
s_N^0 & \longrightarrow & r_N^0 & r_N^1 & r_N^2 & \dots & r_N^{L-1} & r_N^L
\end{array}$$

Figure 3.10: Chains of hashes in the OSK protocol.

$$\begin{array}{ccc}
\mathcal{R} & & \mathcal{T}_i \\
 & \xrightarrow{\hspace{2em} request \hspace{2em}} & \\
\begin{array}{c}\text{find } i \text{ and } j \text{ so that} \\ G(H^j(s_i^0)) = \sigma\end{array} & \xleftarrow{\hspace{2em} \sigma = G(s_i^j) \hspace{2em}} & s_i^{j+1} \leftarrow H(s_i^j)
\end{array}$$

Figure 3.11: The OSK protocol.

3.4.2 OSK/AO Protocol

Avoine and Oechslin propose in [30] to apply Hellman's time-memory trade-offs [96] to the search procedure of OSK, which has two main implications. First, the complexity of the search procedure varies from $O(1)$ to $O(NL)$, depending on the amount of memory one is willing to devote to the time-memory trade-off[6]. Moreover, the search is intrinsically randomized, which prevents timing attacks [22].

Avoine, Dysli, and Oechslin also suggest in [25] a variant of OSK that ensures authentication as OSK is originally designed to provide private identification only (i.e., it does not resist to replay attacks). To do so, they suggest using nonces: instead of simply sending a *request* message, the reader sends a nonce r, and the tag answers $G(s_i^j \oplus r)$ along with $G(s_i^j)$.

Finally, Avoine proposes in [10] an extended version of OSK that provides reader authentication to the tag: the reader sends a last message $G(s_i^{j+1} \oplus w)$, where w is a public static value.

However, a traceability issue exists in this extension: an adversary can eavesdrop a legitimate authentication between $\mathcal{R}$ and $\mathcal{T}_i$, and record the last message (i.e. $G(s_i^{j+1} \oplus w)$); after a while, she sends w as a nonce to a tag, and if the tag answers with the previously recorded value, this tag is almost certainly $\mathcal{T}_i$, and it has not been queried since then.

Preventing this attack can be done easily using a third hash function for the last message. In practice, a single hash function is implemented and an additional input enables to derive it into several functions, for instance, by concatenating 0, 1, or 2 to the value to hash. Figure 3.12 shows our modification to the mutual authentication extension of OSK/AO.

[6]The authors mention that, for instance, a complexity of $O((NL)^{2/3})$ can be reached with a memory of size $O((NL)^{2/3})$.

$$\begin{array}{ccc}
\mathcal{R} & & \mathcal{T}_i \\
 & \xrightarrow{\hspace{3em} r \hspace{3em}} & \\
\text{search for } i & \xleftarrow{\hspace{1em} H_1(s_i^j \oplus r),\ H_1(s_i^j) \hspace{1em}} & s_i^{j+1} \leftarrow H_0(s_i^j) \\
s_i^{j+1} \leftarrow H_0(s_i^j) & \xrightarrow{\hspace{2em} H_2(s_i^{j+1}) \hspace{2em}} &
\end{array}$$

Figure 3.12: Patched OSK with replay-attack protection and reader authentication.

3.4.3 OSK/BF Protocol

Description

Nohara, Inoue and Yasuura propose in [145] another innovative time-memory trade-off for OSK, which is labeled OSK/BF in the following. They use Bloom Filters [49], a space-efficient data structure, to store all the hash-chains of each tag. When identifying a tag, the reader first queries all the Bloom Filters for the received σ, and then computes the whole hash-chain of each candidate to confirm the identity of the tag. Once identified, the corresponding Bloom Filter is re-computed for the next hash-chain. On that point OSK/BF contrasts with OSK/AO, in which updates of the database occur less frequently but are more costly.

As presented in [145], OSK/BF is an identification protocol and does not resist impersonation. However, it could be easily adapted to an authentication scheme using the same construction as the one in [25].

In [144], Nohara and Inoue present an analogous protocol using a similar architecture but a different data structure, d-left Hash Tables [56], an extension of Bloom Filters. The resulting protocol has, according to the authors, a better update efficiency than OSK/BF, but it turns out to be the same. Furthermore, the identification time seems to be very comparable to that of [145], and it has the further disadvantage of being less parameterizable.

Traceability Timing Attacks

Discussed below are two potential traceability weaknesses of OSK/BF due to timing analysis, not mentioned in [145]. The first one uses the fact that the search is linear in [145], meaning that $\mathcal{T}_1$ will on average be authenticated much faster than $\mathcal{T}_N$, for instance. The reason is that when a tag has a record (and a corresponding Bloom Filter) at the start of the table, the reader has to go through few false positives invalidations before actually confirming the identity of the tag, whereas when it has a record near the end of the table, it might go through several of them. The second attack uses the fact that it is possible to trace a tag being desynchronized more than L times by observing whether the identification time remains constant (it should be constant when the reader refuses identification, but not when the Bloom Filters get updated). Countermeasures might exist against these attacks (simply shuffling the search seems to be a solution to the first one), but in any case, OSK/BF is more fragile regarding timing analysis than OSK/AO, and avoiding them without artificially waiting for $O(N)$ cryptographic operations does not seem to be trivial.

Comparison with OSK/AO

As in OSK/AO, the time of identification can be lowered by increasing the memory of the reader. In OSK/BF, this is done by tuning the false positive rate of the Bloom Filters. Doing so results in more time needed to compute the hash-chains in order to infirm false positives, increasing identification time, but also in a decrease of the size of Bloom Filters and thus of memory. In OSK/AO, this is done by tuning the size of the Rainbow table, and also determining the amount of intermediate columns stored.

A slight advantage of OSK/BF over OSK/AO is that, despite it also has a probabilistic nature, the successful identification rate is of 100% while being *close to* 100% (fixed by parameters) in OSK/AO. However, the two protocols have the same disadvantage regarding desynchronization, i.e., a tag desynchronized more than L times is lost.

Regarding the trade-off efficiency, OSK/AO seems slightly more efficient than OSK/BF, although comparable. Numbers from [25] were used, i.e. a system of 2^{20} tags and chains of 2^7 hashes, to provide a comparison between the two protocols, which is depicted in Figure 3.13. The saturation in OSK/BF after some point comes from the fact that the update part takes $2L$ cryptographic operations, no matter how much memory is dedicated to the trade-off. Note also that the random hash calculations is not taken into account. This could, depending on the functions used, increase the identification time significantly.

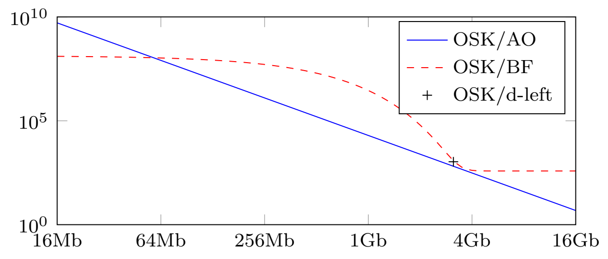


Figure 3.13: Average number of cryptographic hashes during identification for variants of OSK.

3.4.4 O-RAP Protocol

Description

O-RAP, which stands for *Optimistic RFID Authentication Protocol*, has been originally introduced in [57] by Burmester, van Le and de Medeiros. Its former name was O-TRAP (see Table 3.1) and a slightly modified version is re-presented in [58]. The authors call the protocol "optimistic" for the reason that the security overhead is minimal when the system is not under attack. The steps of O-RAP are shown in Figure 3.14. The reader contains a hash table indexed by r_{tag} with entries K_i (the static keys of the tags). When starting an authentication, the reader sends a random number r_{sys} to the tag. The tag computes the hash of r_{sys} and r_{tag} with its key K_i and gets r and h output values. Then the tag sends h and r_{tag} values to the reader. The tag also updates r_{tag} with r value. The system searches r_{tag} to find the corresponding K_i in the database, and if found, it checks the correctness of the hash. If r_{tag} is not found, then it exhaustively searches among all the keys. If found, it validates the tag and updates r_{tag} with r value. This allows the reader to re-synchronize the tag automatically.

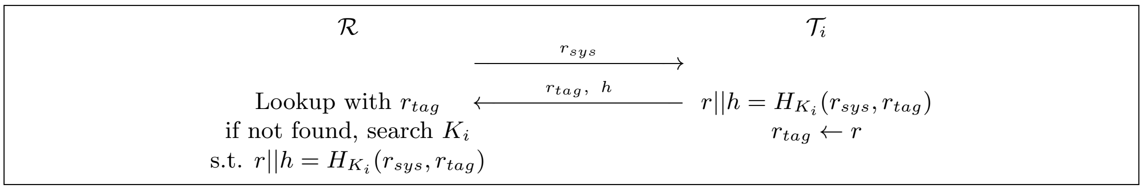

$\mathcal{R}$
$\mathcal{T}_i$
r_{sys}
$r_{tag},\ h$
Lookup with r_{tag}
if not found, search K_i
s.t. $r||h = H_{K_i}(r_{sys}, r_{tag})$
$r||h = H_{K_i}(r_{sys}, r_{tag})$
$r_{tag} \leftarrow r$

Figure 3.14: O-RAP Protocol.

Attack by Ouafi and Phan

In [151], Ouafi and Phan propose a traceability attack on O-RAP based on the desynchronization of a tag. The idea is that an adversary can make enough queries to a tag in order to make it update its secret r_{tag} a lot of times to the point that a legitimate reader is unable to authenticate it anymore.

However, this attack seems erroneous. Indeed, the tag always sends r_{tag} in its answer so the resynchronization is trivial, and because K_i does not change, the authentication is always correct, regardless of how many queries the attacker has performed.

Forward-Privacy Issue and O-FRAP

Although the authors raise the problem in [57], no particular attention has been drawn on the forward-privacy of O-RAP. An attacker compromising $\mathcal{T}_i$ at some point can recover r_{tag} and K_i. This allows him to trace $\mathcal{T}_i$ in the past, because r_{tag} is sent in the clear and is updated by r. This update can be computed by the adversary, since K_i does not change.

The authors propose in [182] the O-FRAP protocol, adding the forward-privacy to O-RAP. This comes at the cost of an extra pass in order to authenticate the reader to the tag, as well as a memory overhead for storing previous keys. However, the protocol is not forward-private strictly speaking. Indeed, suppose than an adversary queries the tag some times without answering to it. Afterwards, she compromises the tag, and if the tag has not been authenticated since, she will be able to trace it in the past. This is the same idea as protocols using pseudonyms for identification, described in Section 3.2.

Also note that in [58] and [182], the authors propose key exchange extensions to O-RAP and O-FRAP respectively, namely O-RAKE and O-FRAKE. Their goal is to provide features outside of authentication, and are not analyzed further.

Traceability Timing Attack

The fact that O-RAP behaves differently according to synchronization makes it work very efficiently in "normal" situations, but allows an adversary to carry out the following timing attack. The adversary first sends a random number to a tag and ignores its answer. The tag will thus be desynchronized with the system, and the next legitimate reader trying to authenticate it will take much more time, because in that case, the search is linear. The adversary can easily notice that by measuring time differences, and can thus trace the tag she desynchronized.

A possible countermeasure is to artificially add time for the search in a normal situation, but this would be equivalent to a protocol with linear complexity.

3.4.5 Discussion

OSK and O-RAP are two convincing proposals with a simple design and interesting properties.

As pointed by Avoine and Oechslin in [30] and by Nohara *et al.* in [145], OSK can be easily accommodated to using time-memory trade-offs, which make the identification procedure efficient. It also provides forward-privacy to the tags. However, the synchronization issue present in OSK and its variants, although mitigable, remains significant.

In that regard, the O-RAP protocol has no such synchronization issue because tags automatically "re-synchronize" with each authentication attempt. It is also the reason why the identification procedure is constant-time in normal situations. However, it is very easy to make the next search linear by querying the tag once. This also leads to traceability issues using reader-side timing analysis. Additionally, it provides no forward-privacy.

Despite their respective weaknesses, these protocols are nonetheless probably the most solid solutions analyzed here that are applicable to RFID tags (with symmetric key capabilities).

3.5 Counter-Based Protocols

The *counter-based protocols* all share the same characteristics: they use a strictly increasing number[7] and maintain a periodically updated hash table for each counter. The idea is to pre-compute the table at each counter tick, in order to reduce the online search to a constant time on the server-side.

This section examines a family of counter-based protocols, namely RIP, RIP+, RAP, and YA-TRAP* (see Table 3.1 for the names given in different papers). A traceability attack is given on the most advanced protocol proposed in [179], namely YA-TRAP*, based on timing analysis.

3.5.1 YA-TRAP Family

A family of tag identification and authentication protocols that use strictly increasing counters is proposed in the papers [57, 58, 178, 179]. The first protocol, RIP, stands for *RFID Identification Protocol*. It is followed by authentication protocols called RIP+, YA-TRAP*, and a variant of YA-TRAP* with forward-privacy (called here YA-TRAP*&fwd).

[7]In some previous papers [57, 58, 178, 179] the name "*timestamp*" is used to denote a strictly increasing number. Since the tags do not have any clock and this number is not a cryptographic timestamp, the more generic term *counter* is preferred here.

Description

The RIP [58] protocol, which is the simplest and earliest proposal in the family, is described below.

Each tag $\mathcal{T}_i$ is initialized with a starting counter T_0 and a maximum counter value $T_{\max}$, as well as with a unique secret key K_i. When initiating an authentication, the reader sends its current counter T_r. The tag checks that T_r is less than $T_{\max}$ and that the received counter is bigger than the one it currently stores, T_t, which it received during the last successful identification. If these conditions hold, it stores the new counter and computes and sends the hash of T_r with its key K_i. Otherwise, the tag sends a random number to prevent an adversary from drawing any conclusion. The authors added that to avoid timing attacks against a tag at this point, the nonce generation must be designed to take approximately the same time as the hash computation.

As stated above, every now and then, the server increases the value of the counter, and re-computes the table accordingly. This allows for a constant time identification online, but takes time offline.

The authors identified several drawbacks in this protocol. First, it is vulnerable to a trivial DoS attack: the adversary can temporarily or permanently incapacitate a tag by sending a future counter. Although the authors point out that DoS resistance is not the main goal of this protocol, the attack is very easy to perform and very hard to recover from. Second, it is implicitly assumed that a tag is never identified more than once between two consecutive counter ticks. A short time interval (e.g., a second) between two counter updates makes this assumption realistic, but it causes heavy computational burden for the server. RIP is also vulnerable to replay attacks: an adversary can send a counter slightly ahead to a tag and wait until this counter is sent by the server. She can repeat this attack and thus impersonate its victim for a long time without the original tag being present. RIP is depicted in Figure 3.15.

$$\begin{array}{ccc} \mathcal{R} & & \mathcal{T}_i \\ & \xrightarrow{\quad T_r \quad} & \text{if } (T_r \leq T_t) \text{ or } (T_r > T_{\max}), \\ & & \text{then } h_{id} \text{ random} \\ & & \text{else } h_{id} = H_{K_i}(T_r) \text{ and } T_t \leftarrow T_r \\ \text{Lookup } h_{id} & \xleftarrow{\quad h_{id} \quad} & \end{array}$$

Figure 3.15: RIP protocol.

In RIP+, the protocol is modified in order to provide authentication. The reader sends a random nonce R_r along with the counter. The tag chooses its random nonce R_t and computes a hash for authentication $h_{auth} = H_{K_i}(R_t, R_r)$. The reader first identifies the tag, then checks the correctness of the hash.

Note that although this prevents the replay attack, the two aforementioned issues are still present.

$$\begin{array}{ccc} \mathcal{R} & & \mathcal{T}_i \\ & \xrightarrow{T_r,\ R_r,\ ET_r} & \nu = \lfloor T_r/INT \rfloor - \lfloor T_t/INT \rfloor \\ & & \text{if } (T_r \leq T_t) \text{ or } (T_r > T_{\max}) \\ & & \text{or } H^{\nu}(ET_r) \neq ET_t, \\ & & \text{then } h_{id} \text{ random and } h_{auth} \text{ random} \\ & & \text{else } T_t \leftarrow T_r,\ ET_t \leftarrow ET_r,\ h_{id} = H_{K_i}(T_t), \\ & & h_{auth} = H_{K_i}(R_t, R_r) \\ \text{Lookup } h_{id} & \xleftarrow{h_{id},\ R_t,\ h_{auth}} & \\ \text{check } h_{auth} \overset{?}{=} H_{K_i}(R_t, R_r) & & \end{array}$$

Figure 3.16: YA-TRAP* protocol.

In order to cope with DoS attacks, Tsudik proposed YA-TRAP*, which is illustrated in Figure 3.16. DoS resistance is achieved by using a system-wide hash-chain. At setup, the system initializes a long Lamport-chain [119] of hashes, and sets the value ET_t of all tags to the last hash computed. Every INT counter ticks, a value of the hash-chain is popped, and the next one is used as ET_r. During an authentication session, a tag receiving T_r, R_r and ET_r will compute the number of intervals skipped since the last authentication (i.e. $\nu = \lfloor T_r/INT \rfloor - \lfloor T_t/INT \rfloor$), and will verify that the hash ET_r is the corresponding predecessor of ET_t by checking whether $H^\nu(ET_r) = ET_t$[8].

Note that DoS resistance in YA-TRAP* is limited by the magnitude of INT value. When ET_r is sent by the system it is no longer secret. Therefore, the adversary can still incapacitate tags up to the duration of INT by querying the tag with the maximum possible T_r value within the current epoch.

All the aforementioned protocols do not provide forward-privacy because the long-term key of the tags are static. Tsudik introduces an additional operation for updating the keys of the tags. In this extension, which is called YA-TRAP*&fwd hereafter, a tag takes ν times hash of the key for each authentication namely $K_i^\nu = H^\nu(K_i)$. With this modification, the tag's key is changed once per INT interval, and this brings ν additional hash operations on the tag-side.

Attacks on YA-TRAP*

In YA-TRAP*, the tag computes ν times the hash function depending on the difference between the T_t and T_r values. If the received T_r value is within the same interval as T_t, the tag computes no hash function for the interval check. If the difference between these two counters is large, the tag has to compute many hash functions. This leads to two potential attacks.

The first one is a traceability attack. It is simply that if a tag has not been authenticated in a long time, it is traceable due to the amount of time it spends computing the hashes. Distinction is thus possible between two tags in some situations.

The second one is a DoS. If an adversary sends a big T_r and a random ET_r to a tag, the latter needs to compute many hashes, even if it will eventually discard the request since the ET_r is not correct. Depending on INT, this can make the authentication impossible due to the amount of time needed by the tag to complete its calculation.

The parameter INT must be carefully chosen: the bigger, the less it mitigates the DoS already present in RIP+; and the smaller, the more computation on tags, leading to the two problems described above.

Other Protocols

Another counter-based protocol called YA-TRAP+ is proposed by Burmester *et al.* in 2006 [57, 62]. A slightly modified version of it is presented in [58] with a new name "RAP." This protocol is very similar to O-RAP in terms of security properties. In [58] it is also stated that "O-RAP is simpler than RAP, at the cost of not supporting kill-keys. The security for O-RAP is similar to that of RAP." In particular, the two issues mentioned in Section 3.4.4 are also applicable to RAP. Additionally, in O-RAP a desynchronized tag is resynchronized automatically after each legitimate authentication, however RAP does not support automatic resynchronization. For these reasons, only O-RAP is analyzed among those two similar protocols.

3.5.2 Discussion

Counter-based protocols, embodied by the YA-TRAP family, provide an interesting approach to constant-time identification. However, since the counter must be provided in the clear and, as such, is not authenticated, DoS attacks are extremely easy to accomplish and hard to prevent. YA-TRAP* attempts to alleviate this problem but at the same time introduces other weaknesses as indicated in Section 3.5.1.

3.6 Comparison

In this section, the most of the protocols analyzed in this chapter are analyzed and compared on several criteria, as shown in Table 3.2. Evaluated here are the schemes that provide sub-linear

[8]Note that in [179], the authors mistakenly stated this check was $H^\nu(ET_t) = ET_r$.

Table 3.2: Comparison of the protocols analyzed in this chapter. Letters in brackets link to comments described below.

Class	**Shared secrets**			**Hash-chains**					**Counter-based**		
Protocol	CHT plain	CHT with auth	CTI	OSK	OSK/AO with Auth	OSK/BF with Auth	O-RAP	O-FRAP	RIP+	YA-TRAP*	YA-TRAP* &fwd
Main reference	[63]	[63]	[4]	[149]	[10, 25, 30]	[145]	[57]	[182]	[179]	[179]	[179]
Year of publication	2009	2009	2010	2003	2005	2008	2006	2007	2007	2007	2007
Identification/ Authentication	auth.[A]	auth.	auth.	id.	auth.	auth.	auth.	auth.	auth.	auth.	auth.
Off-line Computation Complexity[B]	0	0	$O(NC)$[C]	$2N$[C]	$\frac{NL^2}{2}$ ([25])[D]	$2NL$[C]	0	0	N/counter update[E]	N/counter update[E]+ Lamport Chain	N/counter and key update[E] + Lamport Chain
Normal case online complexity	$O(\sqrt{N})$	$O(N^a)$	4	2	$O((NL)^{2/3})$[F]	$L(\epsilon N+3)$ on average	1	2	0[E]+1[G]	0[E]+1[G]	0[E]+1[G]
Desynchronized case online complexity	N/A	N/A	N/A	lower than $2N(L-1)$[H]	$O((NL)^{2/3})$[F]	$L(\epsilon N+3)$ on average	$O(N)$	$O(N)$	out of order after desync.[H]	out of order after desync.[I]	out of order after desync.[I]
Memory Complexity	$2\sqrt{N}$	$2N^a+N$	$O(N)$[J]	N	$O((NL)^{2/3})$[F]	$\frac{NL\log\epsilon}{-\log^2 2}$	$2N$	$3N$	N[E]	N[E]	N[E]
Tag Computation	2 PRFs + 1 Nonce	3 PRFs + 1 Nonce	5 hashes	2 hashes	3 hashes	3 hashes	2 hashes	4 hashes	2 hashes + 1 Nonce	$\nu+2$ hashes + 1 Nonce	2ν + 2 hashes + 1 Nonce
Tag Resources	PRF, PRNG	PRF, PRNG	PRNG, Hash func.	Hash func.	Hash func.	Hash func.	Hash func.	Hash func.	PRNG, Hash func.	PRNG, Hash func.	PRNG, Hash func.
Privacy	no	no★	no★	yes[K]	yes◊,[K]	no★	no★,[K]	no★,[K]	not private after desync.	not private after desync.★,[L]	not private after desync.★,[L]
Forward-privacy	no	no	no[M]	yes[K]	yes[K]	no[M]	no	no★	no	no	no[N]
Desynchronization resistance	N/A	N/A	yes	yes up to L consecutive[O]	yes up to L consecutive[P]	yes up to L consecutive[O]	no[Q]	no[Q]	no[R]	yes[S]	yes[S]
Impersonation Resistance	no★	yes	yes	N/A	yes	yes	yes	yes	yes	yes	yes

★ : Weaknesses discovered in this thesis. ◊ : Weaknesses discovered and fixed in this thesis.

[A] Although the authors implicitly consider it to be an identification protocol (because of the existence of an *authentication* extension), it is referred to as an authentication protocol here since the reader sends a nonce, and since the protocol is at first designed to cope with impersonation.

[B] Excluding key generation.

[C] Done during the setup.

[D] Done each time one tag reaches L authentications since the last table update.

[E] The whole hash table is periodically updated or can be precomputed for forecoming counters.

[F] Using rainbow tables. The complexity provided is an example, but the identification complexity can be set anywhere between $O(1)$ and $O(NL)$ according to the memory available for the trade-off (see Section 3.4.2 for discussion).

[G] Additional hash for authentication.

[H] The desynchronized tag might not be identified/authenticated by the reader.

[I] The tag cannot be authenticated within the time interval. It gets resynchronized in the next one.

[J] Can be big due to constant terms (see Section 3.3.3 and [4]).

[K] Private if and only if the adversary is not able to tell whether the protocol session was successful.

[L] However, private after resynchronization. The tag resynchronize automatically at each interval start.

[M] Since it is not private.

[N] Not forward private until the last ET update.

[O] After L desynchronizations, the tag owner can go to some central office to fix the issue.

[P] Including legitimate authentications. If the tag owner goes to the office the tag can be resynchronized in the next precomputation. However, if the number of the illegitimate authentications is less than L, then the resynchronization of the tag will be done automatically during the next update.

[Q] Tags can be desynchronized, but resynchronize automatically after each authentication.

[R] Either up to $T_{\max}$ (tag becomes useless) or with a smaller counter (for to traceability and replay).

[S] Tags can be easily desynchronized within a time interval (e.g. one day), making them unusable and/or traceable. Tags are automatically resynchronized at the beginning of each interval. Tags have limited lifetime because the Lamport chain must have a start (although the system can be initialized with a really big hash-chain).

complexity, at least during the normal case online interaction, and that intend to provide at least user privacy (not necessarily forward-privacy). Those for which new weaknesses are highlighted in this chapter are also included. For clarity reasons, additional remarks are provided (superscripted capital letters in the table) below the table. It is difficult to compare these protocols objectively because of the number of criteria available. Nonetheless, some hindsights are given below about their comparison.

A major design goal for authentication protocols is the protection against impersonation attacks. Cheon, Hong, and Tsudik's plain protocol can therefore be discarded since it does not satisfy this requirement. OSK and its variants do not satisfy it either, but they are identification protocols, and it is possible to extend them to authentication protocols as explained in Section 3.4.2. For the rest of the protocols, a trade-off between efficiency of the reader authentication complexity, and privacy weaknesses and/or other issues is generally observed.

The protocols using shared secrets, although presenting alluring identification efficiency, have important security and privacy problems, as stated earlier. They are particularly vulnerable in scenarios where tag corruption is easy. Nonetheless, future ideas might lower the impact of tag compromise, and this approach remains interesting.

The counter-based protocols, embodied by the YA-TRAP family, seem to be promising as well, but are easily desynchronized, which decreases the privacy they provide. Their usability (a maximum of one authentication per counter tick) might be a problem in some applications too. If this is not a problem, YA-TRAP* provides a decent level of privacy and allows automatic re-synchronization.

Finally, although not ideal, the protocols based on hash-chains seem to be the most solid solutions to date among the protocols analyzed in this chapter. OSK/AO and OSK/BF provide forward-privacy and a very good efficiency for the authentication on the reader side, but have desynchronization issues due to the finite size of the chains. O-RAP is also quite good and does not have desynchronization problems. However, if an adversary capable of performing timing analysis is considered, it has a lower privacy. O-FRAP also brings some forward-privacy to O-RAP (but not completely as pointed out in Section 3.4.4).

Some protocols require fewer assets on the tag. For instance, hash-chains protocols do not require randomness to originate from tags and might therefore be more easily implemented on low-cost tags.

In conclusion, the choice of the best protocol depends on the scenario, and on the privacy and efficiency requirements. In any case, the protocols based on hash-chains clearly stand out.

3.7 Conclusion

This chapter studied the problematic of the scalability of private symmetric-key authentication protocols in large systems. In order to ensure the privacy of a tag's identity in such protocols means that no information sent over the channel should help an attacker to identify a tag. A the same time, the reader needs some information to identify the tag quickly. These two contradictory requirements (along with other constraints) have been addressed by a number of proposals.

These protocols have been analysed in this chapter, and a number of attacks and weaknesses have been identified. Two new attacks were described on the CHT protocol [63], a very efficient protocol in terms of key search complexity (i.e., $O(\sqrt{N})$). Two new traceability attacks were also introduced on the CTI protocol [4]. Furthermore, a traceability weakness was found on the mutual authentication version of OSK/AO [25] protocol, and a possible way to repair this problem was proposed, with no additional cost. Finally, traceability attacks were introduced on OSK/BF [145], O-RAP [62] and YA-TRAP* [179], emphasizing the importance of timing attacks [22] on the reader side.

All candidates have been extensively evaluated and comapred according to their security and performance. The conclusion is that in the current state of the art, protocols based on hash-chains, such as OSK/AO and O-RAP are those that have the best characteristics overall. The next chapter takes a closer look at OSK/AO in practical settings.

Chapter 4

The OSK/AO Protocol

Articles related to this chapter

[12] Gildas Avoine, Muhammed Ali Bingol, Xavier Carpent, and Suleyman Kardas. Deploying OSK on low-resource mobile devices. In *Workshop on RFID Security – RFIDSec'13*, Graz, Austria, July 2013.

This chapter takes a closer look at OSK/AO, an extension to the OSK protocol that improves significantly its efficiency without lowering its security. It is one of the most promising solutions to the scalability of privacy-friendly symmetric-key authentication protocols discussed in Chapter 3.

This protocols makes use of time-memory trade-offs, a technique to carry out efficient brute-force search. Time-memory trade-offs are typically used to perform cryptanalysis such as password cracking, but they are used in a constructive way in OSK/AO. An important leverage of time-memory trade-offs is the "lunchtime attack" scenario (see Chapter 1), where the user may benefit from a long preparation time but has a limited window of opportunity and limited resources to perform the actual search. This aspect of time-memory trade-offs is used in OSK/AO to provide efficient identification. Reader unfamiliar with time-memory trade-offs are referred to Section 5.1 for a detailed description.

As part of the work on OSK/AO, the protocol was also implemented on an NFC-compliant[1] cellphone and a ZC7.5 contactless tag. This is, as far as we know, the first implementation of a protocol of this type. This implementation demonstrates the practicability and efficiency of the OSK protocol and illustrates that privacy-by-design is achievable to some extent in constrained environments.

The OSK protocol is described in Section 4.1, along with its online search and full storage approaches. Section 4.2 presents the OSK/AO protocol in details. Section 4.3 presents our implementation, and the chapter is concluded in Section 4.4

4.1 Ohkubo, Suzuki, and Kinoshita's Protocol

4.1.1 Description

The OSK protocol is proposed by Ohkubo, Suzuki, and Kinoshita in [149]. It is one of the most well-known RFID-devoted authentication protocols and is the earliest one that achieves *forward privacy*[2]. In the RFID context, forward privacy is the property that guarantees the security of past interactions of a tag even if it is compromised at a later stage. Namely, if the secret information of

[1]The NFC (Near Field Communication) technology is a set of standards built upon RFID standards that allow two-way communication on smartphones and other mobile devices.

[2]It is also known as *backward untraceability* and used interchangeably in some papers [115, 132, 162].

$$
\begin{array}{ccc}
\mathcal{R} & & \mathcal{T}_i \\
 & \xrightarrow{\quad request \quad} & \\
\text{find } i \text{ and } j \text{ so that} & \xleftarrow{\quad \sigma=G(s_i^j) \quad} & s_i^{j+1} \leftarrow H(s_i^j) \\
G(H^j(s_i^0)) = \sigma & &
\end{array}
$$

Figure 4.1: The OSK protocol.

$$
\begin{array}{cccccccc}
s_1^0 & \longrightarrow & r_1^0 & r_1^1 & r_1^2 & \dots & r_1^{L-1} & r_1^L \\
\dots & \longrightarrow & \dots & \dots & \dots & \dots & \dots & \dots \\
s_i^0 & \longrightarrow & \dots & \dots & \dots & \boxed{r_i^j = G(H^j(s_i^0))} & \dots & r_i^L \\
\dots & \longrightarrow & \dots & \dots & \dots & \dots & \dots & \dots \\
s_N^0 & \longrightarrow & r_N^0 & r_N^1 & r_N^2 & \dots & r_N^{L-1} & r_N^L
\end{array}
$$

Figure 4.2: Chains of hashes in the OSK protocol.

a tag $\mathcal{T}_i$ ($1 \leq i \leq N$) is corrupted by an adversary at a given time, the adversary can not associate any transaction with $\mathcal{T}_i$ at any earlier time (See Chapter 1 for more details).

The OSK protocol is described in Section 3.4.1 and may be summarized as follows. Each tag $\mathcal{T}_i$ has an initial secret s_i^0 that is updated after each authentication query. The update consists in hashing the current secret with a one-way function H. Upon reception of the authentication query, the tag answers by hashing the current secret with a different hash function[3], G. Fig. 4.1 shows the OSK protocol.

System Setup.

Each tag $\mathcal{T}_i$ of the system is initialized with a randomly chosen secret s_i^0. The N initial secrets are stored in a database, sometimes called back-end system. In some settings the back-end system and the reader are two different devices, connected in a way that is considered secure. In some other settings the back-end system is embedded in the readers.

Interrogation.

When the tag is queried by a reader it answers with a response using the current secret such that $\sigma = G(s_i^j)$ and also updates the secret immediately using a different hash function: $s_i^{j+1} = H(s_i^j)$.

Search & Identification.

When receiving an answer the database searches for an initial secret s_i^0 that leads to σ. In other words, it checks whether there exists i and j such that $G(H^j(s_i^0)) = \sigma$. To do that, from each of the N initial secrets s_i^0, the reader computes the hash chains as shown in Fig. 4.2 until it finds a value matching σ, or until it reaches a given maximum limit L (the "lifetime" of a tag[4]) on the chain length.

The value $\sigma = G(S_i^j)$ does not leak any information to an attacker on the secret of $\mathcal{T}_i$ when G and H behave as pseudo-random functions. However, given that the authentication process is bounded by L, the OSK protocol is prone to desynchronization when an adversary queries the tag more than L times. In such a case, the tag can no longer be authenticated and a privacy issue arises for a certain type of adversary, capable of detecting the success status of an authentication (see [113] for a discussion). Fortunately, the synchronization can be retrieved by the back-end

[3]Note that although these two functions need to be different, only one algorithm may be implemented on the tag, and an additional 1-bit input parameter used to select the function.

[4]The lifetime of a tag corresponds to the maximum number of times it may be interrogated by an adversary without being desynchronized with the database. On genuine authentications, the tag is resynchronized with the database, and its lifetime reset.

without replacing the tag; for example, the holder of a *desynchronized* tag can ask the system operator to recompute the chain of his tag.

Beside this desynchronization issue – and although the protocol is very efficient when all the tags are synchronized [149] – the worst-case complexity of the search procedure makes the protocol unsuitable for most practical applications.

4.1.2 Real-life Applications

This section discusses the possibility of implementing OSK in real-life applications. A system of $N = 2^{20}$ tags with a lifetime of $L = 2^7$ is chosen to represent a typical application throughout this chapter. These are reasonable parameters and are in accordance with [25].

Online Search

A naïve approach for the server is to only keep the initial secrets and recompute the $N \times L$ possibilities each time it receives a given σ. With a server capable of 2^{20} cryptographic hash operations per second, this takes 2^6 seconds $\approx$ 1 minute on average for these parameters. This is far beyond the limit of 200 milliseconds for a reasonable authentication time.

Full Storage

The other extreme solution consists in storing all the chains in a table and letting the server perform a simple look-up whenever it receives σ. This solution has the advantage of requiring no cryptographic operation on the reader side during the authentication, which makes the authentication very fast. Unfortunately, this approach has two major drawbacks.

First of all, a large memory is needed to store the table: given our parameters ($N = 2^{20}$ tags with a lifetime of $L = 2^7$, and a hash size of 128 bits), the full storage approach requires 2^{34} bits = 2 GB[5]. In a system where readers are permanently connected to the back-end server, requiring such a memory (RAM) for the server is not a problem. However, in systems consisting of mobile readers sporadically connected to the database, the authentication material should be replicated in each of these low-resource devices. In such a scenario, this amount of memory is very large for small devices such as PDA's or handheld RFID readers, which typically have a memory of 128 MB. It might however be reasonable for more elaborate devices such as NFC-enabled smartphones, which have several gigabytes of flash memory.

A second issue is that, every now and then, the table needs to be either computed in a central server and uploaded on the smartphones, or computed by the smartphones themselves after reception of the first column of the table. This might take a significant time for both cases, and might be an issue in certain situations.

In the context of [45] for instance, where a central server is used, the full storage technique makes sense, and is more simple and efficient.

Time-Memory Trade-off

An intermediate solution is the time-memory trade-off (TMTO). The idea is to use memory to reduce the authentication time, making both memory and time suitable to our application. Note that the goal of the TMTO is here to reach an authentication time that is below the acceptable threshold of 200 ms. Once this requirement is fulfilled, still decreasing the time does not make sense because (i) this implies a memory cost (ii) the authentication time would become a negligible factor in the whole communication time.

In the following, the TMTO version is explored. It uses rainbow tables of maximal size, which are introduced in Chapter 1.

[5]If one wants to index the hashes with (i, j) couples, the memory increases by 25% (32 bits appended to each of the 128-bit hashes).

$$\underbrace{\begin{array}{cccccc}
s_1^0 & s_0^{\kappa} & s_0^{2\kappa} & \dots & s_0^{(\lfloor\frac{L}{\kappa}\rfloor-2)\kappa} & s_0^{(\lfloor\frac{L}{\kappa}\rfloor-1)\kappa} \\
s_2^0 & s_1^{\kappa} & s_1^{2\kappa} & \dots & s_1^{(\lfloor\frac{L}{\kappa}\rfloor-2)\kappa} & s_1^{(\lfloor\frac{L}{\kappa}\rfloor-1)\kappa} \\
\vdots & & & & & \vdots \\
s_i^0 & s_i^{\kappa} & s_i^{2\kappa} & \dots & s_i^{(\lfloor\frac{L}{\kappa}\rfloor-2)\kappa} & s_i^{(\lfloor\frac{L}{\kappa}\rfloor-1)\kappa} \\
\vdots & & & & & \vdots \\
s_n^0 & s_{n-1}^{\kappa} & s_{n-1}^{2\kappa} & \dots & s_{n-1}^{(\lfloor\frac{L}{\kappa}\rfloor-2)\kappa} & s_{n-1}^{(\lfloor\frac{L}{\kappa}\rfloor-1)\kappa}
\end{array}}_{\frac{L}{\kappa}\text{ columns}}$$

Figure 4.3: The rapid-hash table.

4.2 OSK/AO

4.2.1 Description

Avoine and Oechslin propose in [30] to apply a time-memory trade-off to the search procedure of OSK, leading in this way to a variant known as OSK/AO.

The protocol execution is the same, only the search procedure is modified. The reader performs an exhaustive search over the NL possible states using a time-memory trade-off. The complexity of the search procedure thus varies from $O(1)$ to $O(NL)$, depending on the amount of memory devoted to the trade-off. For example, they mention that a complexity of $O((NL)^{2/3})$ can be reached with a memory of size $O((NL)^{2/3})$.

Avoine, Dysli, and Oechslin also suggest in [25] a variant of the OSK protocol that ensures strong authentication, as OSK is originally designed to ensure identification only, without consequently considering replay attacks. To do so, [25] suggests using nonces as follows: the reader sends a nonce r in the authentication request message and the tag answers $G(s_i^j \oplus r)$ along with $G(s_i^j)$. The latter value is used by the reader to identify the tag, and the former to authenticate it.

Another advantage of OSK/AO is that the search done in the identification is intrinsically randomized, which makes timing attacks irrelevant [22].

The specific time-memory trade-off technique introduced in [25, 30] is described below.

In this technique there are two main functions namely a *response generating function* F and a *reduction function* R. F takes two indices as an input (i.e., tag index and life time index) and outputs a tag response such that

$$F(i,j) \mapsto G(H^j(s_i^0)) = r_i^j$$

The reduction function R is such that

$$R(r_i^j) \mapsto (i', j'),$$

where $1 \leq i, i' \leq N$, and $0 \leq j, j' \leq L$.

Note that F requires $j+1$ cryptographic operations to be computed, which would drastically lower the efficiency of the search if it were used directly. What is suggested instead in [30] is to use a second kind of-time-memory trade-off, called the rapid-hash table, to compute F efficiently. This trade-off table is rather straightforward: the secrets s_i^j of the tags are computed from lifetime values 0 to L, but only $\frac{L}{\kappa}$ columns are stored. This is illustrated in Fig. 4.3. As explained in Section 4.2.3, this means that an average of $\frac{\kappa+1}{2}$ cryptographic operations are required per evaluation of F.

4.2.2 Analysis

As explained in Section 4.2.1, there are two things that need to be stored in memory: the rainbow tables and the rapid-hash table. The proportion of memory that should be dedicated to each is discussed hereafter.

Let ρ denote the proportion of memory dedicated to the rainbow tables. The trade-off efficiency follows the rule $T = N^2\gamma/M_{RT}^2$ (see [27, 96]), with γ being a small factor depending on the

Table 4.1: Notations used throughout the paper.

N	Number of tags in the system.
L	Lifetime of a tag in the system (in terms of authentication executions).
ℓ	Number of rainbow tables.
t	Length of the chains in each rainbow table.
s_i^j	Secret of the i-th tag used for the $j+1$-th authentication where $1 \le i \le N$, and $0 \le j \le L$.
H, G	Collision resistant one-way functions.
$H_{\text{fast}}(i,j)$	Function that computes the j-th secret of the i-th, that is $s_i^j = H^j(s_i^0)$. This function uses a precomputed rapid-hash table to compute hashes faster. The construction of this function is showed in Algorithm 3.
κ	Length of the interval between hash indices in the rapid-hash table.
`state`$[i][k]$	`state` is a precomputed two-dimensional array that represents the rapid-hash table. `state`$[i][k]$ stores the $k \times \kappa$-th state of the i-th tag, that is $s_i^{k\times\kappa}$.
$F(i,j)$	The response generating function. It outputs a tag response such that $F(i,j) = G(H_{\text{fast}}(i,j))$, that is the output of tag i in its j-th authentication attempt.
Table$_v$	The v-th rainbow table, where $1 \le v \le \ell$.
$R_w^v(s)$	For w-th column where $1 \le w \le t$ of the v-th table where $1 \le v \le \ell$, a reduction function that maps input s into arbitrary indices (i', j'), where $1 \le i' \le N$ and $0 \le j' \le L$.

probability of success of the trade-off, and M_{RT} the memory dedicated to the rainbow tables (that is ρM). As for the rapid-hash table, we have:

$$\kappa = \left\lceil \frac{N|hash|}{M_{RH}} \right\rceil,$$

with $|hash|$ the size of a hash, and M_{RH} the memory for the rapid-hash table (that is $(1-\rho)M$). Each operation in the rainbow tables requires an average of $\frac{\kappa+1}{2}$ cryptographic operations in the rapid-hash table. Therefore:

$$T = \frac{N^2\gamma}{M_{RT}^2}\frac{\kappa+1}{2} \approx \frac{N^2\gamma}{\rho^2 M^2}\frac{N|hash|}{2(1-\rho)M}.$$

The optimal value of ρ can be found easily by deriving:

$$\frac{\partial T}{\partial \rho} = 0 \quad \Leftrightarrow \quad \frac{\partial}{\partial \rho}\left[\frac{1}{\rho^2(1-\rho)}\right] = \frac{3\rho-2}{(\rho-1)^2\rho^3} = 0,$$

which yields $\rho_{opt} = \frac{2}{3}$. In the following, the memory for the rainbow tables is fixed to two thirds of the total memory[6].

4.2.3 Algorithms

This section presents the algorithms used in OSK/AO, namely (i) the algorithm to compute the rapid-hash table (Algorithm 3), (ii) the algorithm to build the rainbow tables (Algorithm 4), and (iii) the algorithm to identify the tag (Algorithm 5). The material in this section mostly comes from [45]. The notations used in the algorithms are given in Table 7.1.

First, the system randomly generates the initial secrets for all the tags such that $s_i^0 \in_R \{0,1\}^\lambda$ where $1 \le i \le N$, and λ is the length of the secrets. The system defines a κ parameter then computes the interval secret values of all the tags. After that all the secrets are stored into a two dimensional array such that `state`$[i][k] := H^{k\times\kappa}(S_i^0)$ where $k = 0,1,2,\ldots$ and $0 \le k \times \kappa \le L$.

Now, for a given secret of tag i, the j-th rapid-hash computation of the secret is presented in Algorithm 3. The algorithm requires only at most κ hashes by the help of the precomputed RH table. Whenever κ decreases, the memory usage increases but the on-line computation decreases.

[6]Note that this result is compliant with the analysis done in [30]. The development done in this section is somewhat simpler and matches the notations used in the rest of this chapter.

Algorithm 3 Compute $s = H_{\text{fast}}(i, j)$

```
Require: $1 \leq i \leq N, 0 \leq j \leq L$
Ensure: $s = s_i^j$
  $s \leftarrow \texttt{state}[i][\lfloor \frac{j}{\kappa} \rfloor]$
  $a \leftarrow j \bmod \kappa$
  while $a \neq 0$ do
    $s = H(y)$
    $a \leftarrow a - 1$
  end while
  return $s$
```

Algorithm 4 shows the procedure to construct a single rainbow table. For the construction, only two parameters are needed: the number of startpoints used in the precomputation phase (named m_1, as in [27]) and the number of the table to be generated. The startpoints of a rainbow table are fed into the F function sequentially. The output is actually a response of a tag in the system and is fed into the reduction function which outputs arbitrary indices. For a single chain this process is repeated consecutively up to a predefined chain size t, then the starting and endpoints are stored in the table. Finally, each generated endpoint is compared in the table to detect fusions. When two chains generate a fusion, one of them is discarded. This procedure eventually leads to a perfect table.

Algorithm 4 Construction of $Table_v$ (j, m_1, v)

```
Require: $1 \leq j$, $1 \leq m_1 \leq N \times j$ , $v \geq 1$
  $table \leftarrow \{\emptyset\}$
  for $i = 1$ to $\left\lceil \frac{m_1}{j} \right\rceil$ do
    for $k = 0$ to $j$ do
      $nextResp \leftarrow F(i, k)$
      for $w = 1$ to $t - 1$ do
        $z[\,] \leftarrow R_w^v(nextResp)$
        $nextResp = F(z[0], z[1])$
      end for
      $z[\,] \leftarrow R_t^v(nextResp)$
      if $z \notin table$ then
        add the record $\{(i, k); (z[0], z[1])\}$ into $table$
      end if
      if $(i - 1) \times j + k \geq m_1$ then
        break
      end if
    end for
  end for
  clean $table$
  return $table$
```

Finally, Algorithm 5 shows the identification process of a tag by extracting the preimage of a given response using rainbow tables. This part of the system runs during the authentication of a tag. First, $TagResp$ (the answer of the tag) is fed into the reduction function R_t^v and searched among the endpoints of the rainbow table. (i) If a match is found, the corresponding startpoint is iterated as explained in Algorithm 4 up to the $(t-1)^{th}$ reduction function R_t^v in order to get a candidate response. If the candidate response is equal to $TagResp$ then identification is completed. Otherwise (ii) $TagResp$ fed into the reduction function such that $R_{t-1}^v(TagResp)$, then the resulting indices fed into F, and then the resulting response fed into $R_t^v(TagResp_{next})$ consecutively.

Algorithm 5 Identify ($Table_v$, TagResp)

```
Require: TagResp $\in \{0,1\}^\lambda$, $v \geq 1$
Ensure: TagResp $\leftarrow G(y)$
  for $q = t$ down to 1 do
    $nextResp \leftarrow$TagResp
    for $i = q$ to $t-1$ do
      $z[\,] \leftarrow R_i^v(nextResp)$
      $nextResp \leftarrow F(z[0], z[1])$
    end for
    $z[\,] \leftarrow R_t^v(nextResp)$
    if $z \in Table_v$ then
      $\{z'; z\} \leftarrow Table_v(z)$
      $nextResp \leftarrow F(z'[0], z'[1])$
      for $w = 1$ to $q-1$ do
        $\tilde{z}[\,] \leftarrow R_w^v(nextResp)$
        $nextResp \leftarrow F(\tilde{z}[0], \tilde{z}[1])$
      end for
      if $nextResp = TagResp$ then
        return $true$
      end if
    end if
  end for
  return $false$
```

4.3 Experiments and Comparison

This section presents the details of the implementation of OSK/AO and discusses its results. This section is partially based on the master thesis of Muhammed Ali Bingöl [45]. In his master thesis, Muhammed Ali Bingöl did an implementation of OSK/AO for a centralized system. As it was pointed out however, a solution where the OSK chains are all stored is feasible for such systems, and is both simpler and more efficient. Nevertheless, in distributed systems, OSK/AO happens to be much more viable than the full storage solution, in particular when the authentication is performed by low-resource mobile devices, such as PDA's or NFC-compliant cellphones.

4.3.1 Environment

The precomputations are performed with a personal computer having Intel 2.8GHz Core2 Duo processor, 4GB RAM and Windows 7 - 64-bit operating system. The NFC-enabled mobile phone is LG OPTIMUS 4X HD, having 1.5GHz processor and 1GB RAM [125]. The cell phone has an open source Linux-based operating system, Android. This OS has a large community of contributors who develop applications primarily written in a customized version of the Java programming language [170]. The phone supports both ISO/IEC 14443 and ISO/IEC 15693 standards which are the common standards in order to read/write 13.56 MHz contactless smart cards.

For the tags, professional version of ZeitControlers basic card $ZC7.5$ ($ZC-Basic$) were chosen. It has a programmable processor card as hardware environment for protocol implementation [89]. It has a micro-controller with 32kB user EEPROM that holds its own operating system and it has 2.9kB RAM for user data. It supports ISO/IEC 14443. The EEPROM contains the user's Basic code, compiled into a virtual machine language known as P-Code (the Java programming language uses the same technology). The RAM contains runtime data and the P-Code stack. The overview of the system is depicted in Figure 4.4.

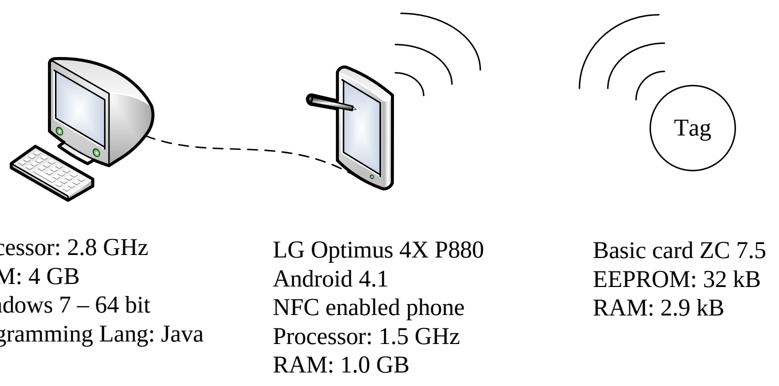


Figure 4.4: Overview of the system [credits: Muhammed Ali Bingöl].

4.3.2 Parameters and Functions

The parameters for the experiments[7] are $N = 2^{20}$, $L = 2^7$ and the one-way functions selected are (4.1) and (4.2).

$$H(x) = AES_K(x) \oplus x, \tag{4.1}$$

$$G(x) = AES_K(x+1) \oplus (x+1), \tag{4.2}$$

where K is a 128-bit constant key. This is known as the Matyas-Meyer-Oseas construction [137]. Its goal is to build a one-way function from a block cipher.

The AES algorithm was used in the construction because it is commonly implemented on fewer gates than classical hash functions (see e.g. [80]), and, in particular, is also available in the ZC7.5. This construction requires only one key schedule during the initialization phase of the tags.

To construct rainbow tables each column of each table uses a different reduction function. The function takes three parameters that are the table index ($v = 0, 1, \ldots, \ell - 1$), the column index ($w = 1, 2, \ldots, t$) and the response output as a byte array ($val[.]$). This function produces two output values; the first one is for the tag index ($i = 0, \ldots, N-1$), the second one is for the lifetime index ($j = 0, \ldots, L-1$). The i value is computed as $i = (Int32(val[v, v+3]) + w) \mod N$ where the function $Int32$ converts a given input 4-byte array into an unsigned 32-bit integer. The j value is computed as $i = (Int32(val[v+1, v+4]) + w) \mod L$. The construction of our reduction functions are given in Algorithm 6.

Algorithm 6 Compute $R_w^v(val[.])$

```
Require: $v \geq 0$, $w \geq 1$
Ensure: $i \in \mathbb{Z}_N$, $j \in \mathbb{Z}_L$
  $i \leftarrow Int32(val[v, v+3]) + w$
  $j \leftarrow Int32(val[v+1, v+4]) + w$
  $i = i \; mod \; N$
  $j = j \; mod \; L$
  return $\{i, j\}$
```

4.3.3 Precomputation of the Tables

In order to use OSK/AO on low-resource devices (such as hand-held readers, PDA's and NFC compliant cellphones), it is necessary to build tables that can fit to small RAMs.

For the total memory there are two parts: (i) the rapid-hash table that stores some intermediate values of the OSK table and (ii) the rainbow tables[8]. Optimal parameters are used, and therefore the values of κ and t are computed such that the memory consumption is as described in 4.2.2.

[7]The parameters are the same than the ones in [25].

[8]A technique known as the prefix-suffix decomposition was used in order to reduce to some extent the size of the rainbow tables. It is described for instance in [48].

Table 4.2: Results of experiments on an NFC compliant cellphone.

Memory Identification time	253MB 15.26ms	113MB 117.54ms
Length of the chains of the TMTO (t)	27	72
Number of chains of the TMTO (m_t)	8968214	3566605
Rapid-hash parameter(κ)	22	43
Authentication rate	99.9%	99.9%

Another significant choice for the time-memory trade-off construction is the probability of success. It should be high enough to avoid false negatives during the authentication process. In the scenario described in this chapter, using $\ell = 4$ rainbow tables of maximal size makes the probability to identify a tag greater than 0.999 (see Theorem 1 in Chapter 5). Note that trying to reach a higher success probability is probably not useful in this situation. It may happen that authentication fails because of noise on the channel for instance, and the successful probability of the rainbow table need only sufficiently low as to not lower the overall probability of success of the whole authentication process.

Finally, regarding the number of startpoints m_1, the trick described in [27] to reduce the precomputation effort is used. Namely, in order to build a perfect table of maximal size, it is technically necessary to build NL chains. However in practice, and as described in [27], fewer chains are computed (but still enough to achieve a table of almost maximal size). In the case described here, about 98% of the maximal number of endpoints are computed by starting with 50 times that number.

In total, the precomputation cost is $\ell \times m_1 \times t$ evaluations of F, which is about $4 \times 50 \times m_t \times t = 400NL$ (see Theorem 2 in Chapter 5). Since these are F evaluations, this number is also multiplied by $\frac{\kappa+1}{2}$ hash operations. For instance, if $\kappa = 6$ and on a server capable of 2^{20} hash operations per second, the precomputation stage would take about 50 hours. Some details about the precomputation of rainbow tables seem to have been overlooked in [25, 30], which would explain their optimistic result. However, one can do much better than that if a table containing the NL secrets is built and used during the precomputation instead of the H_{fast} table. This table needs $NL|hash|$ bits, that is 2GB in our case, and takes about 2 minutes to build on the server. In this second solution, there are actually no hash operations during the building of the rainbow table, making it much faster. In our experimental setting, the whole precomputation process takes about an hour.

4.3.4 Experiments

This section presents experimental validation of Algorithm 5, by measuring the performance of the identification process with randomly chosen tags. The mobile phone that was used [125] is able to compute about 187,750 hashes per second. For both settings, the experiment is run 1,000,000 times. The experimental results are depicted in Table 4.2.

There are three phases on the tag's side: receiving a query, computing the response (two hash calculations), and sending the response. The total time is 70 ms on average, including 50 ms for the calculation of the two hash values and 20 ms for the communication. It can be seen in Table 4.2 that the average identification time is below the 200 ms threshold (including the 70 ms for the tag computation and the communication) even for a memory below 128MB. This shows that one can achieve very fast authentication even with limited memory.

4.4 Conclusion

This chapter focused on the OSK/AO protocol, a scalable privacy-friendly symmetric-key authentication protocol. This protocol is arguably the best solution to date to the scalability issue raised in Chapter 3. It is a variant of the hash-chain-based OSK protocol that uses cryptanalytic time-memory trade-offs constructively to accelerate the search in low-memory settings. The OSK protocol may also be used with a full storage approach, where the chains are completely stored in memory. This technique is however only applicable if the memory is large enough, which is the case in centralized systems for instance.

An implementation of the OSK/AO protocol was realized on an NFC-compliant cellphone and a ZC7.5 contactless tag. The implementation is fully operational and is, to the best of our knowledge, the first implementation of a privacy-friendly authentication protocol based on symmetric-key cryptography. The implementation is suited to large-scale applications, e.g. a million of tags, as this can be the case in mass transportation systems, even on low-resource mobile devices such as hand held readers, PDA's or NFC compliant cellphones. Several experiments carried out on the implemented RFID system show that the results obtained match the theory and are favorable to a practical deployment.

This chapter also illustrated a constructive use case of time-memory trade-offs. OSK/AO uses rainbow tables along with an additional ad-hoc data structure in a creative way and shows that they can be used for more than cracking passwords. The rest of this thesis is dedicated to algorithmic improvements to time-memory trade-offs.

Chapter 5

Rainbow Tables with Fingerprints

Articles related to this chapter

[14] Gildas Avoine and Adrien Bourgeois and Xavier Carpent. Analysis of Rainbow Tables with Fingerprints. *Financial Cryptography and Data Security – FC'15.* (submitted)

In spite of the wide use of cryptanalytic time-memory trade-offs, few significant advances have been done since Oechslin introduced the rainbow tables at Crypto 2003 [147], illustrated with the instant cracking of alphanumerical Windows LM Hash passwords. However, any improvement of their efficiency may render attacks more practical, especially when they are time-constrained. It is thus very important to minimize their cost.

In 2005, Avoine, Junod, and Oechslin [26] introduced a new feature to the rainbow tables, known as the *checkpoints*. The authors observed that more than 50% of the cryptanalysis time is devoted to rule out false alarms. Their technique consists in storing information (e.g., a parity bit) on some intermediate points of the chains alongside the endpoints. During the online phase, when a match with an endpoint occurs, its checkpoints must be compared with the checkpoints of the chain which construction is ongoing. When there is a match of the endpoints, but no match of the checkpoints, the adversary can conclude that a false alarm occurred without re-computing the colliding chain. Although the addition of checkpoints increases the performance of the trade-off, their impact is limited given that their storage consumes additional memory.

In an effort to lower the memory consumption of the rainbow tables, an idea is to truncate the endpoints by a fixed amount of bits. It is not clear where this idea first appeared, but it most likely originated from the (more straightforward) truncation in the distinguished points method [70]. In the case of rainbow tables, it was hinted in [38]. As noted in [121], endpoint truncation however comes with a cost in the form of an increase in false alarms (due to fortuitous matching endpoints).

This chapter revisits checkpoints and endpoint truncation and unifies them in a new model, where the information stored to characterize a chain is named a *fingerprint*. Section 5.1 is a technical introduction to time-memory trade-offs. Section 5.2 introduces the fingerprints and discusses how to build and use rainbow tables with fingerprints. Section 5.3 analyses rainbow tables with fingerprints and in particular their average cryptanalysis time. Section 5.4 presents an algorithm for finding optimal configurations of fingerprints. Theoretical and experimental results are presented in Section 6.5, and the chapter is concluded in Section 5.6.

5.1 Background on Cryptanalytic Time-Memory Trade-offs

5.1.1 Hellman Tables

Although the work regarding TMTO that is done in this chapter relies on rainbow tables, Hellman tables are described below because rainbow tables are heavily based on them. Hellman's method

$$\begin{tikzcd}
X_{1,1} \arrow[r, "roh"] & X_{1,2} \arrow[r, "roh"] & \dots \arrow[r, "roh"] & X_{1,i} \arrow[r, "roh"] & \dots \arrow[r, "roh"] & X_{1,t-1} \arrow[r, "roh"] & X_{1,t} \\
X_{2,1} \arrow[r, "roh"] & X_{2,2} \arrow[r, "roh"] & \dots \arrow[r, "roh"] & X_{2,i} \arrow[r, "roh"] & \dots \arrow[r, "roh"] & X_{2,t-1} \arrow[r, "roh"] & X_{2,t} \\
\vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \vdots \\
X_{j,1} \arrow[r, "roh"] & X_{j,2} \arrow[r, "roh"] & \dots \arrow[r, "roh"] & X_{j,i} \arrow[r, "roh"] & \dots \arrow[r, "roh"] & X_{j,t-1} \arrow[r, "roh"] & X_{j,t} \\
\vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \vdots \\
X_{m,1} \arrow[r, "roh"] & X_{m,2} \arrow[r, "roh"] & \dots \arrow[r, "roh"] & X_{m,i} \arrow[r, "roh"] & \dots \arrow[r, "roh"] & X_{m,t-1} \arrow[r, "roh"] & X_{m,t}
\end{tikzcd}$$

Figure 5.1: Structure of a Hellman table. The framed columns, respectively the startpoints and the endpoints, are the parts stored in memory.

and variants based on it have an online phase complexity of N^2/M^2, with N the size of the input space, and M the amount of memory dedicated to the trade-off.

Precomputation Phase

In the *precomputation phase*, a series of *chains* of hashes[1] is constructed by alternating the h function, and $r : B \to A$, a *reduction function*. The purpose of the reduction function is to map a point in B to a point in A in a uniform and efficient way. Typically, it is implemented with a modulo: $r(y) = y \bmod N$. A chain j starts at an arbitrarily chosen startpoint $X_{j,1} \in A$, and it is iteratively built with $X_{j,i+1} = h(r(X_{j,i}))$ until the endpoint $X_{j,t}$, where the length of the chain t is a fixed parameter. Once m chains are computed this way, the startpoint/endpoint pairs are sorted according to their endpoints, and stored in a table[2]. Figure 5.1 is a representation of the structure of such a table.

Online Phase

Given $y \in B$, the online phase aims to retrieve $x \in A$ such that $y = h(x)$. For that, $r(y)$ is computed and a lookup is performed in the table to check whether it matches a stored endpoint[3]. Assuming it matches $X_{j,t}$ for some $1 \leq j \leq m$, the chain j is rebuilt from $X_{j,1}$ up to $X_{j,t-1}$. If $h(X_{j,t-1}) = y$, then the problem is solved given that $X_{j,t-1}$ is the expected value. If they are not equal or if no match was found in the first place, the process moves up to the next step by computing $r(h(r(y)))$ and repeating the same procedure. This goes on until a solution is found, or until the table is completely searched through (t steps).

The Hellman method is probabilistic: not all points in N are found with equal efficiency, and some are not even covered. However, parameters may be tuned so as to have an adequate probability of success.

A major drawback of Hellman's method is that two colliding chains in a given table lead to a *fusion*. Such artifacts substantially decrease the trade-off performance. Two significant improvements have been introduced to mitigate this problem: the *distinguished points* in 1982 by Rivest [70] and the *rainbow table* in 2003 by Oechslin [147]. The latter is significantly faster in practice [147, 121].

5.1.2 Rainbow Tables

Rainbow tables [147] are an important improvement over Hellman tables, even though the difference is subtle. Instead of a single reduction function within one table, a different one per column is used. An example of a typical reduction function family is $r_i(y) = r(y + i)$, with r a reduction function such as a modulo. The drawback is that, during the online phase, the chain $r_{t-1}(h(r_{t-2}(h(\dots))))$

[1]The technique works for inverting any one-way function, but the term "hash functions" and "hashes" is used for simplicity.

[2]Several tables actually need to be constructed this way. The reason for this is that the coverage quickly saturates in Hellman tables. See [97]. Also note that with rainbow tables, although 3 or 4 tables are generally required for a good probability of success, the number of Hellman tables required is usually much bigger.

[3]Since the endpoints are sorted, this lookup is inexpensive.

$$\begin{tikzcd}
S_1 = X_{1,1} \arrow[r, "r_1 \circ h"] & X_{1,2} \arrow[r, "r_2 \circ h"] & \dots \arrow[r, "r_{i-1} \circ h"] & X_{1,i} \arrow[r, "r_i \circ h"] & \dots \arrow[r, "r_{t-1} \circ h"] & X_{1,t} = E_1 \\
S_2 = X_{2,1} \arrow[r, "r_1 \circ h"] & X_{2,2} \arrow[r, "r_2 \circ h"] & \dots \arrow[r, "r_{i-1} \circ h"] & X_{2,i} \arrow[r, "r_i \circ h"] & \dots \arrow[r, "r_{t-1} \circ h"] & X_{2,t} = E_2 \\
\vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\
S_j = X_{j,1} \arrow[r, "r_1 \circ h"] & X_{j,2} \arrow[r, "r_2 \circ h"] & \dots \arrow[r, "r_{i-1} \circ h"] & X_{j,i} \arrow[r, "r_i \circ h"] & \dots \arrow[r, "r_{t-1} \circ h"] & X_{j,t} = E_j \\
\vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\
S_m = X_{m,1} \arrow[r, "r_1 \circ h"] & X_{m,2} \arrow[r, "r_2 \circ h"] & \dots \arrow[r, "r_{i-1} \circ h"] & X_{m,i} \arrow[r, "r_i \circ h"] & \dots \arrow[r, "r_{t-1} \circ h"] & X_{m,t} = E_m
\end{tikzcd}$$

Figure 5.2: Structure of a rainbow table. The framed columns, respectively the startpoints and the endpoints, are the parts stored in memory.

must be recomputed entirely at each step rather than just computing $r \circ h$ of the previous result. However, it is much easier in rainbow tables to build *clean*[4] tables, which are tables without *merges*. A merge is a situation in which two chains contain an identical point in a given column. The two chains "merge" because the series of values obtained afterwards are the same. A merge appearing in rainbow tables therefore always results in identical endpoints, but it is generally not the case with Hellman tables, in which merges are the cause of a drastic decrease in performances. At the end of the precomputation phase, duplicate endpoints are thus filtered out of rainbow tables. This results in much more efficient tables.

Precomputation Phase

The precomputation phase is very similar to that of the Hellman method. Instead of iterating a function $f : x \mapsto r(h(x))$ to compute a chain, a different reduction function is used in each column. Chains therefore consists of elements computed iteratively using $f_i : x \mapsto r_i(h(x))$, where r_i is the reduction function associated with column i. A typical reduction function family is $r_i : y \mapsto (y + i) \bmod N$, with $N = |A|$. A series of chains is computed in order to form a table.

In order to build a clean table, only one chain per different endpoint is kept. These endpoints, along with their corresponding startpoints, are stored in memory. Furthermore, a table of *maximal size* is obtained when all (or almost all) the possible endpoints are saved, which happens when the number of chains computed is sufficiently large (i.e. when any new chain would have a negligible probability of having an endpoint that is not yet saved). The structure of a rainbow table is shown in Figure 5.2.

As for Hellman tables, the probability of success of a single rainbow table is bounded. In the case of rainbow tables of maximal size, this probability is about 86.47% (see Section 5.1.3. In order to obtain a higher probability of success, multiple tables are used, with a different family of reduction functions per table. A typical number is 4 (achieving a total probability of success of about 99.97%).

In the rest of this thesis, clean rainbow tables of maximal size are assumed, because they offer the best efficiency for a given memory.

Online Phase

The online phase in rainbow tables is again very similar to that of the Hellman method. In order to invert a given y, one starts by computing $r_{t-1}(y)$ and searching through the endpoint list whether there exists j such that $E_j = r_{t-1}(y)$. If so, a chain is rebuilt from the corresponding startpoint S_j in order to compute $X_{j,t-1}$ and verify whether $h(X_{j,t-1}) = y$. If so, the attack succeeds with $x = X_{j,t-1}$, and if not, this match was a false alarm. In that case, or when no matching endpoint is found, the attack proceeds to the next table. Once all tables are cycled[5], the attack proceeds

[4]Although the word "perfect" is usually attributed to tables without merges in the literature, this terminology is more intuitive and seems more adapted.

[5]The order of search in rainbow tables is to search in all tables in a given column, then proceed to the next and so on, rather than searching a table entirely, then proceeding to the next and so on. This is due to the fact that the search is increasingly more expensive towards the left (longer online chain) of the rainbow tables. In the Hellman method, this does not matter.

with the next column, computing $r_{t-1}(h(r_{t-2}(y)))$, and so on until the search succeeds or that all columns are searched through.

5.1.3 Main Results on Rainbow Tables

A thorough analysis of the rainbow tables has been done by Avoine, Junod, and Oechslin in [28]. Below are some results from [28] that are relevant for the discussions of in this chapter and Chapters 6 and 7.

Result 1. *The probability of success of a set of ℓ clean rainbow tables of maximal size with m chains of size t on a problem set of size N is:*

$$P^* = 1 - \left(1 - \frac{m}{N}\right)^{t\ell} \approx 1 - e^{-2\ell}.$$

Result 2. *In a problem of size N, given the N possible chains started at column 1, there are on average m_c different points in column c with:*

$$m_c = \frac{2N}{c+1}.$$

In particular, the average maximum number of chains with different endpoints in a rainbow table of t columns, for a problem of size N is:

$$m_t^{max} = \frac{2N}{t+1}.$$

Result 3. *The optimal parameters for a rainbow table, for a problem of size N, given a memory of M and a desired probability of success P^* are:*

$$\begin{aligned} \ell &= \left\lceil \frac{-\log(1-P^*)}{2} \right\rceil, \\ m &= \frac{M}{\ell}, \\ t &= \frac{\log(1-P^*)}{\ell \log\left(1 - \frac{m}{N}\right)} \approx -\frac{N}{M}\log(1-P^*). \end{aligned}$$

Result 4. *The average number of h evaluations during a search in column k (column 0 being the rightmost one) of a set of ℓ clean rainbow tables of maximal size with chains of size t is:*

$$C_k = k + (t-k+1)q_{t-k+1},$$

with

$$q_i = 1 - \frac{i(i-1)}{t(t+1)}.$$

Result 5. *The average number of h evaluations during the online phase of a set of ℓ clean rainbow tables of maximal size with m chains of size t on a problem set of size N is:*

$$\begin{aligned} T = &\sum_{k=1}^{t} \left(1 - \left(1 - \frac{m}{N}\right)^{\ell}\right) \left(1 - \frac{m}{N}\right)^{(k-1)\ell} \sum_{i=1}^{k} \ell C_k \\ &+ \left(1 - \frac{m}{N}\right)^{t\ell} \sum_{k=1}^{t} \ell C_k, \end{aligned}$$

5.2 Fingerprints

Despite having been analyzed separately (see e.g. [121]), checkpoints and endpoint truncation have never been addressed together. Moreover, and more importantly, good checkpoint positions and truncation amount were up to now found empirically. In this section, a new model is proposed. It encompasses these two improvements of rainbow tables in a sensible and unified way. A technique for determining configurations is discussed in Section 5.4.

5.2.1 Rationale

In regular rainbow tables, chains are composed of the startpoints, which allows one to rebuild that chain without ambiguity, and the endpoints, which are used to select the chain to rebuild in each step of the online phase. Although it is necessary for the startpoints to be points in A for the chain reconstruction to be meaningful, it is not necessary for endpoints to have that property. Indeed, their purpose is solely to compare the online chain with each chain of the table, in order to determine which should be rebuilt (if there is one). An issue with using the endpoint as characterization is that when the online chain merges with a precomputed chain, they cannot be distinguished. This leads to the false alarms, which are the pet hate of the time-memory trade-offs. The fingerprints reduce this problem by providing a better way to characterize the chains.

5.2.2 Description

Definition

In the model presented in this chapter, the characterization of a chain is the list of checkpoints, and the endpoint is considered as a regular checkpoint. Formally, a fingerprint F_j is defined as the concatenation of the outputs of the functions Φ_i applied to each element $X_{j,i}$ of the chain:

$$F_j \quad = \quad \Phi_1(X_{j,1}) \quad || \quad \Phi_2(X_{j,2}) \quad || \quad \dots \quad || \quad \Phi_t(X_{j,t})$$

where $j \in [1, m]$, "$||$" denotes the concatenation, and Φ_i $(1 \leq i \leq t)$ is a *checkpoint function*, used in column i. A checkpoint function is such that:

$$\Phi_i : \quad A \quad \rightarrow \begin{cases} \{0,1\}^{\sigma_i} & \text{if } \sigma_i > 0 \\ \epsilon & \text{otherwise} \end{cases}$$

with $0 \leq \sigma_i \leq \lceil \log_2 N \rceil$. The output of the checkpoint function is called a *checkpoint*. A fingerprint is therefore the concatenation of the checkpoints in the chain. Note that a checkpoint function is expected to have a uniform distribution of its output, as it is the case with reduction functions. A typical checkpoint function $\Phi_i(x)$ returns the σ_i least significant bits of x for instance.

Note that, depending on the configuration (i.e. the value of σ_i in all columns), a fingerprint F_j is not necessarily $\lceil \log_2 N \rceil$ bits long. In fact, fingerprints are typically smaller than the endpoints in regular rainbow tables, as shown in Section 6.5. This allows the rainbow table with fingerprints to contain more chains than what a regular rainbow table would, for the same memory.

Precomputation Phase

Precomputation in rainbow tables with fingerprints is very similar to that of regular rainbow tables. The difference is that during the computation of a chain, the checkpoint functions are applied on each point, such that a fingerprint is computed for that chain. Once a chain is complete, the startpoint, the fingerprint, as well as the endpoint (i.e. the last point of the chain) are all temporarily stored. Similarly to rainbow tables, chains are then sorted according to their endpoints in order to remove the merging chains. The table thus becomes clean (or perfect). Endpoints are then discarded, as they are no longer required. A final step consists in sorting the chains according to their fingerprints, in order to make the search more efficient[6]. Note that at this point, there might be several chains sharing the same fingerprint even though they had different endpoints. However, this is marginal for reasonable configurations. This final step (sorting fingerprints) requires negligible time with respect to the earlier work, much like sorting the endpoints requires negligible time over the computation of the chains. Rainbow tables with fingerprints therefore require the same amount of precomputation as their regular counterparts.

[6]The sort is performed in reverse order, that is the fingerprints are considered flipped, because partial fingerprints $\Phi_c(X_{j,c})|| \dots ||\Phi_t(X_{j,t})$ are searched in the online phase. This ensures that multiple candidate matching fingerprints are contiguous in the list, making the search more efficient.

Online Phase

Again, the algorithm for performing the search in rainbow tables with fingerprints is very close to the one of regular rainbow tables, as described in Chapter 1. However, the checkpoint functions are applied to the online chain, which gives a *partial fingerprint* (rather than an endpoint). The fingerprint is partial because the online chain is shorter than chains of the table, and therefore it might be that not all checkpoints are part of it. The partial fingerprint is then compared to stored fingerprints (the comparison is done for the available bits only). A second particularity is that there might be several fingerprints matching the partial fingerprint of the online chain, leading to possibly several false alarms per step (there can only be one in regular rainbow tables).

The fact that endpoints are possibly not completely part of the fingerprint means that it is possible that two non-merging chains share the same fingerprint. This leads to false alarms, although they are not of the same type as false alarms caused by merges. Consequently, false alarms are divided into two types: Type-I false alarms are those that are merge-induced, and Type-II false alarms are those that are caused by partial fingerprint matching of non-merging chains. Although Type-II false alarms do not appear in regular rainbow tables, the additional cost they incur in tables with fingerprints is more than made up for by the other benefits (in good configurations).

5.3 Analysis

This section provides an analysis of rainbow tables with fingerprints. Theorem 1 presents preliminary results regarding fortuitous collisions in checkpoint functions, and Theorem 2 is presents the average execution cost of a rainbow table with fingerprints, for a given configuration. In this section, the following notations are used, in agreement with the existing litterature on the analysis of rainbow tables: ℓ is the number of tables, m is the amount of chains per table, and t is the length of the chains.

Let ϕ_c be the probability that two different points have the same checkpoint in a given column c:

$$\phi_c := \Pr\left[\Phi_c(x) = \Phi_c(y) | x, y \in A : x \neq y\right]. \tag{5.1}$$

This probability is useful to describe the event of two non-merging chains having the same partial fingerprint.

Theorem 1. *If $N \equiv 0 \pmod{2^{\sigma_c}}$, given a column c, the probability that two different points chosen uniformly at random in A have the same checkpoint in c is:*

$$\phi_c = \frac{N/2^{\sigma_c} - 1}{N - 1}. \tag{5.2}$$

Proof. Given a value x and its corresponding checkpoint $\Phi_c(x)$, there are $N-1$ different possible values $y \neq x$. Among those, there are on average $N/2^{\sigma_c} - 1$ values y such that $\Phi_c(x) = \Phi_c(y)$. $\square$

This theorem assumes that both the points in a column and the checkpoints are uniformly distributed. This is for instance ensured if the reduction and checkpoint functions are modulos, and if the h function has a uniformly distributed output, which is the case in virtually all practical cases. This assumption is already made in previous analyses, such as [28] and [147].

The following theorem gives the average cost of a search using a rainbow table with fingerprints.

Theorem 2. *The average amount of evaluations of h during the online phase using the rainbow tables with fingerprints is:*

$$T = \sum_{k=1}^{\ell t} \frac{m}{N}\left(1 - \frac{m}{N}\right)^{k-1} (W_k + Q_k) + \left(1 - \frac{m}{N}\right)^{\ell t} (W_{\ell t} + Q_{\ell t}), \tag{5.3}$$

with

$$c_i = t - \left\lfloor \frac{i-1}{\ell} \right\rfloor, \qquad q_c = 1 - \prod_{i=c}^{t}\left(1 - \frac{m_i}{N}\right),$$

$$W_k = \sum_{i=1}^{k}(t - c_i), \qquad P_c = \frac{m}{N} + \frac{m_c - m}{N}\phi_c \widetilde{P}_{c+1} + \frac{N - m_c}{N}\phi_c P_{c+1},$$

$$Q_k = \sum_{i=1}^{k}(c_i - 1)(P_{c_i} + E_{c_i}), \qquad \widetilde{P}_c = \frac{1}{t-c+1} + \frac{t-c}{t-c+1}\phi_c \widetilde{P}_{c+1}$$

$$E_c = (m - q_c)\prod_{i=c}^{t}\phi_i, \qquad P_t = \frac{m}{N}, \qquad \widetilde{P}_t = 1.$$

Proof. Given the online phase described in Section 5.2, we have:

$$T = \sum_{k=1}^{\ell t} p_k T_k + p_{\text{fail}} T_{\ell t}, \tag{5.4}$$

with p_{fail} and p_k the probabilities that the attack fails, and that it succeeds after k steps respectively. Here, T_k denotes the average amount of evaluations of h when the attack stops after k steps.

Determination of p_k and p_{fail}:
The probability that the current point is found in a column is m/N, and p_k is the probability that the current point is found in a column, but is not found in the preceding $k-1$ searches:

$$p_k = \frac{m}{N}\left(1 - \frac{m}{N}\right)^{k-1}. \tag{5.5}$$

The probability of failure p_{fail} is simply the probability that the current point is not found in any of the ℓt previous searches, that is:

$$\left(1 - \frac{m}{N}\right)^{\ell t}. \tag{5.6}$$

Determination of T_k:
At each step, h is computed for the following reasons: when building the online chain (this work is noted W), and when filtering false alarms (noted Q), hence $T_k = W_k + Q_k$.

Determination of W_k:
At step k, the fingerprint comparison is done at column

$$c_k = t - \left\lfloor \frac{k-1}{\ell} \right\rfloor.$$

The number of h computations needed for the online chain is the number of columns separating c_k from t, that is $t - c_k$. Consequently,

$$W_k = \sum_{i=1}^{k}(t - c_i). \tag{5.7}$$

Determination of Q_k:
Similarly, for each false alarm, a chain from column 1 to column c_k has to be computed, needing $c_k - 1$ computations. Hence, we have:

$$Q_k = \sum_{i=1}^{k}(c_i - 1)F_{c_i}, \tag{5.8}$$

with F_c the average number of false alarms at column c. False alarms in rainbow tables with fingerprints are of two types. Type-I false alarms are the same as the ones in rainbow tables and

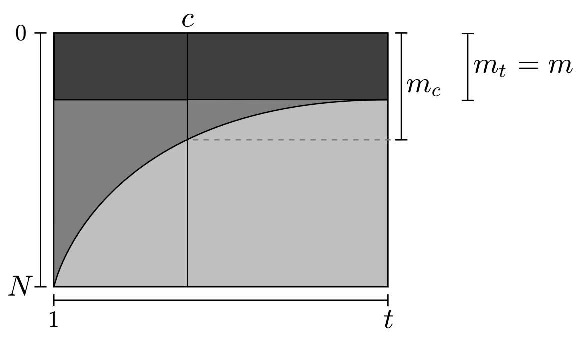


Figure 5.3: Illustration of the different types of merges. At column c, a concrete merge (1) occurs when the online chain is in the dark gray area; an abstract merge (2) occurs when the online chain is in the medium gray area; finally, the online chain is free (3) while it is in the light gray area.

occur because of merges induced by reduction functions. Type-II false alarms occur because two chains may have the same partial fingerprint. P_c and E_c respectively denote the average number of Type-I and Type-II false alarms at column c. In the following, all alarms (that is, including the one true alarm) are counted as false alarms, for the sake of simplicity. The resulting loss in accuracy is negligible.

Determination of P_c:
Because the tables are clean (or perfect), there can be at most one Type-I false alarm per step. This false alarm, if it exists, is due to a chain merging with the online chain, somewhere between c and t. Moreover, for this chain to cause a false alarm, the partial fingerprint must match before the merge as well. The online chain is started by computing $r_c(y)$. There can only be three separate outcomes:

(1) there exists a chain j in the table such that $X_{j,c} = r_c(y)$ ("concrete" merge),

(2) $r_c(y)$ belongs to the $m_c - m$ other points that could also be in the table ("abstract" merge),

(3) $r_c(y)$ is one of the $N - m_c$ remaining points ("free" chain).

These three cases are illustrated in figure 5.3. Although this might not seem intuitive, the cases (2) and (3) are different. This is because when a point is one of the m_c points that could appear in this column, it results in a merge in all cases, while it might not be the case in (3). We have:

$$\begin{aligned}
P_c &= \Pr(\text{Type-I false alarm from column } c) \\
&= \quad \Pr(\text{Type-I FA from column } c \mid \text{concrete merge in } c) \times \Pr(\text{concrete merge in } c) \\
&\quad + \Pr(\text{Type-I FA from column } c \mid \text{abstract merge in } c) \times \Pr(\text{abstract merge in } c) \\
&\quad + \Pr(\text{Type-I FA from column } c \mid \text{free in } c) \times \Pr(\text{free in } c).
\end{aligned}$$

The following details this part by part.

The first factor of the first term is simply equal to one. Indeed, in case (1) the online chain merges entirely with chain j, inevitably leading to a partial fingerprint match. The second factor is equal to $\frac{m}{N}$ because there are N possibilities, and m of them lead to a point in one of the chains.

For the second term, the second factor is equal to $\frac{m_c - m}{N}$, for similar reasons. For the first factor, we have:

$$\begin{aligned}
&\Pr(\text{Type-I FA from column } c \mid \text{abstract merge in } c) \\
&\quad = \Pr(\text{Same checkpoint in } c \wedge \text{Type-I FA from column } c+1 \mid \text{abstract merge in } c) \\
&\quad = \Pr(\text{Same checkpoint in } c \mid \text{abstract merge in } c) \\
&\qquad\qquad \times \Pr(\text{Type-I FA from column } c+1 \mid \text{abstract merge in } c) \\
&\quad = \phi_c \widetilde{P}_{c+1}.
\end{aligned}$$

The computation of $\widetilde{P}_c$ is addressed below. The second equality is valid because the two event in the joint probability are independent. Indeed, given that there is an abstract merge in c, the two chains have not merged yet in c and thus the checkpoint in c does not give any information on a possible future Type-I false alarm. Conversely, the behavior after c does not change the situation in c.

Finally, for the last term, the second factor is equal to $\frac{N-m_c}{N}$, again for the same reasons. For the first factor, we have:

$$\begin{aligned}
&\Pr(\text{Type-I FA from column } c \mid \text{free in } c)\\
&\quad= \Pr(\text{Same checkpoint in } c \wedge \text{Type-I FA from column } c+1 \mid \text{free in } c)\\
&\quad= \Pr(\text{Same checkpoint in } c \mid \text{free in } c)\Pr(\text{Type-I FA from column } c+1 \mid \text{free in } c)\\
&\quad= \phi_c\, P_{c+1}.
\end{aligned}$$

We have that $\Pr(\text{Type-I FA from column } c+1 \mid \text{free in } c) = P_{c+1}$ because the fact of having a free chain in c means that it can lead to any of the N points in column $c+1$. Therefore it does not restrict the situation for the next column. The validity of the second equality can be argued as above.

Finally,

$$P_c = \frac{m}{N} + \frac{m_c - m}{N}\phi_c \widetilde{P}_{c+1} + \frac{N - m_c}{N}\phi_c P_{c+1}. \tag{5.9}$$

For the initial case, when $c = t$, the only solution is that a concrete merge occurs right away, that is:

$$P_t = \frac{m}{N}. \tag{5.10}$$

Determination of $\widetilde{P}_c$:

The probability $\widetilde{P}_c$ is calculated in a very similar way to P_c. The difference is that this is a situation of abstract merge before column c. It will therefore result in a merge in all cases, but not necessarily to a false alarm since for that, all the intermediary checkpoints must match. When computing $r_c(y)$, there are only two possible outcomes (the chain cannot be free again since it has merged):

(1) there exists a chain j in the table such that $X_{j,c} = r_c(y)$ (a concrete merge),

(2) $r_c(y)$ belongs to the $m_c - m$ other points that could also be in the table (a continuation of the abstract merge).

This gives:

$$\begin{aligned}
\widetilde{P}_c &= \Pr(\text{Type-I FA from column } c \mid \text{abstract merge in } c-1)\\
&= \quad \Pr(\text{Type-I FA from column } c \mid \text{abstract merge in } c-1 \wedge \text{concrete merge in } c)\\
&\qquad\quad \Pr(\text{concrete merge in } c \mid \text{abstract merge in } c-1)\\
&\quad + \Pr(\text{Type-I FA from column } c \mid \text{abstract merge in } c-1 \wedge \text{abstract merge in } c)\\
&\qquad\quad \Pr(\text{abstract merge in } c \mid \text{abstract merge in } c-1).
\end{aligned}$$

This is analyzed part by part below.

In the first term, the first factor is equal to one. Indeed, the fact that there is a concrete merge in c supersedes the fact that there was an abstract merge in $c-1$, which leads to the same expression as the corresponding probability in P_c. The second factor may be rewritten as such:

$$\begin{aligned}
&\Pr(\text{concrete merge in } c \mid \text{abstract merge in } c-1)\\
&\quad= \qquad \Pr(\text{online chain merges in } c \mid \text{abstract merge in } c-1)\\
&\qquad\quad \times \Pr(\text{concrete merge in } c \mid \text{online chain merges in } c \wedge \text{abstract merge in } c-1).
\end{aligned}$$

The first factor is equal to $\frac{m_{c-1}-m_c}{m_{c-1}-m}$. Indeed, among the $m_{c-1} - m$ chains that were abstract in $c-1$, there are $m_{c-1} - m_c$ of them that merged in c (that is, $m_{c-1} - m_c$ chains were “lost”). The second factor is equal to $\frac{m}{m_c}$: among the m_c possible chains to merge with, m chains correspond to a concrete merge. Using the approximation from Result 2, we can rewrite:

$$\frac{m_{c-1} - m_c}{m_{c-1} - m} \frac{m}{m_c} = \frac{\frac{2N}{c} - \frac{2N}{c+1}}{\frac{2N}{c} - \frac{2N}{t+1}} \frac{\frac{2N}{t+1}}{\frac{2N}{c+1}} = \frac{(c+1)(\frac{1}{c} - \frac{1}{c+1})}{(t+1)(\frac{1}{c} - \frac{1}{t+1})} = \frac{1}{t-c+1}.$$

In the second term of $\widetilde{P}_c$, the second factor is the complement of the probability discussed above, that is $\frac{t-c}{t-c+1}$. For the first factor, again the fact that there is an abstract merge in $c-1$ is superseded by the fact that there is an abstract merge in c. This leads to the same expression as the corresponding probability in P_c, that is $\phi_c \widetilde{P}_{c+1}$.

To summarize, we have:

$$\widetilde{P}_c = \frac{1}{t-c+1} + \frac{t-c}{t-c+1} \phi_c \widetilde{P}_{c+1}. \tag{5.11}$$

For the initial case when $c = t$, a merge may only be concrete, and therefore:

$$\widetilde{P}_t = 1. \tag{5.12}$$

Determination of E_c:
The probability of having a merge between columns c and t, already identified in [28], is $q_c = 1 - \prod_{i=c}^{t} \left(1 - \frac{m_i}{N}\right)$. The probability that the online chain merges with a chain of the table is q_c, and therefore, there are on average $m - q_c$ non-merging chains. Among those, each creates a Type-II FA with probability $\prod_{i=c}^{t} \phi_i$, which gives:

$$E_c = (m - q_c) \prod_{i=c}^{t} \phi_i. \tag{5.13}$$

Using equations (5.5), (5.6), (5.7), (5.8), (5.9), (5.11), and (5.13) into (5.4) gives (7.1), allowing us to conclude. □

5.4 Algorithm for Finding Optimal Configurations

This section presents an algorithm for determining the configuration of a rainbow table with fingerprints. Up to now, the configurations for checkpoints and endpoint truncation were found empirically.

5.4.1 Hill Climbing

Hill Climbing [168] is a local search technique designed to obtain a local optimum of a function f using an iterative procedure. If f has a single local optimum which is thus the global optimum (in our case, minimum), it will be found by in a finite (and small) number of steps with Hill Climbing. The idea is the following. Let $f : X \to \mathbb{R}$ be the target function of which one wants to find the global minimum. An initial $x_0 \in X$ is chosen, and each step consists in exploring the neighbors of the current point, evaluating f in these neighbor positions, and comparing these values to that of the current position. The new current point is then set to be the neighbor that minimized f among all the neighbors. The search goes on until a situation where all neighbor have small values through f than the current point, which consists in a local minimum.

The definition of neighbor depends on the application, but when minimizing a discrete function, it is rather straightforward. An additional improvement that is used in order to accelerate the search is starting with a large step, and decreasing it gradually whenever the search stalls, until the step is sufficiently small.

See [168] for a more thorough description of the Hill Climbing algorithm.

5.4.2 Application to Checkpoint Functions

As shown in Section 5.3, the performance of the rainbow tables with fingerprints strongly depends on their configuration.

One way to find the best configuration for the checkpoint functions is to apply a brute-force technique to compute T for the $(1 + \lceil \log_2 N \rceil)^t$ possibilities (from 0 to $\lceil \log_2 N \rceil$ bits included for each column), and keep the one that minimizes T. This however is not feasible for any practical instance, because the parameter space is too large. Instead, Hill climbing is used, as described in Section 5.4.1, to compute these configurations efficiently. In order to apply it, the two following assumptions are made.

First, all the non-ϵ checkpoint functions, except the one in column t, output exactly one bit. This hypothesis makes sense because one two-bit checkpoint in a column or two one-bit checkpoints in adjacent columns give nearly identical results. Moreover, it has been observed experimentally that good configurations tend to have their non-ϵ checkpoints scattered, except in the last column where a more important concentration is more efficient. This tendency is most likely explained by Type-II false alarms.

Then, it is assumed that the local minimum is a global minimum (that is, T is unimodal with respect to the positions of the one-bit checkpoints). This has been observed experimentally.

Note that this technique is conservative in the sense that if either assumption is not verified, then the real optimal configuration can only lead to yet better performance. In the following, the "optimal configurations" refer to the best solutions found using this approach. The determination of the configuration is of course only done once, before the precomputation phase, and adds a marginal overhead to its cost.

The overall technique used to find the optimal configurations is displayed in Algorithm 7. The variables `nbits` and `ncp` represent respectively the number of bits for a fingerprint (that is, $\texttt{nbits} = \sum_{i=1}^{t} \sigma_i$) and the number of one-bit non-final checkpoints (that is, $\texttt{ncp} = \texttt{nbits} - \sigma_t$). The values $\texttt{nbits}^{\min}$ and $\texttt{nbits}^{\max}$ represent the range of search of `nbits`. In theory, this goes from 1 up to $t \times \lceil \log_2 N \rceil$, but it is impractical to explore that range. These bounds need to be defined heuristically. Experiments seem to point out that it is hardly necessary to search beyond about $[0.75 \lceil \log_2 N \rceil, \ldots, 1.25 \lceil \log_2 N \rceil]$. The same holds for the values $\texttt{ncp}^{\min}$ and $\texttt{ncp}^{\max}$, which represent the range of search of `ncp`. Again, in theory these bounds are respectively 0 and `nbits`, but in practice, the optimal configuration is in most cases in a range of about $[0.1\,\texttt{nbits}, \ldots, 0.4\,\texttt{nbits}]$.

Algorithm 7 Algorithm for finding optimal configurations of rainbow tables with fingerprints.

```
Require: $N, M, \ell$
  bestcfg ← $\emptyset$
  for nbits ∈ [$\texttt{nbits}^{\min}, \ldots, \texttt{nbits}^{\max}$] do
    $m \leftarrow M/(\ell \times \texttt{nbits})$
    $t \leftarrow 2N/m - 1$
    for ncp ∈ [$\texttt{ncp}^{\min}, \ldots, \texttt{ncp}^{\max}$] do
      newcfg ← Hill Climbing on $T$ with ncp one-bit checkpoints and a final (nbits − ncp)-bits checkpoint
      if $T$(newcfg) < $T$(bestcfg) then
        bestcfg ← newcfg
      end if
    end for
  end for
  return bestcfg
```

Finally, there is a clear constancy in the optimal positions for the one-bit checkpoints, as represented in Figure 5.4. One can notice that, for a given $\texttt{ncp}^{\text{opt}}$, the relative positions of the one-bit checkpoints remain the same. When $\texttt{ncp}^{\text{opt}}$ changes (as is the case between $N = 2^{44}$ and $N = 2^{46}$ for instance), the relative checkpoint positions change as well, although they retain the same structure. Although no analytic way to use this was found, one could spare the somewhat tedious procedure exposed in this section by using known optimal relative checkpoint positions, or using them as initial startpoints for the Hill Climbing search.

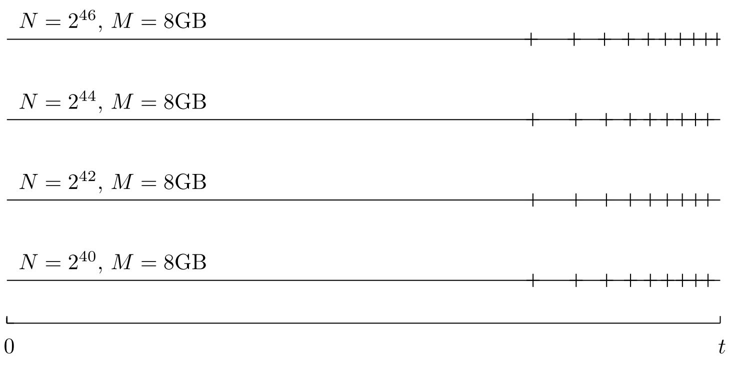


Figure 5.4: Repartition of one-bit checkpoints relatively to t (perfect tables with $\ell = 4$).

Table 5.1: Gain due to fingerprints over regular rainbow tables for various N and M.

	2^{38}	2^{40}	2^{42}	2^{44}	2^{46}	2^{48}
2GB	32.48%	35.94%	39.01%	41.73%	44.16%	46.35%
4GB	30.76%	34.36%	37.54%	40.36%	42.88%	45.14%
8GB	29.04%	32.76%	36.05%	38.97%	41.58%	43.93%

5.5 Theoretical and Experimental Results

5.5.1 Theoretical Results

Table 5.1 presents the gain $1 - T_{\text{fingerprint}}/T_{\text{regular}}$ between rainbow tables with fingerprints in optimal configurations and regular rainbow tables, for different sizes of key space and memory dedicated to the trade-off. The value $\ell = 4$ is considered in both cases[7], and m and t are set for regular rainbow tables to the optimal values as presented in [28]. For the fingerprint version, Algorithm 7 is used to find the optimal configurations.

Table 5.1 is filled with memory sizes that belong to a reasonable range for an average personal computer. Note that M denotes the memory dedicated for endpoints/fingerprints only (adding the startpoints about doubles the memory required). The problem sizes are also driven by practical considerations, to avoid a prohibitive online search time. Analyzing the results leads to two trends. The advantage of the rainbow tables with fingerprints over the regular version tends to increase as the memory decreases. Secondly, the gain increases with the problem size. This behavior can be explained by the fact that when N is large, or when M is small, t and therefore T increases, and there is thus more freedom in the configuration of the checkpoint functions.

Table 5.2 lists differences in the parameters and results between rainbow tables with fingerprints in optimal configuration and regular rainbow tables in several settings. Again, the memory M is the one dedicated to endpoint/fingerprint storage (identical in both cases). The row "Positions" corresponds to the set of columns associated with a one-bit checkpoint. In the optimal configurations found, all checkpoints but the one in the last column consist of at most one bit, as described in Section 5.4.2. The use of the following checkpoint functions are assumed:

$$\begin{aligned} \Phi_i : A &\rightarrow \{0,1\}^{\sigma_i} \\ x &\mapsto \begin{cases} \mathrm{lsb}_{\sigma_i}(x) & \text{if } \sigma_i > 0 \\ \epsilon & \text{otherwise,} \end{cases} \end{aligned}$$

[7]This value for ℓ represents an overwhelming success probability, and is the default in Ophcrack [148], for instance. Other values of ℓ lead to very similar results.

where $\text{lsb}_n(x)$ is a function that outputs the n least significant bits of x.

One can observe that when additional memory is available, the optimal solutions tend to use some extra memory per chain, and vice versa. Additionally, it is noteworthy that the cost of a false alarm for rainbow tables with fingerprints is around one third of the one for regular rainbow tables.

Table 5.2: Analytical performance for the best configurations of rainbow tables with fingerprints (perfect tables with $\ell = 4$).

(a) $N = 2^{48}$, $M = 2\text{GB}$

	regular rainbow tables	rainbow tables with fingerprints
m	4.47×10^7	4.88×10^7
$\lvert E_j\rvert, \lvert F_j\rvert$	48	40
t	1.26×10^7	1.15×10^7
T	2.32×10^{13}	1.24×10^{13} (-46.35%)
Average FA cost	1.33×10^{13}	0.41×10^{13} (-68.88%)
Positions	8476358, 9172110, 9663530, 10050060, 10371170, 10647046, 10889558, 11106326, 11302572, 11482032	

(b) $N = 2^{48}$, $M = 4\text{GB}$

	regular rainbow tables	rainbow tables with fingerprints
m	8.95×10^7	9.65×10^7
$\lvert E_j\rvert, \lvert F_j\rvert$	48	41
t	6.29×10^6	5.83×10^6
T	5.79×10^{12}	3.18×10^{12} (-45.14%)
Average FA cost	3.33×10^{12},	1.06×10^{12} (-68.20%)
Positions	4285001, 4636802, 4885286, 5080733, 5243100, 5382597, 5505222, 5614831, 5714062, 5804807	

(c) $N = 2^{48}$, $M = 8\text{GB}$

	regular rainbow tables	rainbow tables with fingerprints
m	1.79×10^8	1.89×10^8
$\lvert E_j\rvert, \lvert F_j\rvert$	48	43
t	3.15×10^6	2.98×10^6
T	1.45×10^{12}	0.81×10^{12} (-43.93%)
Average FA cost	8.31×10^{11}	2.58×10^{11} (-68.99%)
Positions	2155147, 2334199, 2460731, 2560281, 2642997, 2714070, 2776554, 2832410, 2882981, 2929230, 2971872	

5.5.2 Experimental Validation

A time-memory trade-off with rainbow tables with fingerprints has been implemented in order to illustrate the theory with experimental results. NTLM Hash alphanumeric (both lowercase and uppercase) passwords of length 1 to 7 were considered. This represents a search space of $N = \sum_{i=1}^{7} 62^i \approx 2^{41.70}$.

The parameters are $m = 5.03 \times 10^8$, $t = 13554$, $\ell = 4$. Prefix/suffix decomposition [26] was also used in order to save some extra memory, but this has no influence on the online performance (it decreases M, the memory used in practice, but not m, the number of chains). The four tables take up about 14.8GB in total. Using the methodology described in Section 5.4.2, the following

Table 5.3: Theoretical and experimental (average of 25000 searches) results for NTLM problem.

	Theoretical	Experimental
# operations total ($\times 10^6$)	19.88	19.29
# operations for false alarms ($\times 10^6$)	7.28	7.15
# false alarms	716.57	697.15

configuration was found:

$$\Phi_i(x) = \begin{cases} \mathrm{lsb}_1(x) & \text{if } i \in \{10077, 10928, 11530, 12004, 12398, 12736, 13034, 13301\} \\ \mathrm{lsb}_{34}(x) & \text{if } i = 13554 \\ \epsilon & \text{otherwise.} \end{cases}$$

The results of our experiment are presented in Table 5.3. It shows that the practice matches with the theoretical estimations.

The precomputing phase for these tables took roughly a month on about a hundred machines. The online phase takes place on a machine with a i7-3770 CPU and 16GB of RAM. Recovering any alphanumeric NTLM Hash password (whose length is 1 to 7 characters) in this setting takes 3.5 seconds on average.

5.6 Conclusion

The fingerprint model for rainbow tables highlights that endpoints (truncated or not) and checkpoints have the same nature. It provides a more intuitive and natural way to describe the rainbow trade-off than endpoints, checkpoints, and truncation taken separately.

The technique discussed for finding optimal configurations, although biased towards configurations of a certain type (scattered one-bit checkpoints along with a final, multi-bits checkpoint), is practical and systematic. In typical scenarios, such configurations present a speedup of about two with respect to regular rainbow tables.

Chapter 6

Rainbow Tables Optimal Storage

Articles related to this chapter

[17] Gildas Avoine and Xavier Carpent. Optimal Storage for Rainbow Tables. In Sukwoo Kim and Seok-Yeol Kang, editors, *International Conference on Information Security and Cryptology – ICISC 2013*, Seoul, Korea, November 2013.

Chapter 5 presents an variant of rainbow tables that improves their efficiency. This chapter studies their storage. Although algorithmic improvements are important, implementation optimizations are also very valuable: while it is stated in [147] that rainbow tables present a gain of a factor 2 in time with respect to Hellman tables for instance, a gain of 3 can be gained through storage optimizations alone. Despite some having been discussed in the past [26, 28, 48, 121], they have never been the focus of a rigorous analysis and the optimal parameters have consequently never been investigated. This is the subject of our work [17], presented at ICISC 2013, and the focus of this chapter.

Although the techniques discussed throughout this chapter work in other trade-off algorithms such as Hellman's, the focus is on rainbow tables, that are significantly more efficient in practice [121, 147]. Likewise, and as discussed in Chapter 5, the analysis is done on clean tables of maximal size, because they offer the best efficiency for a given memory.

The conventions and notations regarding cryptanalytic time-memory trade-offs introduced in Section 5.1 are used throughout this chapter. Section 6.1 discusses rainbow table storage and presents a lower bounds for endpoint storage. Section 6.2 presents the Prefix/Suffix Decomposition technique used in past implementations to compress endpoints, and provides an analysis of its optimal parameterization. A more efficient technique for endpoint compression, dubbed "Compressed Delta Encoding," is introduced and analyzed in Section 6.3. Section 6.4 discusses the compression of startpoints. Section 6.5 discusses practical details, presents results, and puts them in perspective. The chapter is concluded in Section 6.6.

6.1 Bound for Endpoint Storage

The memory available for the tables is a very important factor. Recall from Chapter 1 that cryptanalytic time-memory trade-offs follow the rule $T \propto N^2/M^2$. More memory means faster inversion and/or bigger searching space. It is therefore an interesting objective to reduce the memory required to store the tables. Note that, because of the nature of the search process, an efficient random access to the chains is necessary. This means that one can not simply compress the tables using a regular entropy compression technique (e.g. deflate, lzw).

The data stored in rainbow tables are the startpoints and the endpoints. Their analysis may be done separately, as motivated in Section 6.4. The first part of the discussion is focused on endpoint storage.

In the naive algorithm, each endpoint is stored on $\lceil \log_2 N \rceil$ bits. Therefore, the total memory for endpoints M_{ep} is:

$$M_{\text{ep}}^{\text{orig}} = m\lceil \log_2 N \rceil. \tag{6.1}$$

Before discussing how this can be reduced, a theoretical lower bound for endpoint storage is discussed here. A natural assumption that is made is that the endpoints are uniformly distributed in A. Indeed, it is assumed that the output of h is uniformly distributed in B and that therefore the output of $r_i \circ h$ is uniformly distributed in A (i.e., expected to behave like a random oracle)[1]. A second point is that, since the endpoints are free of duplicates, the number of possible sets of endpoints is $\binom{N}{m}$ (they are all equiprobable). Therefore, the average minimal number of bits to store one such set of endpoints is:

$$M_{\text{ep}}^{\text{opt}} = \log_2 \binom{N}{m}. \tag{6.2}$$

Indeed, if one could index each possible endpoint set by an integer, this would be the size of one such index.

Note that this bound regards the storage of the full endpoints only. Techniques that truncate them (see Chapter 5) reduce the memory usage further, at the cost of increased online time.

6.2 Decomposition of the Endpoints in Prefix and Suffix

6.2.1 Description

The technique that is used in current implementations (see e.g. Ophcrack [148]) for storing endpoints efficiently is the *prefix-suffix decomposition* of the endpoints. It is discussed in [26, 28]. The technique takes advantage of the fact that endpoints are sorted to store them more efficiently. In the following, it is assumed that $N = 2^n$ for the sake of simplicity, but the technique also works if N is not a power of two (see e.g. [26]).

The endpoints are sorted and divided in two parts: the p most significant bits form the *prefix*, and the $s = n - p$ (with $N = 2^n$) least significant ones the *suffix* of an endpoint. They are then stored in two separate files: the *prefix file* and the *suffix file*. The suffix file simply contains all the suffixes, in the order of the sorted whole endpoints. The suffix file will also typically contain the startpoints corresponding to each suffix. The prefix file contains a list of indices relating to the suffix file. Each index indicates up to which suffix the corresponding prefix maps to.

This is illustrated with an example with $n = 32$ (that is a space size of $N = 2^{32}$), a prefix size of $p = 12$ and a suffix size of $s = 20$. Figure 6.1 represents a possible list of sorted endpoints in binary format. The first entry of the prefix table is 62, because the prefix 000000000000 is used up to chain 62; the next entry is 198 because prefix 000000000001 is used up to there, and so on. Whenever a prefix actually does not appear in the list of endpoints (this is possible although unlikely if p is adequate), the index put in the prefix table is simply the same as the previous one. In the previous example, the index related to prefix 000000000010 is also 198 because it does not appear in the list of endpoints.

For the online phase, what needs to be specified is the operation that gives the startpoint corresponding to some point x, or nothing if x is not in the endpoint list. For that, x is again decomposed in a prefix and suffix (with the same p and s parameters). The prefix of x, let's say 011101011100, is used to fetch the corresponding entry in the prefix table. This gives a number, let's say 11152, as well as the value just before this one, let's say 10440 (note: if the entry is the first chunk, then the previous index is simply 0). So, by construction, if x appears in the list of endpoints, it must lie between entry 10441 and entry 11152 of the suffix table (included). A simple binary search (or linear for that matter, the range should be very small anyway) in that area of the suffix table is then carried out to find the right entry. The fetching operation is therefore very comparable to the naive approach in terms of speed.

Finally, note that both prefix-suffix decomposition and compressed delta encoding (described in Section 6.3) add negligible overhead in time in the precomputation and online phases. Indeed,

[1]This is the case with cryptographic hash functions, which are the focus of time-memory trade-offs such as rainbow tables. Work has been done on more general (but less efficient) time-memory trade-offs for any function [82].

	p	s
chain 0	000000000000	00001011010100110110
chain 1	000000000000	00001100110101101001
chain 2	000000000000	00010001010101001010
chain 3	000000000000	00010111110001010011
	...	...
chain 62	000000000000	10010101010111011100
chain 63	000000000001	00000010110101101001
	...	...
chain 198	000000000001	11010101111101101011
chain 199	000000000011	00000100010011110111
	...	...
chain $m-1$	111111111111	11010100110000011110

$\Downarrow$

Prefix file

(implicit prefix 0)	62
(implicit prefix 1)	198
(implicit prefix 2)	198
...	...
(implicit prefix $2^p - 1$)	$m - 1$

Suffix file

chain 0	00001011010100110110
chain 1	00001100110101101001
chain 2	00010001010101001010
chain 3	00010111110001010011
	...
chain 62	10010101010111011100
chain 63	00000010110101101001
	...
chain 198	11010101111101101011
chain 199	00000100010011110111
	...
chain $m-1$	11010100110000011110

Figure 6.1: Prefix-suffix decomposition example with parameters $N = 2^{32}$, $p = 12$, $s = 20$.

the added cost consists in a handful of simple operations (shifts, additions, ...) versus typically thousands of cryptographic hashes to compute a chain.

6.2.2 Analysis and Optimality

Let p be the size in bits of the prefix, and $s = n - p$ the size in bits of the suffix. The total memory used for endpoints is therefore:

$$M_{\text{ep}}^{\text{ps}} = 2^p \lceil \log_2 m \rceil + ms. \tag{6.3}$$

The first term is for the prefix file, with each possible prefix having an index in the suffix file. Since the latter has m entries, the size of this index is $\lceil \log_2 m \rceil$. The second term is for the suffix file and its size is straightforward.

Theorem 3. *The optimal parameter p^{opt} for the prefix-suffix decomposition is one of $\lfloor p^* \rfloor$ or $\lceil p^* \rceil$, where*

$$p^* = \log_2 \frac{m}{\lceil \log_2 m \rceil \log 2}. \tag{6.4}$$

Proof. First note that (6.3) is convex, because:

$$\frac{\partial^2 M_{\text{ep}}^{\text{ps}}}{\partial p^2} = 2^p \lceil \log_2 m \rceil \log^2 2 > 0.$$

A simple way to find the optimal value is thus to find the minimum of the relaxed optimization problem, where p is a real-valued parameter, and test the two neighboring integer values. We have:

$$\frac{\partial M_{\text{ep}}^{\text{ps}}}{\partial p} = 2^p \lceil \log_2 m \rceil \log 2 - m.$$

Therefore,

$$p^* = \log_2 \frac{m}{\lceil \log_2 m \rceil \log 2},$$

and the two neighboring integers are $\lfloor p^* \rfloor$ and $\lceil p^* \rceil$. Finding which of the two is best is simply a matter of evaluating (6.3). $\square$

6.3 Compressed Delta Encoding of the Endpoints

6.3.1 Description

A new technique called Compressed Delta Encoding is introduced in this section. The technique consists in computing the vector of differences between each consecutive endpoint[2], which is then compressed using Rice coding [164] (this choice is argued in Section 6.3.2). This results in endpoint differences of varying size (small differences being stored on fewer bits, and larger ones in more bits).

The online phase requires to efficiently perform random accesses on the elements stored in the table. In order to make the encoding efficient, an additional index table is computed and stored. The space A is divided into L blocks of the same size. The start of each block is indexed in a dedicated area of the memory, which size should be small compared to the compressed endpoints. The index table contains L pairs of values that each indicate the starting (bit) position of the corresponding block and the number of the chain. The former is used to jump into the right block, and the latter is used to know what startpoint to use in case the point is found.

When the differences are computed during the offline phase, at the start of each i-th block, the first difference is computed with respect to $\lfloor \frac{iN}{L} \rfloor$, and not the last endpoint encoded. The reason is that if the difference was computed with respect to the previous endpoint instead, one would need to decompress all the endpoints from the beginning. The resulting gain in space due to the smaller differences is negligible.

During the online phase, given x, the corresponding block is found by computing $\lfloor \frac{xL}{N} \rfloor$, go to the address pointed by that block by looking up in the small index table, and start recomputing the sum of the differences with the offset $\lfloor \lfloor \frac{xL}{N} \rfloor \frac{N}{L} \rfloor$ (i.e. the start of the corresponding block). Once the sum is bigger or equal to x, the search is over (if it is equal, then a matching endpoint is found, and if it is bigger, no such point exists). On average, $\frac{m}{2L}$ decodings are required for each search (each block contains on average $\frac{m}{L}$ compressed endpoints). Experiences show that a number of blocks of about $L = \lfloor \frac{m}{2^8} \rfloor$ is reasonable for practical applications, as explained in Section 6.5.1.

Section 6.3.2 describes the technique quantitatively. A complete example of compressed delta encoding is given in Section 6.3.3.

6.3.2 Analysis and Optimality

Let E_i denote the i^{th} sorted endpoint and $D_i := E_{i+1} - E_i - 1$. We have:

$$\Pr(D_i = d) = \begin{cases} \frac{\binom{N-d-1}{m-1}}{\binom{N}{m}} & \text{if } 0 \le d \le N - m, \\ 0 & \text{else.} \end{cases} \tag{6.5}$$

Indeed this corresponds, among all possible choices for m endpoints in A, to choosing the other $m - 1$ endpoints among the $N - d - 1$ values left. Since the probability does not depend on i, we simply note $D := D_i$.

[2]Note that endpoints are unique meaning that a zero difference is not possible. One can therefore also decrease the differences by one.

Theorem 4. *The expected value of the difference (diminished by one) D between two consecutive endpoints is:*

$$\mathbb{E}[D] = \frac{N-m}{m+1} \tag{6.6}$$

Proof. We have $\mathbb{E}[D] = \sum_{d=0}^{\infty} d \Pr(D = d)$. This is the expression that is addressed in p.175–177 of [88]. $\square$

One can observe that the probability mass function (6.5) has a striking similarity with a geometric distribution. Let D' be a geometrically distributed random variable having the same average as D, that is:

$$\Pr(D' = d) = \left(\frac{N-m}{N+1}\right)^d \frac{m+1}{N+1}.$$

Geometrically distributed data is best compressed when using schemes known as *exponential codes* such as *Rice coding* [164] (Rice coding of parameter k is a special case of Golomb coding of parameter $m = 2^k$). Rice coding (see [164] for a thorough description) is a lossless data compression scheme that creates variable-sized codes. Given an input integer x to compress, its corresponding code is a binary string comprised of $\lfloor x/2^k \rfloor$ times the bit "1" followed by a bit "0", and finally the k least significant bits of x. For instance with a Rice parameter of $k = 3$, the integer "00110101" is compressed to "1111110101", and the integer "00000110" is compressed to "0110".

It has been shown (see [83]) that Golomb codes are optimal to compress a geometrically distributed source. In order to select the parameter k that minimizes the rate (the average size of a code), the results of [117] can be used. In the case of compressing endpoints, this leads to Theorem 5.

Theorem 5. *The optimal parameter k^{opt} for the Rice coding of the differences (diminished by one) of the endpoints is:*

$$k^{\text{opt}} = 1 + \left\lfloor \log_2 \left(\frac{\log(\varphi - 1)}{\log \frac{N-m}{N+1}} \right) \right\rfloor, \tag{6.7}$$

with φ the golden ratio $(1+\sqrt{5})/2$. The rate of the corresponding code is:

$$R_{k^{\text{opt}}} = k^{\text{opt}} + \frac{1}{1 - \left(\frac{N-m}{N+1}\right)^{2^{k^{\text{opt}}}}} \tag{6.8}$$

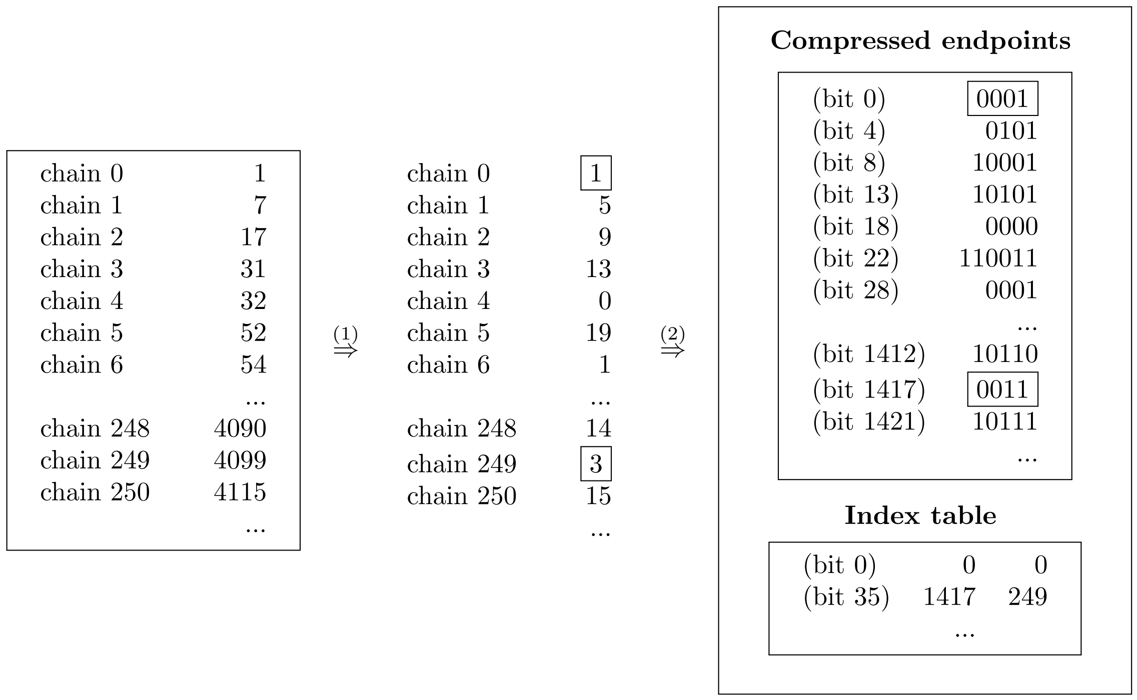
Proof. See Section 3.A in [117], with $\mu = \frac{N-m}{m+1}$ (from eq. (6.6)). $\square$

The memory usage of compressed delta encoding is determined as follows. Each of the m compressed endpoints requires on average $R_{k^{\text{opt}}}$ bits as showed in Theorem 5. Additionally, each entry in the index table comprises the position in bits of the beginning of its block, which requires $\lceil \log_2 mR_{k^{\text{opt}}} \rceil$ bits, and the chain number for possible chain reconstruction, which requires $\lceil \log_2 m \rceil$ bits. The total memory follows easily:

$$M_{\text{ep}}^{\text{cde}} = mR_{k^{\text{opt}}} + L\left(\lceil \log_2 mR_{k^{\text{opt}}} \rceil + \lceil \log_2 m \rceil\right). \tag{6.9}$$

6.3.3 Example

Figure 6.2 presents an example of compressed delta encoding. The parameters are a space size of $N = 2^{20}$, a number of chains of $m = 2^{16}$, a number of blocks $L = \frac{m}{2^8} = 2^8$, and a Rice parameter $k = k^{\text{opt}} = 3$. The first step consists in computing the delta encoding of the sorted endpoints (left box) minus one. The list is divided in L blocks, with the beginning of each block marked (these are the framed numbers in Figure 6.2). For instance, the second block should begin right after $\frac{N}{L} = 4096$, which corresponds in this case to chain 249. The difference is computed with respect to 4096 rather than the previous endpoint 4090, in order to make the reconstruction possible. In the second step, Rice compression is applied to these differences (top right box), and the index is built (bottom right box). In the index, each block is mapped with its corresponding starting bit as well as the corresponding chain number. Each entry in the index is 35 bits long, $\lceil \log_2 mR_{k^{\text{opt}}} \rceil = 19$ for the bit address of the block and $\lceil \log_2 m \rceil = 16$ for the chain number.



chain 0	1
chain 1	7
chain 2	17
chain 3	31
chain 4	32
chain 5	52
chain 6	54
	...
chain 248	4090
chain 249	4099
chain 250	4115
	...

$\overset{(1)}{\Rightarrow}$

chain 0	$\boxed{1}$
chain 1	5
chain 2	9
chain 3	13
chain 4	0
chain 5	19
chain 6	1
	...
chain 248	14
chain 249	$\boxed{3}$
chain 250	15
	...

$\overset{(2)}{\Rightarrow}$

Compressed endpoints

(bit 0)	$\boxed{0001}$
(bit 4)	0101
(bit 8)	10001
(bit 13)	10101
(bit 18)	0000
(bit 22)	110011
(bit 28)	0001
	...
(bit 1412)	10110
(bit 1417)	$\boxed{0011}$
(bit 1421)	10111
	...

Index table

(bit 0)	0	0
(bit 35)	1417	249
	...	

Figure 6.2: Compressed delta encoding example with parameters $N = 2^{20}$, $m = 2^{16}$, $L = \frac{m}{2^8} = 2^8$, and $k = k^{\text{opt}} = 3$. Step (1) is delta encoding (minus one) and step (2) is Rice compression and index construction.

6.4 Compressing the Startpoints

Startpoint compression is addressed in this section. The main difference with the endpoints is that startpoints are not sorted, which means that a list needs to be compressed, rather than a set. Nevertheless, there is some slack in their choice. In particular, the startpoints are chosen in $\{0, \ldots, m_1 - 1\}$ in [26, 28] (with m_1 being the total number of chains generated during the offline phase). They are, as far as we know, chosen in the same way in modern implementations, and are therefore stored on $\lceil \log_2 m_1 \rceil$ bits rather than n.

One might try to somehow compress this further, but the possible gain is very small. Indeed the startpoints and endpoints may be seen as a choice of m couples in $\{0, \ldots, m_1 - 1\} \times A$, of which there are $\binom{Nm_1}{m}$. Therefore, using the same reasoning as in Section 6.1, one finds that the size of startpoints and endpoints together is lower-bounded by:

$$M_{\text{tot}}^{\text{opt}} = \log_2 \binom{Nm_1}{m}. \tag{6.10}$$

The total memory (the memory for the startpoints and the endpoints) is therefore given by $m\lceil \log_2 m_1 \rceil$ (that is, $\lceil \log_2 m_1 \rceil$ bits for each startpoint) plus the memory for the endpoints, lower-bounded by (and close to) eq. 6.2. This gives:

$$M_{\text{tot}} = m\lceil \log_2 m_1 \rceil + \log_2 \binom{N}{m}. \tag{6.11}$$

Consider the following practical example: a space size of $N = 2^{40}$, a number of chains $m = 2^{24}$, and $m_1 = 25 \times m$ chains generated during the offline phase. In this case, the theoretical lower bound for the total memory (from eq. (6.10)) is a mere 0.59% lower than the theoretical lower bound for startpoints and compressed endpoints (from eq. (6.11)).

This tells us that it is not possible to do significantly better than compressing the endpoints as explained in Section 6.3, and adding the startpoints on $\lceil \log_2 m_1 \rceil$ bits. Nevertheless, recall that the startpoints are in $\{0, \ldots, m_1 - 1\}$. Whether other choices allowing them to be compressed more exist is an open question.

6.5 Experiments and Comparison

Theoretical and practical experiments were carried out in order to compare the two techniques (prefix-suffix decomposition and compressed delta encoding), and evaluate the gain realized with respect to the naive implementation. The value of the number of blocks L for compressed delta encoding is also discussed in this section.

6.5.1 Choice of the L Parameter

Recall from Section 6.3 that the compressed delta encoding technique separates the endpoints into L blocks, and requires them to be indexed. The choice of the parameter L is a trade-off between memory and online time. Indeed, if L is too big, the index part is large and the memory increases too much. If it is too small, the online time is impacted in a noticeable way because the number of items that need to be decompressed on the fly increases.

Let $q = \frac{m}{L}$ be the average number of compressed endpoints per block. As mentioned in Section 6.3.2, the average overhead to recover a point in the endpoint list is $\frac{m}{2L} = \frac{q}{2}$. This overhead should be negligible compared to the work done by computing the online chain and verifying the false alarms at each step. However, note that this additional cost is highly implementation-dependant, since the hash function can be more or less expensive to compute.

Considering that the decoding procedure relies on other computations than cryptographic hash operations, it is difficult to have a reliable point of comparison other than measured time. The decoding procedure was implemented in C, and a decoding speed of about 18ns/entry was observed[3]. What is suggested here is that recovering an endpoint should take about the time of a cryptographic hash operation. This fixes $q \approx 2^8$ in this case. For instance with $N = 2^{40}$ and $m = 2^{24}$, this results in a negligible 0.006% increase in online time, and in a memory overhead of about 0.6%.

6.5.2 Measure of the Gain

Endpoint Compression

Figure 6.3 shows the ratios $M_{\text{ep}}^{\text{ps}}/M_{\text{ep}}^{\text{orig}}$ and $M_{\text{ep}}^{\text{cde}}/M_{\text{ep}}^{\text{orig}}$, which are a measure of the relative memory realized with respectively prefix-suffix decomposition and compressed delta encoding on the memory required to store the endpoints (the lower, the better). The two measures are contrasted with $M_{\text{ep}}^{\text{opt}}/M_{\text{ep}}^{\text{orig}}$, the optimal gain. The original method consists in storing the endpoints on $\lceil \log_2 N \rceil$ bits. In both cases, the optimal configurations are assumed, and for compressed delta encoding, a value of $L = \left\lfloor \frac{m}{2^8} \right\rfloor$ is considered. In this example, $N = 2^{40}$ is used, which corresponds to a medium-sized search space for today's hardware. Using a small or bigger searching space retains the same overall picture, with similar conclusions. The horizontal axis is the number of chains m.

One can observe that compressed delta encoding offers a significantly better compression than prefix-suffix decomposition, showing a 10 to 15% improvement in terms of memory. Figure 6.3 also clearly indicates that compressed delta encoding is very close to the lower bound.

Total Compression

As discussed in Section 6.4, startpoints can not be compressed significantly better than when encoded on $\lceil \log_2 m_1 \rceil$ bits. When one includes the startpoints in the memory (assuming $m_1 = 25 \times m$), the relative gain appears a bit smaller, as shown in Figure 6.4. It shows that compressed delta encoding is 5 to 7% better than prefix-suffix decomposition assuming optimal configuration. However, it is hard to tell if this number is very relevant because prefix-suffix decomposition might have been used with non-optimal parameters in past implementations.

[3]This experiment has been done on a laptop with an i5 Intel Processor, with the encoded differences in RAM. SHA-1 was used as a point of comparison for a hash function.

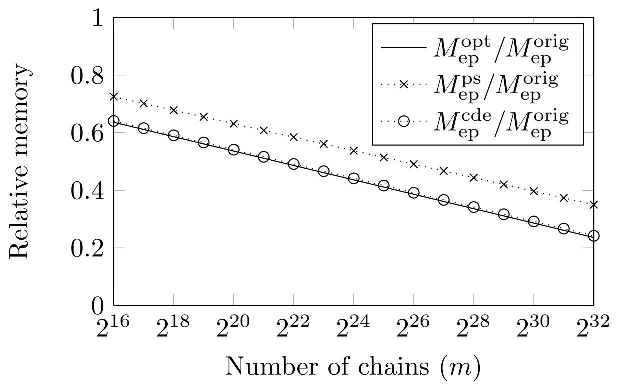


Figure 6.3: Endpoints relative memory for the two techniques ($N = 2^{40}$).

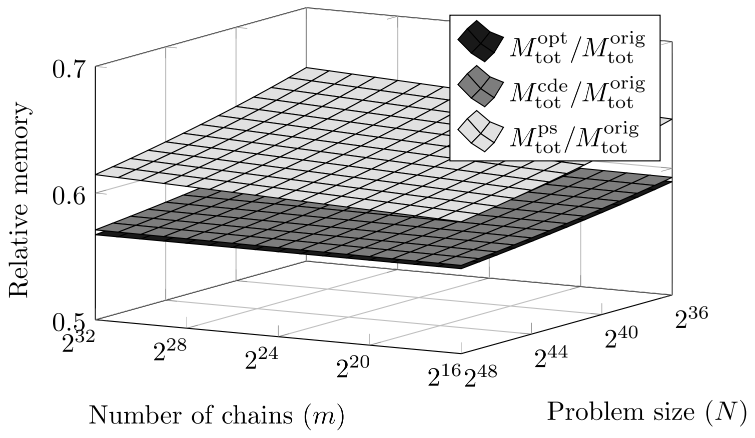


Figure 6.4: Total relative memory for the two techniques.

Table 6.1: Improvements in terms of memory and time for the two methods discussed in this chapter compared to the naive approach. Parameters are $m = 2^{24}$, $m_1 = 25 \times m$ and $N = 2^{40}$. Optimal configurations are assumed.

	M (fixed time)	T (fixed memory)
Naive approach	100.00%	100.00%
Prefix-suffix decomposition	63.44%	40.24%
Compressed delta encoding	58.44%	34.15%

Finally, Table 6.1 shows the gain brought by the two techniques (optimal parameters assumed) with respect to the naive approach, on a problem with parameters $m = 2^{24}$, $m_1 = 25 \times m$ and $N = 2^{40}$. The naive startpoint compression discussed in Section 6.4 is assumed for the two latter cases. The memory for the naive approach, the prefix-suffix decomposition, and compressed delta encoding are computed using respectively:

$$M_{\text{tot}}^{\text{orig}} = 2mn \tag{6.12}$$

$$M_{\text{tot}}^{\text{ps}} = m\lceil \log_2 m_1 \rceil + 2^p \lceil \log_2 m \rceil + ms \tag{6.13}$$

$$M_{\text{tot}}^{\text{cde}} = m\lceil \log_2 m_1 \rceil + mR_{k^{\text{opt}}} + L\left(\lceil \log_2 mR_{k^{\text{opt}}} \rceil + \lceil \log_2 m \rceil\right) \tag{6.14}$$

In any case, the impact of good storage optimizations on the overall memory is clearly illustrated, showing a reduced usage to 58.44% of the initial memory (a bout a 42% reduction). Recall that, since time-memory trade-offs follow the rule $T \propto N^2/M^2$ as described in Section 5.1, a reduction of 42% of the memory (as it is the case for instance with $m = 2^{24}$, $N = 2^{40}$ in Figure 6.4) is about the same as a speedup of 3 in time.

6.6 Conclusion

This chapter clearly illustrates the importance of implementation optimizations for storage in rainbow tables. The compressed delta encoding (almost) reaches optimality and improves the state of the art, prefix-suffix decomposition. These storage improvements represent a speedup of about 3 in time (with respect to the naive method), which is of great practical significance.

Chapter 7

Interleaving for Non-Uniform Input Distribution

Articles related to this chapter

[18] Gildas Avoine and Xavier Carpent. Interleaving Cryptanalytic Time-memory Trade-offs on Non-Uniform Distributions. *ACM Symposium on Information, Computer and Communication Security – ASIACCS'15.* (submitted)

7.1 Introduction

Security experts are often facing the problem of guessing secret values such as passwords. Performing an exhaustive search in the set of possible secrets is an ad-hoc approach commonly used. In practice, the searching time can usually be reduced (on average) by exploiting side information on the values to be guessed. For example, a password cracker checks the most commonly used passwords and their variants before launching an exhaustive search. This optimization is very effective, as described in [53, 142]. More generally, the distribution of the secrets can be exploited to reduce the average cryptanalysis time [136].

Nowadays, cryptanalytic time-memory trade-offs are used by most password crackers. Unfortunately, TMTOs do not behave well with non-uniform distributions of secrets because TMTOs, by construction, uniformly explore the set of considered secrets. Duplicating secrets in the search set artificially creates a non-uniform distribution but this approach does not make sense in practice due to the excessive waste of memory. Providing a solution would be very impactful in practice, though, not only for cracking passwords, but also for solving any problem that can be reduced to a chosen plaintext attack. For example, anonymization techniques based on hashing email addresses or MAC addresses are vulnerable targets for non-uniform TMTOs.

This chapter introduces a technique to make cryptanalytic time-memory trade-offs compliant with non-uniform distributions. More precisely, the approach consists in (i) dividing a set into subsets of close densities, and (ii) exploring the related time-memory trade-offs in an interleaved way (instead of sequentially) defined by a density-related metric. The technique significantly improves the cryptanalysis time when considering non-uniform distributions: it was employed to crack passwords and it is shown to be 16 times faster than state-of-the-art techniques [147] when considering real-life password distributions.

Cryptanalytic time-memory trade-offs are introduced in Section 5.1, and the same notations and conventions are followed throughout this chapter. This chapter focuses on clean maximal-sized rainbow tables. This choice was done because rainbow tables have been shown to be superior to Hellman tables [147, 121], and maximal tables have the best online performance (despite being

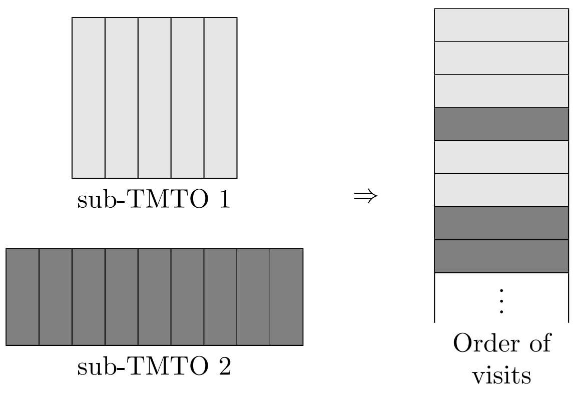


Figure 7.1: Intuitive illustration of the interleaved order of visit of two sub-TMTOs.

slower to precompute). However, the interleaving technique discussed here can be easily adapted to Hellman tables or non-maximal rainbow tables.

The interleaving technique is described and analyzed in Section 7.2), and the interleaving order discussed in Section 7.3. Section 7.4 explains the memory allocation, and Section 7.5 provides experimental results. Finally, the chapter is brought to a close in Section 7.6.

7.2 Interleaving

7.2.1 Description

The nature of rainbow tables dictates that each point of the input set is recovered using on average the same time. There is no bias in the coverage either: each point is covered a priori with the same probability.

In order to work efficiently with non-uniform input distribution, what we suggest is to divide the TMTO into several *sub-TMTOs*, one for each subdivision of the input set that can be considered as roughly uniform. For instance, if one wants to build a TMTO against passwords containing alphanumeric characters as well as special characters, two sub-TMTOs can be built on a partition of the input set: the "alphanumeric" password set and the "alphanumeric+special" password set (that is without passwords that are purely alphanumeric). This makes sense because the second set is considerably larger, despite most users having passwords from the first set. This disparity is not exploited in a regular TMTO over the whole input set. However, having two separate sub-TMTOs allows to dedicate a bigger share of the memory to the first one comparatively, thus accelerating the search where it matters the most on average.

Formally, let the input set A be partitioned into n input subsets $[A]_b$[1] of size $|[A]_b| = [N]_b$. Each subset has a probability p_b that the answer to the challenge in the online phase lies in $[A]_b$. The part of the trade-off dedicated to $[A]_b$ is named "sub-TMTO b". The memory is divided and a slice $[M]_b = \rho_b M$ is allocated for each sub-TMTO b, where M is the total memory available for the trade-off. Each sub-TMTO b is built on $[A]_b$ using a memory of $[M]_b$, exactly in the same way than a regular TMTO.

In order to search through the sub-TMTOs, one naive approach could be to search through each of them one by one, in decreasing order of probability. However, this technique is very slow when the point to recover ends up being in one of the last sub-TMTOs. Indeed, the search in the rainbow scheme is slower and slower when one moves towards the left of the TMTO. This drives the average search time down and makes the technique wasteful. A more efficient approach referred to as *interleaving* is discussed in the rest of this chapter. The idea of interleaving is to rather search through all sub-TMTOs at the same time, but to pick one sub-TMTO to search through at each step, effectively interleaving the order of columns visit. The intuition behind this may be found in Fig. 7.1.

[1]For notations that already exist for rainbow tables, the convention adopted throughout the article to avoid confusion is to surround them with brackets, as summarized in Table 7.1.

Table 7.1: Notations used in this chapter (specific to the interleaving technique).

Notation	Meaning	Notes
n	number of subsets (and sub-TMTOs)	
$[A]_b$	input subset	$\bigcup_{b=1}^{n}[A]_b = A$, $[A]_b \cap [A]_{b'} = \emptyset \ \ \forall b \neq b'$
$[N]_b$	input subset size	$[N]_b = \lvert[A]_b\rvert$, $\sum_{b=1}^{n}[N]_b = N$
p_b	intrinsic probability of subset $[A]_b$	$p_b = \Pr(x \in [A]_b \vert y = h(x))$, $\sum_{b=1}^{n} p_b = 1$
$[M]_b$	memory size for sub-TMTO b	$\sum_{b=1}^{n}[M]_b = M$
ρ_b	memory proportion for sub-TMTO b	$\rho_b M = [M]_b$, $\sum_{b=1}^{n} \rho_b = 1$
$[m]_b$	number of chains of the sub-TMTO b	$[m]_b = \frac{[M]_b}{2\lceil \log_2 N \rceil}$
$[t]_b$	length of chains of the sub-TMTO b	$[t]_b = \frac{2[N]_b}{[m]_b} - 1$
$\hat{t}$	total number of steps of the TMTO	$\hat{t} = \sum_{b=1}^{n}[t]_b$
$[C_i]_b$	cost for column i of sub-TMTO b	see Theorem 4

7.2.2 Analysis

Notations

In the analysis done in this chapter, the same number ℓ of tables is used in each sub-TMTO. It is also possible to use a different number of tables per sub-TMTO, but this results in a different probability of success for each of them.

A step is defined as being a search in one column for the ℓ tables of a given sub-TMTO. One could choose to define a step as being a search in a column for a single table of a given sub-TMTO, but doing so results in a negligible difference of performance at the cost of a more complicated analysis and implementation.

The notations used in this chapter are presented in Table 7.1.

Online Phase

Probability of Success The probability of success is the same in rainbow tables that use interleaving than in the undivided case, provided clean tables of maximal size are used.

Theorem 6. *The probability of success of a set of interleaved clean rainbow tables of maximal size is:*

$$P^* \approx 1 - e^{-2\ell}.$$

Proof. The probability of success is

$$P^* = \sum_{b=1}^{n} p_i P_i^*,$$

with P_i^* the probability of success of the sub-TMTO i. Since each sub-TMTO is a clean rainbow table of maximal size, we have that $P_i^* \approx 1 - e^{-2\ell}$, as computed in Result 1 of Chapter 5. The results follows from $\sum_{b=1}^{n} p_i = 1$. □

Average Time The average search time for interleaved rainbow tables is given in Theorem 7. An example of the average speedup realized is given in Section 7.5.

Theorem 7. *The average number of hash operations required in the online phase of a set of interleaved clean rainbow tables of maximal size, given a set of n input subset sizes $\{[N]_1, ..., [N]_n\}$, intrinsic probabilities $\{p_1, ..., p_n\}$, numbers of chains $\{[m]_1, ..., [m]_n\}$, and a vector $V = (V_1, ..., V_{\hat{t}})$*

representing the order of visits (i.e. V_k is the sub-TMTO chosen at step k) is:

$$T = \sum_{k=1}^{\hat{t}} p_{V_k} \left[1 - \left(1 - \frac{[m]_{V_k}}{[N]_{V_k}}\right)^{\ell}\right] \left[1 - \frac{[m]_{V_k}}{[N]_{V_k}}\right]^{(S_k-1)\ell} \sum_{i=1}^{k} \ell [C_{S_i}]_{V_i} + \left[\sum_{b=1}^{n} p_b \left(1 - \frac{[m]_b}{[N]_b}\right)^{[t]_b \ell}\right] \sum_{b=1}^{n} \sum_{s=1}^{[t]_b} \ell [C_s]_b, \tag{7.1}$$

with $\hat{t} = \sum_{b=1}^{n} [t]_b$ the total maximum number of steps[2]*, and S_k the number of steps for the sub-TMTO V_k after k steps in total, that is:*

$$S_k = \#\{i \leq k | V_i = V_k\}.$$

Proof. The formula is a relatively direct adaptation of the average time in the undivided case (Result 5 of Chapter 5) to the interleaved case. Let y be the given hash in the online phase, and $x \in A$ be the the preimage of y. The average cryptanalysis time for a TMTO is:

$$T = \sum_{k}^{\hat{t}} \Pr(\text{search succeeds at step } k) \times (\text{cost up to step } k) + \Pr(\text{search fails}) \times (\text{cost of a complete search})$$

V_k is the sub-TMTO chosen to visit at some step k. V_k has been visited $S_k - 1$ times until step k. The probability Pr(search succeeds at step k) is therefore:

$$p_{V_k} \left(1 - \left(1 - \frac{[m]_{V_k}}{[N]_{V_k}}\right)^{\ell}\right) \left(1 - \frac{[m]_{V_k}}{[N]_{V_k}}\right)^{(S_k-1)\ell}. \tag{7.2}$$

The first factor in (7.2) is simply $\Pr(x \in [A]_{V_k})$ (the search may not succeed otherwise). Then, for the search to succeed at step k (or step S_k within the sub-TMTO V_k), it must have failed up to now. This is the third factor in (7.2). The expression $\frac{[m]_{V_k}}{[N]_{V_k}}$ is the probability that x lies within any column of any table of the sub-TMTO V_k, provided that $x \in [A]_{V_k}$. The expression $\left(1 - \frac{[m]_{V_k}}{[N]_{V_k}}\right)^{(S_k-1)\ell}$ is then just the probability that x is not in any of the $(S_k - 1)\ell$ first columns visited, provided that $x \in [A]_{V_k}$. Finally, the second factor in (7.2) expresses the probability that x lies within one of the ℓ columns visited at this step[3], provided that $x \in [A]_{V_k}$ and that it has not been found up to now.

The value "cost up to step k" is the sum of the cost of each step up to the current one. For each step $i \leq k$, the sub-TMTO visited is V_i (and it is its S_i-th visit), and the associated cost is $[C_{S_i}]_{V_i}$ (see Result 4 of Chapter 5).

The probability Pr(search fails) is:

$$\sum_{b=1}^{n} p_b \left(1 - \frac{[m]_b}{[N]_b}\right)^{[t]_b \ell}. \tag{7.3}$$

A failed search means that x is not in any of the $[t]_b$ columns of the sub-TMTO b where b is such that $x \in [A]_b$. Since the subsets $[A]_i$ form a partition of A, the law of total probability gives (7.3). The factor $\left(1 - \frac{[m]_b}{[N]_b}\right)^{[t]_b \ell}$ is the probability that, x is not in any of the $[t]_b$ columns of the sub-TMTO b, given $x \in [A]_b$.

[2]$\hat{t}$ is used instead of t in order to avoid confusion with the number of columns of the undivided TMTO, which is a different number.

[3]Note that one would normally stop the search as soon as x is found rather than continuing with all ℓ tables of this step. This results in a more complex formula for the average time, and a negligible difference numerically.

Finally, the "cost of a complete search" is the sum of the cost of each step in all sub-TMTOs, that is $\sum_{b=1}^{n}\sum_{s=1}^{[t]_b}\ell[C_s]_b$. This expression could also be written $\sum_{k=1}^{\hat{t}}\ell[C_{S_k}]_{V_k}$, but the former expression is closer to its counterpart in Theorem 5 and also highlights that the cost of failure is independent of the order of visits V. □

Note that Theorem 7 for interleaved rainbow tables is a generalization of Result 5 of Chapter 5 for classical rainbow tables. In particular, if $n = 1$ and $V = (1, 1, ..., 1)$ with $|V| = t$, Theorem 7 gives the same equation as Result 5 of Chapter 5.

Worst-Case Time A drawback of the interleaving is that it has in general a worse worst-case time than an undivided TMTO. The worst-case in interleaved rainbow tables corresponds to the second factor of the last term in (7.1), that is:

$$\sum_{b=1}^{n}\sum_{s=1}^{[t]_b}\ell[C_s]_b. \tag{7.4}$$

Note that it is independent of the subset probabilities and the order of visits.

Offline Phase

Precalculation of an interleaved TMTO consists in precalculation of each sub-TMTO independently. Precalculation of a clean rainbow table set of m chains requires to build $m_1 = \alpha m$ chains, where α is a factor depending on how close to tables of maximal size the implementer wants to get (typical numbers for α are 20–50).

The precalculation cost is the same regardless of the order of visit (since this only regards the online phase), and asymptotically independent of the memory allocation for each sub-TMTO. In particular, it is the same as in the undivided case, as shown in Theorem 8.

Theorem 8. *The number of hash operations required in the precalculation phase of a set of interleaved clean rainbow tables, given a set of n input subset sizes $\{[N]_1, ..., [N]_n\}$, numbers of chains $\{[m]_1, ..., [m]_n\}$, and given α, the overhead factor for clean tables, is:*

$$P \approx 2\alpha\ell N.$$

Proof. The precalculation consists in computing, for each sub-TMTO b and for each of its ℓ tables, $[m_1]_b = \alpha[m]_b$ chains of $[t_b]$ points.

$$P = \sum_{b=1}^{n}\ell[m_1]_b[t]_b.$$

By replacing the definition of $[t]_b$ for clean tables, we get

$$P = \sum_{b=1}^{n}\ell[m_1]_b\left(\frac{2[N]_b}{[m]_b} - 1\right) \approx \sum_{b=1}^{n}\ell 2\alpha[N]_b.$$

The approximation $\left(\frac{2[N]_b}{[m]_b} - 1\right) \approx \frac{2[N]_b}{[m]_b}$ is good because typically, $[t]_b \gg 1$. For unusually small tables, the precalculation cost is slightly overestimated. The conclusion follows from $\sum_{b=1}^{n}[N]_b = N$ □

Storage

In this analysis, a naive storage consisting in storing both the starting and ending points on $\lceil\log_2 N\rceil$ bits is used. This explains why the number of chains $[m]_b$ is given as $\frac{[M]_b}{2\lceil\log_2 N\rceil}$ in Table 7.1.

Other options could be envisaged, such as storing the starting points on $\lceil\log_2[m_1]_b\rceil$ bits and the ending points on $\lceil\log_2[N]_b\rceil$ bits, or even better, using prefix-suffix decomposition or compressed delta encoding [17]. These storage techniques improve the memory efficiency and therefore implicitly the global efficiency of the trade-off. In fact, they are even more beneficial for interleaved sub-TMTOs than for an undivided TMTO, because sub-TMTOs operate on smaller subsets, and can therefore benefit from a more substantial reduction of the memory.

However, taking these into account makes both the analysis of interleaved sub-TMTOs and their comparison with an undivided TMTO quite a bit more complex. It is however strongly encouraged to take these storage improvements into consideration for practical implementations, and for figuring out the optimal memory allocation (as discussed in Section 7.4.2) for such practical implementations.

7.3 Order of Visit

7.3.1 Discussion

This section discusses the order of visit of the columns of the sub-TMTOs. Before every step during the search, a decision is made regarding in which sub-TMTO to search through during this step. This decision should be made easily and quickly, and the goal is to have an order of visit that minimizes the average search time.

What is suggested is that a metric is computed for each sub-TMTO b. This metric is defined as being the probability to find a solution in $[A]_b$ at the next step, divided by the average amount of work at the next step in sub-TMTO b (Definition 1).

Definition 1. *The metric associated to the k-th step of sub-TMTO b is:*

$$\eta(b,k) = \frac{\Pr(x \textit{ found at the } k\textit{-th step in sub-TMTO } b)}{\mathbb{E}[\textit{work for the } k\textit{-th step in sub-TMTO } b]},$$

with x, an answer in the online phase.

The sub-TMTO that should be visited is the one with the highest metric. This metric is quantified for the rainbow scheme case in Section 7.3.2.

7.3.2 Analysis

It has been shown in [28] that the probability for the preimage to be in any column is m/N. This probability is thus independent of the column visited. Moreover, it may be seen from Result 4 of Chapter 5 that the cost is monotonically increasing towards the left columns in a rainbow table. This means that it is always preferable to visit the rightmost column that is not yet visited first. Therefore, the metric is only computed for each sub-TMTO rather than for each column, since the choice of the column is implicitly the rightmost one[4].

Theorem 9. *The metric associated to the k-th step of sub-TMTO b is:*

$$\eta(b,k) = \frac{p_b \times \left(1 - \left(1 - \frac{[m]_b}{[N]_b}\right)^{\ell}\right)\left(1 - \frac{[m]_b}{[N]_b}\right)^{(k-1)\ell}}{\ell [C_k]_b}$$

Proof. The numerator in Definition 1 is the probability addressed in equation (7.2) in the proof of Theorem 7. The denominator, the expected work required at step k in sub-TMTO b is denoted $[C_k]_b$, and is computed as indicated in Result 4 of Chapter 5. Since the search is done in ℓ tables, the total work done at this step on average is $\ell[C_k]_b$. □

The Lemma 3 provided below helps demonstrating Theorem 10.

Lemma 3. *The metric $\eta(b,k)$ defined in Theorem 9 is a decreasing function of k.*

[4]Note that in rainbow tables with checkpoints [26], this is not entirely the case (columns where checkpoints are placed often have a slightly cheaper cost than the the column immediately to their right, for instance). Nevertheless, the search is performed from right to left as well in such tables (see [26]), and experiments show that the gain of reorganizing columns visit for taking this into account is extremely small.

Proof. The numerator of $\eta(b,k)$ is decreasing, because both p_b and $\left(1-\left(1-\frac{[m]_b}{[N]_b}\right)^{\ell}\right)$ are constant, and because $\left(1-\frac{[m]_b}{[N]_b}\right)^{(k-1)\ell}$ is decreasing (since $1-\frac{[m]_b}{[N]_b}<1$).

The denominator is an increasing function of k since the cost of a step is increasingly expensive towards the left of a rainbow table. □

Theorem 10. *The metric given in Theorem 9 is optimal, that is it minimizes T from Theorem 7 given a set of n input subset sizes $\{[N]_1,...,[N]_n\}$, intrinsic probabilities $\{p_1,...,p_n\}$ and numbers of chains $\{[m]_1,...,[m]_n\}$.*

Proof. For the sake of clarity, the following simplified notations are used in this proof:

$$\begin{aligned} f(b,k) &= p_b\left(1-\left(1-\frac{[m]_b}{[N]_b}\right)^{\ell}\right)\left(1-\frac{[m]_b}{[N]_b}\right)^{(k-1)\ell}, \\ g(b,k) &= \ell[C_k]_b. \end{aligned}$$

Let V be a vector describing an arbitrary order of visit. Let V^* be a vector describing the order of visit dictated by the metric given in Theorem 9. V is thus an arbitrary permutation of V^*. S_k (resp. S_k^*) is defined as in Theorem 7, that is how many times V_k (resp. V_k^*) has been visited up to step k included:

$$\begin{aligned} S_k &= \#\{i \le k | V_i = V_k\}, \\ S_k^* &= \#\{i \le k | V_i^* = V_k^*\}. \end{aligned}$$

Additionally, let $\sigma(b,k)$ be the position of the k-th apparition of the sub-TMTO b in V (and $\sigma^*(b,k)$ its V^* equivalent). In particular, the following identity binds these notations:

$$\sigma(V_i,S_i)=\sigma^*(V_i^*,S_i^*)=i \quad \forall\, 1\le i\le \hat{t}.$$

The goal is to minimize (7.1), that is:

$$\begin{aligned} T =& \\ &\sum_{k=1}^{\hat{t}} p_{V_k}\left[1-\left(1-\frac{[m]_{V_k}}{[N]_{V_k}}\right)^{\ell}\right]\left[1-\frac{[m]_{V_k}}{[N]_{V_k}}\right]^{(S_k-1)\ell}\sum_{i=1}^{k}\ell[C_{S_i}]_{V_i} \\ &+\left[\sum_{b=1}^{n} p_b\left(1-\frac{[m]_b}{[N]_b}\right)^{[t]_b\ell}\right]\sum_{b=1}^{n}\sum_{s=1}^{[t]_b}\ell[C_s]_b \\ &=\sum_{k=1}^{\hat{t}} f(V_k,S_k)\sum_{i=1}^{k} g(V_i,S_i) + constant. \end{aligned}$$

Note that the second term of this expression is constant, regardless of the choice for V. Therefore, in order to prove the optimality of the metric and thus the optimality of the choice of V^*, it suffices to show that $G \ge G^*$, with:

$$G=\sum_{k=1}^{\hat{t}} f(V_k,S_k)\sum_{i=1}^{k} g(V_i,S_i), \tag{7.5}$$

$$G^*=\sum_{k=1}^{\hat{t}} f(V_k^*,S_k^*)\sum_{i=1}^{k} g(V_i^*,S_i^*), \tag{7.6}$$

and with V any permutation of V^*. Equation (7.5) can be re-written such that sub-TMTOs are considered in consecutive order rather than considering them in the order of visit (this is a mere re-ordering of the terms in the sum):

$$G=\sum_{b=1}^{n}\sum_{k=1}^{[t]_b} f(b,k)\sum_{i=1}^{\sigma(b,k)} g(V_i,S_i),$$

and likewise for the sum in (7.6). It is now possible to factorize the difference $G - G^*$ as such:

$$G - G^* = \sum_{b=1}^{n} \sum_{k=1}^{[t]_b} f(b,k) \left[\sum_{i=1}^{\sigma(b,k)} g(V_i, S_i) - \sum_{i=1}^{\sigma^*(b,k)} g(V_i^*, S_i^*) \right]. \tag{7.7}$$

Some terms cancel each other out in the two sums of the bracketed factor in (7.7), i.e. terms that appear in both sums. Let $\Delta^+_{b,k}$ (resp. $\Delta^-_{b,k}$) be the set of positions in V (resp. V^*) that only appear in the left (resp. right) sum. Formally,

$$\Delta^+_{b,k} = \{j < \sigma(b,k) \mid \sigma^*(V_j, S_j) > \sigma^*(b,k)\},$$
$$\Delta^-_{b,k} = \{j < \sigma^*(b,k) \mid \sigma(V_j^*, S_j^*) > \sigma(b,k)\}.$$

This allows to rewrite the difference (7.7) as:

$$G - G^* = \sum_{b=1}^{n} \sum_{k=1}^{[t]_b} f(b,k) \left[\sum_{i \in \Delta^+_{b,k}} g(V_i, S_i) - \sum_{i \in \Delta^-_{b,k}} g(V_i^*, S_i^*) \right]. \tag{7.8}$$

By construction, the following implication holds between Δ^+ and Δ^-:

$$\sigma(b,k) \in \Delta^+_{b',k'} \iff \sigma^*(b',k') \in \Delta^-_{b,k}. \tag{7.9}$$

Indeed, we have:

$$\begin{aligned} &\sigma(b,k) \in \Delta^+_{b',k'} \\ &\iff \sigma(b,k) < \sigma(b',k') \wedge \sigma^*(V_{\sigma(b,k)}, S_{\sigma(b,k)}) > \sigma^*(b',k') \\ &\iff \sigma(b,k) < \sigma(b',k') \wedge \sigma^*(b,k) > \sigma^*(b',k') \end{aligned} \tag{7.10}$$

The first equivalence is the definition of $\Delta^+_{b,k}$, and the second comes from the fact that $V_{\sigma(b,k)} = b$ and $S_{\sigma(b,k)} = k$, by definition of V and S. Likewise,

$$\begin{aligned} &\sigma^*(b',k') \in \Delta^-_{b,k} \\ &\iff \sigma^*(b',k') < \sigma^*(b,k) \wedge \sigma(V^*_{\sigma^*(b',k')}, S^*_{\sigma^*(b',k')}) > \sigma(b,k) \\ &\iff \sigma^*(b',k') < \sigma^*(b,k) \wedge \sigma(b',k') > \sigma(b,k) \end{aligned} \tag{7.11}$$

The implication (7.9) comes from the equivalence between (7.10) and (7.11). As a particular case of (7.9), we have:

$$j \in \Delta^-_{b,k} \iff \sigma(b,k) \in \Delta^+_{V_j^*, S_j^*}.$$

This means that for each negative term $-f(b,k)g(V_j^*, S_j^*)$ in (7.8), there is also a positive counterpart $f(V_j^*, S_j^*)g(b,k)$. This allows to rewrite the difference (7.8) as a simple sum of opposed crossed terms:

$$G - G^* = \sum \Big[f(V_j^*, S_j^*)g(b,k) - f(b,k)g(V_j^*, S_j^*) \Big]. \tag{7.12}$$

We have that $\sigma^*(b,k) > j = \sigma^*(V_j^*, S_j^*)$, for all $j \in \Delta^-_{b,k}$, by definition of $\Delta^-_{b,k}$. Moreover, since the metric used to construct B^* is decreasing (see Lemma 3), we have that:

$$\eta(b,k) \geq \eta(b',k'),$$
$$f(b,k)g(b',k') \geq f(b',k')g(b,k),$$

for all b, k, b', k' such that $\sigma^*(b,k) < \sigma^*(b',k')$. Using this fact in (7.12) shows that each term of the sum is positive, and thus $G \geq G^*$. □

7.4 Input Set Partition and Memory Allocation

7.4.1 Input Set Partition

Partitioning the input set induces a time overhead for the online phase. Doing so is only worth it if the gain outweighs this overhead. The ratio $\frac{p_b}{[N]_b}$ represents the individual probability of occurrence for each point of the $[A]_b$ subset, and is intuitively a measure of the "density" of $[A]_b$. It makes sense to divide a set when it contains subsets of unbalanced densities. A TMTO covering a high-density subset should be devoted a higher memory and searched through more rapidly than average, and vice versa. Once the considered set is partitioned into subsets, one may compute the expected online time given using Theorem 7.

7.4.2 Memory Allocation

Given a partition $\{[N]_1, ..., [N]_n\}$ of the input set and their intrinsic probabilities $\{p_1, ..., p_n\}$, a memory size must be assigned to each subset. Given $[N]_b$, we have $[M]_b = \rho_b M$. The expression T given in Theorem 7 is not simple enough to determine analytically an optimal memory allocation. Instead, the memory allocation can be done solving an optimization problem that consists in minimizing T by changing the variables $\rho_1, ..., \rho_n$.

When the number of subsets n is small, the memory allocation can be found easily with a grid search. That is, T is evaluated at discretized values of the parameters $\rho_1, ..., \rho_n$ (with $\sum_{i=1}^{n} \rho_i = 1$), and the point where T is minimal is kept as the selected memory allocation.

This technique becomes quite costly when the number of subsets n is too large, or when the desired resolution of the discretization is too thin. Metaheuristic techniques of local search such as Hill Climbing [168] may be used instead to find the optimal memory allocation more efficiently.

7.5 Results

In this section, the interleaving technique is illustrated on password cracking. In order to determine the password distribution, two publicly-available datasets have been considered: RockYou and phpBB. The RockYou dataset originated from `rockyou.com`, a gaming website that lost 32.6 million unencrypted passwords in 2009. Among those passwords 14.3 million are unique. The phpBB dataset comes from `phpbb.com`, a forum that was attacked in 2009 due to a vulnerable third-party application. These datasets are for example used for dictionary attacks by the well-known password crackers Hashcat [7] and John the Ripper [160].

Tables 7.2 and 7.3 present some statistics on these datasets. Each cell of both tables represent the percentage of passwords that have the length indicated on the left, and that correspond to the character set indicated on the top. Such statistics can then be used to feed the parameters for the interleaving technique. This is illustrated below in the case of the RockYou dataset.

We decided to set A to the set of passwords of the special character set (96 characters) of length 7 or less, which corresponds to the same set covered in the "XP special" table of the Ophcrack software [148]. Likewise, we set the total memory to 8GB, which is about the memory used for this table. We set the number of tables to be $\ell = 4$, which means a probability of success of about 99.97%. With these settings, an undivided TMTO has an average cryptanalysis time of $T = 6.27 \times 10^9$ operations (obtained from Result 5 of Chapter 5).

We chose the following partition: $[A]_1$ is set to the passwords of length 7 (exactly) that contain at least one special character, and $[A]_2$ the rest of the passwords. This gives the following parameters for the RockYou dataset:

$$[N]_1 = 96^7 - 62^7 = 7.16 \times 10^{13} \qquad p_1 = 0.0143$$
$$[N]_2 = N - [N]_1 = 4.31 \times 10^{12} \qquad p_2 = 0.9857$$

The probabilities are taken from Table 7.2, and are adjusted such that the sum of probabilities up to length 7 is 1.

Length	Special	Lower	Upper	Digit	Alpha	Alnum
0	0.000	0.000	0.000	0.000	0.000	0.000
1	0.000	0.000	0.000	0.000	0.000	0.000
2	0.000	0.002	0.000	0.000	0.000	0.000
3	0.000	0.015	0.001	0.003	0.000	0.001
4	0.001	0.138	0.005	0.063	0.003	0.006
5	0.028	2.893	0.140	0.655	0.051	0.301
6	0.302	12.232	0.453	6.990	0.213	5.844
7	0.571	8.398	0.303	1.968	0.185	7.855
8	0.654	7.575	0.250	2.508	0.158	8.826
9	0.592	4.239	0.138	1.377	0.095	5.668
10	0.515	2.667	0.086	1.659	0.062	4.069
11	0.290	1.401	0.046	0.374	0.036	1.417
12	0.215	0.845	0.028	0.135	0.022	0.859
13	0.156	0.506	0.018	0.096	0.014	0.527
14	0.118	0.314	0.011	0.039	0.009	0.369
15	0.089	0.206	0.008	0.023	0.006	0.219
16	0.081	0.122	0.005	0.021	0.005	0.160
17	0.025	0.048	0.002	0.004	0.001	0.043
18	0.018	0.028	0.001	0.005	0.001	0.024
19	0.015	0.017	0.001	0.002	0.001	0.014

Table 7.2: Statistics for the Rockyou dataset (percent truncated to 10^{-3}).

Length	Special	Lower	Upper	Digit	Alpha	Alnum
0	0.000	0.000	0.000	0.000	0.000	0.000
1	0.001	0.030	0.001	0.033	0.000	0.000
2	0.001	0.062	0.004	0.017	0.000	0.004
3	0.005	0.376	0.013	0.156	0.005	0.023
4	0.013	1.766	0.031	1.174	0.031	0.144
5	0.038	4.160	0.064	0.694	0.121	0.592
6	0.213	14.787	0.218	5.193	0.450	6.355
7	0.329	9.523	0.132	1.272	0.365	6.062
8	0.505	10.765	0.161	2.451	0.786	12.523
9	0.259	4.252	0.062	0.568	0.181	3.766
10	0.123	2.474	0.034	0.327	0.119	2.210
11	0.069	1.058	0.009	0.086	0.049	0.812
12	0.040	0.520	0.005	0.056	0.022	0.410
13	0.022	0.217	0.003	0.014	0.010	0.160
14	0.017	0.099	0.002	0.010	0.006	0.079
15	0.010	0.046	0.000	0.002	0.001	0.034
16	0.004	0.024	0.000	0.003	0.000	0.018
17	0.004	0.007	0.000	0.000	0.000	0.002
18	0.002	0.006	0.000	0.001	0.000	0.002
19	0.003	0.000	0.000	0.000	0.000	0.000

Table 7.3: Statistics for the phpbb dataset (percent truncated to 10^{-3}).

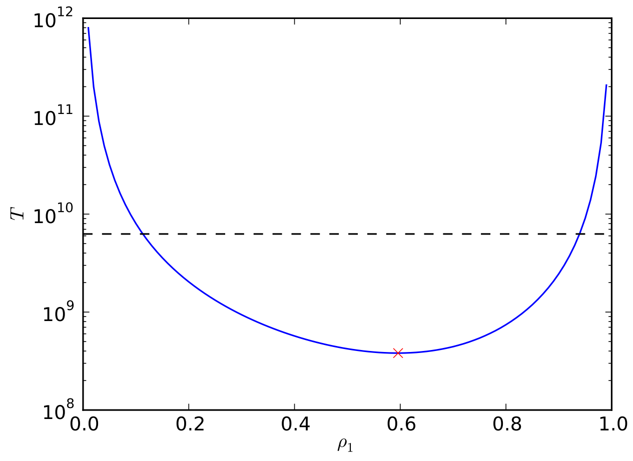


Figure 7.2: Memory allocation for the RockYou database: the solid line represents the average number of operations as a function of the proportion ρ_1 of the memory devoted to the first sub-TMTO. The crossmark is the optimal memory allocation, and the dashed line represents the cost of an undivided TMTO, about 16.45 times slower.

Figure 7.2 represents T according to ρ_1: the memory allocation is optimal when $\rho_1 = 0.5957$, with $T = 3.81 \times 10^8$ operations, which represents a speedup of about 16.45 with respect to the undivided case[5].

7.6 Conclusion

This chapter introduces a technique to improve the efficiency of cryptanalytic time-memory trade-offs when the considered distribution of secrets is non-uniform. It consists – during the precalculation phase – in dividing the input set into smaller subsets of unbalanced densities and applying a time-memory trade-off on every subset. As importantly, the contribution also consists of a method to interleave – during the attack phase – the exploration of the time-memory trade-offs. For that, a metric to select at each step the time-memory trade-off to be explored is introduced, and a proof of optimality is provided. The efficiency of the technique is practically demonstrated to crack 7-character passwords selected in a 96-character alphabet. Password length and alphabet size have been chosen in compliance with the tools currently used by security experts to crack LM Hash passwords. The password distributions used to evaluate the efficiency of the technique come from well-known websites whose password databases recently leaked. It has been shown that the time required to crack such passwords is divided by more than 16 when the technique introduced in this chapter is applied, compared to currently used time-memory trade-offs. The efficiency can be still better when considering distributions with a higher standard deviation. As far as we know, this is the first time an improvement on time-memory trade-off divides by more than two the cracking time since Hellman's seminal work.

[5]For instance, in terms of time elapsed on a laptop capable of performing 3×10^6 SHA-1 operations per second, and on a memory of 8GB, this corresponds to 34'50" (undivided case) reduced to 2'07" (interleaved case) on average.

Chapter 8

Conclusion

This thesis first explored two of the important challenges in RFID authentication: the design of extremely lightweight authentication protocols, and scalability issues in private authentication.

Ultralightweight protocols appeared in the advent of widespread usage of RFID technologies, as security and privacy became important research concerns. They aim to answer a need for secure and privacy-friendly authentication with very inexpensive tags. In this context, tags have very limited capabilities in terms of what they can compute and with how much time and energy at their disposal. In particular, both public-key cryptography and classical primitives of symmetric-key cryptography are deemed too expensive. The former is only used on very high-end tags while the latter is acceptable for intermediate tags. Instead, ultralightweight protocols rely on basic operations only, such as bitwise operations, modular addition, data-dependent rotations and other permutations, etc.

A significant part of this thesis has been dedicated to the cryptanalysis of ultralightweight protocols. Chapter 2 relates the state of the art, describes a couple of cryptanalyses, and discusses the typical weaknesses exploited by attacks on these protocols. Although commendable, the efforts to build a good ultralightweight protocol have so far virtually all failed due to weaknesses in their design. The quest for a perfectly secure and private ultralightweight protocol continues, but will hopefully use a slightly different, more constructive approach, with about 10 years of experience.

As of today, the best security/privacy versus cost compromise is probably to use ultralightweight primitives, such as PRESENT, on top of a classical challenge-response authentication protocol. As mentioned in Chapter 2, such primitives have security issues albeit less seriously so than ultralightweight protocols. Moreover, they are the focus of research of a large community and are getting more and more secure, as well as more and more lightweight. Then again, one might argue that using block ciphers or hash functions might be slightly excessive, as their security requirements may not all be needed for authentication purposes. With that in mind, it might still make sense to try and design an authentication protocol from scratch, being in the vein of existing ultralightweight protocols, or using different techniques (e.g. the HB family of protocols). Work in that direction will probably continue, but hopefully with more care, and with past mistakes in mind.

The second RFID topic discussed in this thesis is the scalability of privacy-friendly authentication protocols. Ensuring privacy in a protocol means that an adversary should not be able to easily guess the identity of the prover being authenticated. Therefore, exchanged information should notably have no correlation with the identity of the prover from the point of view of the attacker. This constraint however hinders the ability of a legitimate verifier to identify (and authenticate) a prover efficiently.

Partial solutions to the problem have been created, using very inventive mechanisms. These are analyzed in depth in Chapter 3. Among other ideas, the two main categories are protocols that use (partially) shared secrets, and those that use hash chains. The former type of protocols let provers share (part of) their secrets with each other, which alleviates the identification procedure on the verifier side. The main drawback is that it also gives way to dangerous compromising attacks, where an attacker that has acquired the secrets of a prover gains an advantage at identifying other provers. Although these protocols are quite efficient and are mostly secure against "weak" adversaries, the danger of compromising attacks is too great for most realistic settings. The protocols in the second notable category, those based on hash chains, use dynamic secrets in the provers, along with ways for the verifier to use former secrets of provers to accelerate the search. Beside not having the issue of compromising attacks, these protocols have the additional advantage of being able to provide forward-secrecy, a strong notion of privacy. They are, on the other hand slightly less efficient and have other miscellaneous minor issues.

On paper, the best solution to this issue is probably the OSK/AO protocol, a variant of the OSK protocol that use a cryptanalytic time-memory trade-off to accelerate the search. As discussed in Chapter 4 however, the AO variant only makes sense in certain settings, such as with mobile readers with limited memory. The full storage variant is on the other hand extremely efficient, and quite realistic, when sufficient memory is available. Other protocols (such as O-RAP) present other characteristics which might make more sense in some scenarios, but OSK (in either its AO or full-storage variant) seems to be the overall winner as of today.

Although some existing solutions are already acceptable in many settings, there might be protocols with a better privacy/complexity trade-off. This could be achieved through refining existing protocols, or using new techniques using shared secrets, hash chains, or any other ways of reducing the complexity level. In all cases analyzed in Chapter 3, protocols also trade away some aspect of their security or usability to lower the privacy/complexity trade-off curve (for instance, shared secrets schemes introduce the compromising attacks, OSK is technically descynchronizable, O-RAP has a high worst-case complexity, YA-TRAP has additional system assumptions, etc.). Finally, it might be extremely valuable to know theoretical bounds on this privacy/complexity trade-off (i.e. minimum complexity achievable for a given level of privacy), but it seems tricky to unify these protocols (and their specific security and usability characteristics) in a model that is both fair and realistic.

The second part of this thesis focused on improvements on cryptanalytic time-memory trade-offs. Although they are at first sight quite unrelated to the two previous research topics, they present an interest in the improvement of OSK/AO, and might be of use in other similar protocols.

Cryptanalytic time-memory trade-offs are a tool to perform efficient brute-force on a one-way function. They consist in a precomputation phase, and an online phase in which the output of a one-way function is provided, and its corresponding preimage is to be found. In the former, chains of hashes are computed and only the start and the end of each chain is kept in memory. In the latter, a chain is built from the given image, hopefully matching a precomputed chain. The principal use case for cryptanalytic time-memory trade-offs is the retrieval of passwords from their stored hashes. There are other cryptanalytic applications, and even constructive ones such as in the OSK/AO protocol. They make sense in three scenarios: (1) the precomputation is carried out by a more powerful entity than the online phase, (2) the online phase is carried out many times over distinct sessions, or (3) the window of opportunity for the attack is small, but the preparation time may be long.

The specific variant of cryptanalytic time-memory trade-offs that is considered the most efficient today is the rainbow table, and it was the focus of the research done in this thesis. Among the approaches at improving its performance that were explored, three showed interesting results: the fingerprints, improved storage, and interleaving.

The fingerprint (see Chapter 5) is an information that is used to represent a chain, along with its startpoint. It is essentially a generalized model built on earlier improvements of rainbow tables, namely the checkpoints, and the endpoint truncation. This model may help to think about chains differently and might spark new ideas, but most importantly, it allows the compound analysis of checkpoints and endpoint truncation. This made possible the systematic determination of optimal configurations. In such optimal configurations, rainbow tables with fingerprints may achieve a speedup of about two with respect to the plain version.

Storage in cryptanalytic time-memory trade-offs is very important, as more memory available or more efficient storage means faster online phase (the time in the online phase being inversely proportional to the square of the memory available). Up to now, endpoint storage was done using a technique called prefix-suffix decomposition, of which configurations were found empirically. Chapter 6 discusses a bound on storage and presents compressed delta encoding, a technique to reach this bound.

While symmetric keys are generated randomly according to a uniform distribution, passwords chosen by users are in practice far from being random, as confirmed by recent leakage of databases.

Unfortunately, the technique used to build classical rainbow tables is not able to capitalize on this bias. Chapter 7 introduces an efficient construction that consists in partitioning the search set into subsets of close densities, and a strategy to explore the TMTOs associated to the subsets based on an interleaved traversal. This approach results in a significant improvement compared to currently used TMTOs. On a typical searching space, interleaving presents a speedup of about 16 with respect to the monolithic approach.

Fingerprints and storage compression are both generic techniques to improve the efficiency of rainbow tables. The efficiency is however quite far from the bound stated in [38], by Barkan, Biham, and Shamir. Chances are however that this bound will never be tight, due to relatively high numerical factors in it. It is thus hard to tell how far these improvements bring rainbow tables from theoretically maximal efficiency. Although techniques such as interleaving can bring an arbitrarily high speedup, it does not contracdict the bound because the assumptions are different: interleaving takes advantage of a bias in the input distribution probability, whereas in [38], the input set is assumed to be uniformly distributed.

Cryptanalytic time-memory trade-offs are an important element of applied cryptography, and there seems to be room for improvement. These improvements may come in the form of algorithmic enhancements, or practical implementations. Areas that deserve scrutiny are the online phase of course, but also the precomputation (which is in some cases the bottleneck, and is somewhat poorly studied), or some specific relaxations or variations of the problem. Although not explored in this thesis, practical implementation improvements also matter a lot. Nowadays, cryptanalytic time-memory trade-offs are often stored on RAM, which limits the memory that is available. Alternatively, one may use hard drives or flash memory, but it has drawbacks (essentially due to the overhead of data transfer). Clever memory management could allow bigger memories and thus much faster research (or bigger input space). Computation done on graphic cards has also proved quite potent, but has constraints (GPU's notably have small memories and are highly parallelized). Finally, cryptanalytic time-memory trade-offs are mostly implemented on general-purpose PC's, but could benefit from specific hardware or configurations with a large amount of fast memory. Improving the efficiency of time-memory trade-offs is important at the fundamental research standpoint, for forensics applications, as well as for promoting stronger security standards.

Bibliography

[1] Mete Akgün, M. Ufuk Caglayan, and Emin Anarim. Secure RFID Authentication with Efficient Key-lookup. In *Proceedings of the 28th IEEE conference on Global telecommunications*, GLOBECOM'09, pages 4777–4784, Piscataway, USA, 2009. IEEE Press.

[2] Mete Akgün and M. Ufuk Çağlayan. On the security of recently proposed RFID protocols. Cryptology ePrint Archive, Report 2013/820, 2013.

[3] Mahdi R. Alagheband and Mohammad R. Aref. Simulation-based traceability analysis of RFID authentication protocols. *Wireless Personal Communications*, December 2013.

[4] Basel Alomair, Andrew Clark, Jorge Cuellar, and Radha Poovendran. Scalable RFID Systems: a Privacy-Preserving Protocol with Constant-Time Identification. In *the 40th Annual IEEE/IFIP International Conference on Dependable Systems and Networks – DSN'10*, Chicago, Illinois, USA, June 2010. IEEE, IEEE Computer Society.

[5] Alessandro Armando, David Basin, Yohan Boichut, Yannick Chevalier, Luca Compagna, Jorge Cuéllar, Paul Hankes Drielsma, Pierre-Cyrille Héam, Olga Kouchnarenko, Jacopo Mantovani, et al. The avispa tool for the automated validation of internet security protocols and applications. In *Computer Aided Verification*, pages 281–285. Springer, 2005.

[6] Frederik Armknecht, Matthias Hamann, and Vasily Mikhalev. Lightweight authentication protocols on ultra-lightweight RFIDs – myths and facts. In *Workshop on RFID Security – RFIDSec'14*, Oxford, UK, July 2014.

[7] atom. The Hashcat password cracker. `http://hashcat.net/hashcat/`, 2014.

[8] Jean-Philippe Aumasson, Luca Henzen, Willi Meier, and María Naya-Plasencia. Quark: A lightweight hash. *Journal of cryptology*, 26(2):313–339, 2013.

[9] Gildas Avoine. Adversary Model for Radio Frequency Identification. Technical Report LASEC-REPORT-2005-001, Swiss Federal Institute of Technology (EPFL), Security and Cryptography Laboratory (LASEC), Lausanne, Switzerland, September 2005.

[10] Gildas Avoine. *Cryptography in Radio Frequency Identification and Fair Exchange Protocols*. PhD thesis, EPFL, Lausanne, Switzerland, December 2005.

[11] Gildas Avoine. RFID lounge. `http://www.avoine.net/rfid/`, January 2011.

[12] Gildas Avoine, Muhammed Ali Bingöl, Xavier Carpent, and Suleyman Kardas. Deploying OSK on low-resource mobile devices. In *Workshop on RFID Security – RFIDSec'13*, Graz, Austria, July 2013.

[13] Gildas Avoine, Muhammed Ali Bingöl, Xavier Carpent, and Sıddıka Berna Örs Yalçın. Privacy-friendly authentication in RFID systems: On sub-linear protocols based on symmetric-key cryptography. *IEEE Transactions on Mobile Computing*, 12(10):2037–2049, October 2013.

[14] Gildas Avoine, Adrien Bourgeois, and Xavier Carpent. Analysis of rainbow tables with fingerprints. In *Financial Cryptography and Data Security – FC'15 (submitted)*, 2015.

[15] Gildas Avoine, Levente Buttyán, Tamás Holczer, and István Vajda. Group-based private authentication. In *IEEE International Workshop on Trust, Security, and Privacy for Ubiquitous Computing – TSPUC*, pages 1–6, Helsinki, Finland, June 2007. IEEE Computer Society.

[16] Gildas Avoine and Xavier Carpent. Yet another ultralightweight authentication protocol that is broken. In *Workshop on RFID Security – RFIDSec'12*, Nijmegen, Netherlands, June 2012.

[17] Gildas Avoine and Xavier Carpent. Optimal storage for rainbow tables. In Sukwoo Kim and Seok-Yeol Kang, editors, *International Conference on Information Security and Cryptology – ICISC 2013*, Seoul, South Korea, November 2013.

[18] Gildas Avoine and Xavier Carpent. Interleaving cryptanalytic time-memory trade-offs on non-uniform distributions. In *ACM Symposium on Information, Computer and Communication Security – ASIACCS'15 (submitted)*, Singapore, 2015.

[19] Gildas Avoine, Xavier Carpent, and Julio C. Hernandez-Castro. Pitfalls in ultralightweight authentication protocol designs. *IEEE Transactions on Mobile Computing (submitted)*, 2014.

[20] Gildas Avoine, Xavier Carpent, and Benjamin Martin. Strong Authentication and Strong Integrity (SASI) is not that Strong. In S.B. Ors Yalcin, editor, *Workshop on RFID Security – RFIDSec'10*, volume 6370 of *Lecture Notes in Computer Science*, pages 50–64, Istanbul, Turkey, June 2010. Springer.

[21] Gildas Avoine, Xavier Carpent, and Benjamin Martin. Privacy-friendly synchronized ultralightweight authentication protocols in the storm. *Journal of Network and Computer Applications*, December 2011.

[22] Gildas Avoine, Iwen Coisel, and Tania Martin. Time Measurement Threatens Privacy-Friendly RFID Authentication Protocols. In S.B. Ors Yalcin, editor, *Workshop on RFID Security – RFIDSec'10*, volume 6370 of *Lecture Notes in Computer Science*, pages 138–157, Istanbul, Turkey, June 2010. Springer.

[23] Gildas Avoine, Iwen Coisel, and Tania Martin. Untraceability model for RFID. *IEEE Transactions on Mobile Computing*, PrePrint, November 2013.

[24] Gildas Avoine, Iwen Coisel, and Tania Martin. Untraceability model for RFID. *IEEE Transactions on Mobile Computing*, PrePrint, November 2014.

[25] Gildas Avoine, Etienne Dysli, and Philippe Oechslin. Reducing Time Complexity in RFID Systems. In Bart Preneel and Stafford Tavares, editors, *Selected Areas in Cryptography – SAC 2005*, volume 3897 of *Lecture Notes in Computer Science*, pages 291–306, Kingston, Canada, August 2005. Springer.

[26] Gildas Avoine, Pascal Junod, and Philippe Oechslin. Time-memory trade-offs: False alarm detection using checkpoints. In *Progress in Cryptology – Indocrypt 2005*, volume 3797 of *Lecture Notes in Computer Science*, pages 183–196, Bangalore, India, December 2005. Cryptology Research Society of India, Springer.

[27] Gildas Avoine, Pascal Junod, and Philippe Oechslin. Characterization and Improvement of Time-Memory Trade-Off Based on Perfect Tables. *ACM Trans. Inf. Syst. Secur.*, 11:17:1–17:22, July 2008.

[28] Gildas Avoine, Pascal Junod, and Philippe Oechslin. Characterization and improvement of time-memory trade-off based on perfect tables. *ACM Trans. Inf. Syst. Secur.*, 11(4):1–22, July 2008.

[29] Gildas Avoine, Benjamin Martin, and Tania Martin. Tree-Based RFID Authentication Protocols Are Definitively Not Privacy-Friendly. In S.B. Ors Yalcin, editor, *Workshop on RFID Security – RFIDSec'10*, volume 6370 of *Lecture Notes in Computer Science*, pages 103–122, Istanbul, Turkey, June 2010. Springer.

[30] Gildas Avoine and Philippe Oechslin. A Scalable and Provably Secure Hash Based RFID Protocol. In *International Workshop on Pervasive Computing and Communication Security – PerSec 2005*, pages 110–114, Kauai Island, Hawaii, USA, March 2005. IEEE, IEEE Computer Society.

[31] Steve Babbage. A space/time tradeoff in exhaustive search attacks on stream ciphers. In *European Convention on Security and Detection*, volume 408, 1995.

[32] Nasour Bagheri, Praveen Gauravaram, Masoumeh Safkhani, and Somitra Kumar Sanadhya. The resistance to intermittent position trace attacks and desynchronization attacks (RIPTA-DA) protocol is not RIPTA-DA. In *Workshop on RFID Security – RFIDSec'13*, Graz, Austria, July 2013.

[33] Nasour Bagheri, Masoumeh Safkhani, Majid Naderi, and Somitra Kumar Sanadhya. Security Analysis of $LMAP^{++}$, an RFID Authentication Protocol. Cryptology ePrint Archive, Report 2011/193, 2011.

[34] Valentina Banciu, Simon Hoerder, and Dan Page. Lightweight primitive, feather-weight security? a cryptanalytic knock-out. (preliminary results). Cryptology ePrint Archive, Report 2013/421, 2013.

[35] Mihály Bárász, Balázs Boros, Péter Ligeti, Krisztina Lója, and Dániel Nagy. Breaking LMAP. In *Conference on RFID Security*, Malaga, Spain, July 2007.

[36] Mihály Bárász, Balázs Boros, Péter Ligeti, Krisztina Lója, and Dániel Nagy. Passive Attack Against the M2AP Mutual Authentication Protocol for RFID Tags. In *First International EURASIP Workshop on RFID Technology*, Vienna, Austria, September 2007.

[37] Gregory V. Bard, Nicolas T. Courtois, Jorge Jr. Nakahara, Pouyan Sepehrdad, and Bingsheng Zhang. Algebraic, AIDA/Cube and side channel analysis of KATAN family of block ciphers. In Guang Gong and Kishan Chand Gupta, editors, *Proceedings of the 11th International Conference on Cryptology in India – Indocrypt 2010*, volume 6498 of *Lecture Notes in Computer Science*, pages 176–196, Hyderabad, India, December 2010. Springer.

[38] Elad Barkan, Eli Biham, and Adi Shamir. Rigorous bounds on cryptanalytic time/memory tradeoffs. In Cynthia Dwork, editor, *Advances in Cryptology – CRYPTO'06*, Lecture Notes in Computer Science, Santa Barbara, California, USA, August 2006. Springer.

[39] David F. Barrero, Julio César Hernández-Castro, Pedro Peris-Lopez, David Camacho, and María D. R-Moreno. A genetic tango attack against the David-Prasad RFID ultra-lightweight authentication protocol. *Expert Systems*, September 2012.

[40] Ramzi Bassil, Wissam El-Beaino, Wassim Itani, Ayman Kayssi, and Ali Chehab. PUMAP: A PUF-based ultra-lightweight mutual-authentication RFID protocol. *International Journal of RFID Security and Cryptography*, 1(1):58–66, March 2012.

[41] Côme Berbain, Olivier Billet, Jonathan Etrog, and Henri Gilbert. An Efficient Forward Private RFID Protocol. In Ehab Al-Shaer, Somesh Jha, and Angelos D. Keromytis, editors, *Conference on Computer and Communications Security – ACM CCS'09*, pages 43–53, Chicago, Illinois, USA, November 2009. ACM, ACM Press.

[42] Michael Beye and Thijs Veugen. Improved anonymity for key-trees. Cryptology ePrint Archive, Report 2011/395, 2011.

[43] Zeeshan Bilal and Keith Martin. Ultra-lightweight mutual authentication protocols: Weaknesses and countermeasures. In *Eighth International Conference on Availability, Reliability and Security – ARES 2013*, pages 304–309. IEEE, September 2013.

[44] Olivier Billet, Jonathan Etrog, and Henri Gilbert. Lightweight Privacy Preserving Authentication for RFID Using a Stream Cipher. In Seokhie Hong and Tetsu Iwata, editors, *Fast Software Encryption – FSE'10*, volume 6147 of *Lecture Notes in Computer Science*, pages 55–74, Seoul, Korea, February 2010. Springer.

[45] Muhammed Ali Bingöl. Security analysis of RFID authentication protocols based on symmetric cryptography and implementation of a forward private scheme. Master's thesis, Istanbul Technical University, Istanbul, Turkey, January 2012.

[46] Alex Biryukov, Sourav Mukhopadhyay, and Palash Sarkar. Improved time-memory tradeoffs with multiple data. In Bart Preneel and Stafford Tavares, editors, *Selected Areas in Cryptography – SAC 2005*, volume 3897 of *Lecture Notes in Computer Science*, pages 110–127, Kingston, Canada, August 2005. Springer.

[47] Alex Biryukov and Adi Shamir. Cryptanalytic time/memory/data tradeoffs for stream ciphers. In Colin Boyd, editor, *Advances in Cryptology – ASIACRYPT'01*, volume 2248 of *Lecture Notes in Computer Science*, pages 1–13, Gold Coast, Australia, December 2001. Springer.

[48] Alex Biryukov, Adi Shamir, and David Wagner. Real time cryptanalysis of A5/1 on a PC. In Bruce Schneier, editor, *Fast Software Encryption – FSE'00*, volume 1978 of *Lecture Notes in Computer Science*, pages 1–18, New York, USA, April 2000. Springer.

[49] Burton Howard Bloom. Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, 13(7):422–426, 1970.

[50] Andrey Bogdanov, Miroslav Knežević, Gregor Leander, Deniz Toz, Kerem Varıcı, and Ingrid Verbauwhede. Spongent: A lightweight hash function. In *Cryptographic Hardware and Embedded Systems–CHES 2011*, pages 312–325. Springer, 2011.

[51] Andrey Bogdanov, Lars R Knudsen, Gregor Leander, Christof Paar, Axel Poschmann, Matthew JB Robshaw, Yannick Seurin, and Charlotte Vikkelsoe. Present: An ultra-lightweight block cipher. In *Cryptographic Hardware and Embedded Systems-CHES 2007*, pages 450–466. Springer, 2007.

[52] Andrey Bogdanov, Gregor Leander, Christof Paar, Axel Poschmann, Matthew Robshaw, and Yannick Seurin. Hash Functions and RFID Tags : Mind The Gap. In Elisabeth Oswald and Pankaj Rohatgi, editors, *Proceedings of the 10th International Workshop Cryptographic Hardware and Embedded Systems – CHES 2008*, volume 5154 of *Lecture Notes in Computer Science*, pages 283–299, Washington, DC, USA, August 2008. Springer.

[53] Joseph Bonneau. The Science of Guessing: Analyzing an Anonymized Corpus of 70 Million Passwords. In *IEEE Symposium on Security and Privacy, S&P 2012*, San Francisco, CA, USA, May 2012. IEEE Computer Society.

[54] Julien Bringer, Hervé Chabanne, and Dottax Emmanuelle. HB^{++}: a Lightweight Authentication Protocol Secure against Some Attacks. In *IEEE International Conference on Pervasive Services, Workshop on Security, Privacy and Trust in Pervasive and Ubiquitous Computing – SecPerU 2006*, Lyon, France, June 2006. IEEE, IEEE Computer Society.

[55] Julien Bringer, Hervé Chabanne, and Thomas Icart. Cryptanalysis of EC-RAC, a RFID identification protocol. In Matthew K. Franklin, Lucas Chi Kwong Hui, and Duncan S. Wong, editors, *7th International Conference on Cryptology And Network Security – CANS'08*, volume 5339 of *Lecture Notes in Computer Science*, pages 149–161, Hong Kong, China, December 2008. Springer.

[56] Andrei Broder and Michael Mitzenmacher. Using multiple hash functions to improve IP lookups. In *Procedings of the twentieth Annual Joint Conference of the IEEE Computer and Communications Societies – INFOCOM 2001*, volume 3, pages 1454–1463. IEEE, 2001.

[57] Mike Burmester, Tri van Le, and Breno de Medeiros. Provably Secure Ubiquitous Systems: Universally Composable RFID Authentication Protocols. In *Conference on Security and Privacy for Emerging Areas in Communication Networks – SecureComm 2006*, pages 1–10, Baltimore, Maryland, USA, August–September 2006. IEEE, IEEE Computer Society.

[58] Mike Burmester, Tri van Le, and Breno de Medeiros. Universally Composable RFID Identification and Authentication Protocols. *ACM Transactions on Information and System Security – TISSEC'09*, 12(4):21:1–21:33, 2009.

[59] Levente Buttyán, Tamás Holczer, and István Vajda. Optimal Key-Trees for Tree-Based Private Authentication. In George Danezis and Philippe Golle, editors, *Workshop on Privacy Enhancing Technologies – PET 2006*, volume 4258 of *Lecture Notes in Computer Science*, pages 332–350, Cambridge, United Kingdom, June 2006. Springer.

[60] Sébastien Canard and Iwen Coisel. Data Synchronization in Privacy-Preserving RFID Authentication Schemes. In *Workshop on RFID Security – RFIDSec'08*, Budapest, Hungary, July 2008.

[61] Mario Cardullo and William L. Parks. Transponder apparatus and system. US Patent 3713148, May 1970.

[62] Christy Chatmon, Tri van Le, and Mike Burmester. Secure Anonymous RFID Authentication Protocols. Technical Report TR-060112, Florida State University, Department of Computer Science, Tallahassee, Florida, USA, 2006.

[63] Jung Hee Cheon, Jeongdae Hong, and Gene Tsudik. Reducing RFID Reader Load with the Meet-in-the-Middle Strategy. Cryptology ePrint Archive, Report 2009/092, 2009.

[64] Hung-Yu Chien. SASI: A New Ultralightweight RFID Authentication Protocol Providing Strong Authentication and Strong Integrity. *IEEE Transactions on Dependable and Secure Computing*, 4(4):337–340, December 2007.

[65] Iwen Coisel and Tania Martin. Untangling RFID privacy models. *Journal of Computer Networks and Communications*, July 2012.

[66] HID Global Corporation. HSPD-12 & FIPS 201 PIV II: How Government Standards Affect Physical Access Control. `http://www.hidglobal.com/sites/hidglobal.com/files/hid-how-gov-stanards-affect-physical-access-control-wp-en.pdf`, 2007.

[67] Mathieu David and Neeli R. Prasad. Providing strong security and high privacy in low-cost RFID networks. In Ozgur Akan, Paolo Bellavista, Jiannong Cao, Falko Dressler, Domenico Ferrari, Mario Gerla, Hisashi Kobayashi, Sergio Palazzo, Sartaj Sahni, Xuemin (Sherman) Shen, Mircea Stan, Jia Xiaohua, Albert Zomaya, Geoffrey Coulson, Andreas U. Schmidt, and Shiguo Lian, editors, *Security and Privacy in Mobile Information and Communication Systems*, volume 17 of *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, pages 172–179, Turin, Italy, June 2009. Springer.

[68] Anindya De, Luca Trevisan, and Madhur Tulsiani. Time space tradeoffs for attacks against one-way functions and PRGs. In Tal Rabin, editor, *Advances in Cryptology – CRYPTO'10*, volume 6223 of *Lecture Notes in Computer Science*, pages 649–665, Santa Barbara, California, USA, August 2010. Springer.

[69] Christophe De Canniere, Orr Dunkelman, and Miroslav Knežević. Katan and ktantana family of small and efficient hardware-oriented block ciphers. In *Cryptographic Hardware and Embedded Systems-CHES 2009*, pages 272–288. Springer, 2009.

[70] Dorothy Denning. *Cryptography and Data Security*, page 100. Addison-Wesley, Boston, Massachusetts, USA, 1982.

[71] Whitfield Diffie and Martin Hellman. Special Feature Exhaustive Cryptanalysis of the NBS Data Encryption Standard. *Computer*, 10(6):74–84, 1977.

[72] Tassos Dimitriou. A Lightweight RFID Protocol to protect against Traceability and Cloning attacks. In *Conference on Security and Privacy for Emerging Areas in Communication Networks – SecureComm 2005*, pages 59–66, Athens, Greece, September 2005. IEEE, IEEE Computer Society.

[73] Danny Dolev and Andrew C. Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, 29(2):198–207, 1983.

[74] Dang Nguyen Duc and Kwangjo Kim. Securing HB+ against GRS Man-in-the-Middle Attack. In *Institute of Electronics, Information and Communication Engineers, Symposium on Cryptography and Information Security*, pages 23–26, 2007.

[75] Pierre Dusart and Sinaly Traoré. Lightweight authentication protocol for low-cost RFID tags. In Lorenzo Cavallaro and Dieter Gollmann, editors, *Information Security Theory and Practice. Security of Mobile and Cyber-Physical Systems – WISTP 2013*, volume 7886 of *Lecture Notes in Computer Science*, pages 129–144, Heraklion, Greece, May 2013. Springer.

[76] Aras Eghdamian and Azman Samsudin. A secure protocol for ultralightweight radio frequency identification (RFID) tags. In Azizah Abd Manaf, Akram Zeki, Mazdak Zamani, Suriayati Chuprat, and Eyas El-Qawasmeh, editors, *Informatics Engineering and Information Science – ICIEIS 2011*, volume 251 of *Communications in Computer and Information Science*, pages 200–213, Kuala Lumpur, Malaysia, November 2011. Springer.

[77] EPCglobal. EPC Radio-Frequency Identity Protocols Class-1 Generation-2 UHF RFID Protocol for Communications at 860MHz-960MHz. `http://www.gs1.org/epcglobal`.

[78] Xinxin Fan, Guang Gong, Daniel W. Engels, and Eric M. Smith. A lightweight privacy-preserving mutual authentication protocol for RFID systems. In *IEEE GLOBECOM Workshops*, pages 1083–1087. IEEE, December 2011.

[79] Martin Feldhofer, Johannes Wolkerstorfer, and Vincent Rijmen. AES Implementation on a Grain of Sand. *IEE Proceedings – Information Security*, 152(1):13–20, October 2005.

[80] Martin Feldhofer, Johannes Wolkerstorfer, and Vincent Rijmen. AES Implementation on a Grain of Sand. *IEE Proceedings – Information Security*, 152(1):13–20, October 2005.

[81] Harinda Fernando and Jemal Abawajy. Mutual authentication protocol for networked RFID systems. In *10th International Conference on Trust, Security and Privacy in Computing and Communications – TrustCom 2011*, pages 417–424, November 2011.

[82] Amos Fiat and Moni Naor. Rigorous time/space tradeoffs for inverting functions. In *ACM Symposium on Theory of Computing – STOC'91*, pages 534–541, New Orleans, Louisiana, USA, May 1991. ACM, ACM Press.

[83] Robert Gallager and David Van Voorhis. Optimal source codes for geometrically distributed integer alphabets. *Information Theory, IEEE Transactions on*, 21(2):228–230, 1975.

[84] Lijun Gao, Maode Ma, Yantai Shu, and Yuhua Wei. A security protocol resistant to intermittent position trace attacks and desynchronization attacks in RFID systems. *Wireless Personal Communications*, pages 1–17, July 2012.

[85] Lijun Gao, Maode Ma, Yantai Shu, and Yuhua Wei. An ultralightweight RFID authentication protocol with CRC and permutation. *Journal of Network and Computer Applications*, 36(6), November 2013.

[86] Jovan Dj. Golić. Cryptanalysis of alleged a5 stream cipher. In *Advances in Cryptology – EUROCRYPT'97*, volume 1233 of *Lecture Notes in Computer Science*, pages 239–255, Konstanz, Germany, May 1997. Springer.

[87] Zheng Gong, Svetla Nikova, and Yee Wei Law. Klein: a new family of lightweight block ciphers. In *RFID. Security and Privacy*, pages 1–18. Springer, 2012.

[88] Ronald L. Graham, Donald Ervin Knuth, and Oren Patashnik. *Concrete mathematics: a foundation for computer science.* Addison-Wesley Reading, 1989.

[89] Tony Guilfoyle. The zeitcontrol basiccard family. `http://www.basiccard.com`, 2009.

[90] Jian Guo, Thomas Peyrin, and Axel Poschmann. The photon family of lightweight hash functions. In *Advances in Cryptology–CRYPTO 2011*, pages 222–239. Springer, 2011.

[91] Jian Guo, Thomas Peyrin, Axel Poschmann, and Matt Robshaw. The led block cipher. In *Cryptographic Hardware and Embedded Systems–CHES 2011*, pages 326–341. Springer, 2011.

[92] Jitendra Gurubani, Harsh Thakkar, and Dhiren Patel. Improvements over extended LMAP+: RFID authentication protocol. In Theo Dimitrakos, Rajat Moona, Dhiren Patel, and D. McKnight, editors, *6th International Conference on Trust Management – IFIPTM 2012*, volume 374 of *IFIP Advances in Information and Communication Technology*, pages 225–231, Surat, India, May 2012. Springer Boston.

[93] Tzipora Halevi, Nitesh Saxena, and Shai Halevi. Using HB Family of Protocols for Privacy-Preserving Authentication of RFID Tags in a Population. In *Workshop on RFID Security – RFIDSec'09*, Leuven, Belgium, July 2009.

[94] Gerhard P. Hancke and Markus Kuhn. An RFID Distance Bounding Protocol. In *Conference on Security and Privacy for Emerging Areas in Communication Networks – SecureComm 2005*, pages 67–73, Athens, Greece, September 2005. IEEE, IEEE Computer Society.

[95] Daniel Hein, Johannes Wolkerstorfer, and Norbert Felber. ECC is Ready for RFID A Proof in Silicon. In *Workshop on RFID Security – RFIDSec'08*, Budapest, Hungary, July 2008.

[96] Martin Hellman. A cryptanalytic time-memory trade-off. *Information Theory, IEEE Transactions on*, 26(4):401–406, 1980.

[97] Martin Hellman. A cryptanalytic time-memory trade off. *IEEE Transactions on Information Theory*, IT-26(4):401–406, July 1980.

[98] Dirk Henrici and Paul Müller. Hash-Based Enhancement of Location Privacy for Radio-Frequency Identification Devices Using Varying Identifiers. In Ravi Sandhu and Roshan Thomas, editors, *International Workshop on Pervasive Computing and Communication Security – PerSec 2004*, pages 149–153, Orlando, Florida, USA, March 2004. IEEE, IEEE Computer Society.

[99] Jens Hermans, Andreas Pashalidis, Frederik Vercauteren, and Bart Preneel. A new RFID privacy model. In *16th European Symposium on Research in Computer Security – ESORICS 2011*, Lecture Notes in Computer Science, Leuven, Belgium, September 2011. Springer.

[100] Julio C. Hernandez-Castro, Juan M. Estevez-Tapiador, Pedro Peris-Lopez, and Jean-Jacques Quisquater. Cryptanalysis of the SASI Ultralightweight RFID Authentication Protocol with Modular Rotations. In *International Workshop on Coding and Cryptography – WCC'09*, Ullensvang, Norway, May 2009.

[101] Julio C. Hernandez-Castro, Pedro Peris-Lopez, Raphael C.W. Phan, and Juan M. Estevez-Tapiador. Cryptanalysis of the David-Prasad RFID Ultralightweight Authentication Protocol. In S.B. Ors Yalcin, editor, *Workshop on RFID Security – RFIDSec'10*, volume 6370 of *Lecture Notes in Computer Science*, pages 22–34, Istanbul, Turkey, June 2010. Springer.

[102] Julio C. Hernandez-Castro, Pedro Peris-Lopez, Juan M. E. Tapiador, Raphael C.-W. Phan, and Tieyan Li. Passive Black-Box Cryptanalysis of an Ultralightweight Protocol after Eavesdropping One Authentication Session. In *Workshop on RFID Security – RFIDSec Asia'11*, volume 6 of *Cryptology and Information Security*, pages 3–17, Wuxi, China, April 2011. IOS Press.

[103] Julio C. Hernandez-Castro, Juan E. Tapiador, Pedro Peris-Lopez, John A. Clark, and El-Ghazali Talbi. Metaheuristic Traceability Attack against SLMAP, an RFID Lightweight Authentication Protocol. In *Proceedings of the 23rd IEEE International Parallel and Distributed Processing Symposium – IPDPS 2009*, Rome, Italy, May 2009. IEEE, IEEE Computer Society.

[104] Hitatchi. Symmetric key encipherment method "m6" for IEEE 1394 bus encryption/authentication. Submission 1997-4-25, Proposal for IEEE 1394, Copy Prorection Technical Working Group, 1997.

[105] Deukjo Hong, Jaechul Sung, Seokhie Hong, Jongin Lim, Sangjin Lee, Bonseok Koo, Changhoon Lee, Donghoon Chang, Jaesang Lee, Kitae Jeong, Hyun Kim, Jongsung Kim, and Seongtaek Chee. HIGHT: A new block cipher suitable for low-resource device. In Louis Goubin and Mitsuru Matsui, editors, *Cryptographic Hardware and Embedded Systems – CHES 2006*, volume 4249 of *Lecture Notes in Computer Science*, pages 46–59, Yokohama, Japan, November 2006. Springer.

[106] Nicholas J. Hopper and Manuel Blum. Secure human identification protocols. In Colin Boyd, editor, *Advances in Cryptology – Asiacrypt 2001*, volume 2248 of *Lecture Notes in Computer Science*, pages 52–66, Gold Coast, Australia, December 2001. Springer.

[107] Michael Hutter, Martin Feldhofer, and Thomas Plos. An ECDSA Processor for RFID Authentication. In S.B. Ors Yalcin, editor, *Workshop on RFID Security – RFIDSec'10*, volume 6370 of *Lecture Notes in Computer Science*, pages 189–202, Istanbul, Turkey, June 2010. Springer.

[108] Salekul Islam. Security analysis of lmap using avispa. *International Journal of Security and Networks*, 9(1):30–39, 2014.

[109] Il-Soo Jeon and Eun-Jun Yoon. A new ultra-lightweight RFID authentication protocol using merge and separation operations. *International Journal of Mathematical Analysis*, 7(52):2583–2593, October 2013.

[110] Ari Juels. Minimalist Cryptography for Low-Cost RFID Tags. In Carlo Blundo and Stelvio Cimato, editors, *International Conference on Security in Communication Networks – SCN 2004*, volume 3352 of *Lecture Notes in Computer Science*, pages 149–164, Amalfi, Italy, September 2004. Springer.

[111] Ari Juels, Ronald Rivest, and Michael Szydlo. The Blocker Tag: Selective Blocking of RFID Tags for Consumer Privacy. In Sushil Jajodia, Vijayalakshmi Atluri, and Trent Jaeger, editors, *Conference on Computer and Communications Security – ACM CCS'03*, pages 103–111, Washington, DC, USA, October 2003. ACM, ACM Press.

[112] Ari Juels and Stephen Weis. Authenticating Pervasive Devices with Human Protocols. In Victor Shoup, editor, *Advances in Cryptology – CRYPTO'05*, volume 3126 of *Lecture Notes in Computer Science*, pages 293–308, Santa Barbara, California, USA, August 2005. Springer.

[113] Ari Juels and Stephen Weis. Defining Strong Privacy for RFID. In *International Conference on Pervasive Computing and Communications – PerCom 2007*, pages 342–347, New York City, NY, USA, March 2007. IEEE, IEEE Computer Society.

[114] BS Kaliski and Yiqun Lisa Yin. On the security of the RC5 encryption algorithm. Technical report, RSA Laboratories Technical Report TR-602, 1998.

[115] Süleyman Kardaş, Albert Levi, and Ertugrul Murat. Providing Resistance against Server Information Leakage in RFID Systems. In *New Technologies, Mobility and Security – NTMS'11*, pages 1–7, Paris, France, February 2011. IEEE, IEEE Computer Society.

[116] John Kelsey, Bruce Schneier, and David Wagner. Mod n cryptanalysis, with applications against RC5P and M6. In *Fast Software Encryption*, pages 139–155. Springer, 1999.

[117] Aaron Kiely. Selecting the golomb parameter in rice coding. *IPN Progress Report*, 42(159), November 2004.

[118] Alexander Klimov and Adi Shamir. New applications of t-functions in block ciphers and hash functions. In *Fast Software Encryption*, pages 18–31. Springer, 2005.

[119] Leslie Lamport. Password authentication with insecure communication. *Communications of the ACM*, 24(11):770–772, 1981.

[120] Gregor Leander, Christof Paar, Axel Poschmann, and Kai Schramm. New lightweight des variants. In *Fast Software Encryption*, pages 196–210. Springer, 2007.

[121] Ga Won Lee and Jin Hong. A comparison of perfect table cryptanalytic tradeoff algorithms. Cryptology ePrint Archive, Report 2012/540, 2012.

[122] Y.-C. Lee, Y.-C. Hsieh, P.-S. You, and T.-C. Chen. A New Ultralightweight RFID Protocol with Mutual Authentication. In *WASE International Conference on Information Engineering – ICIE '09*, pages 58–61, Taiyuan, Shanxi, August 2009. IEEE, IEEE Computer Society.

[123] Yong Ki Lee, Kazuo Sakiyama, Lejla Batina, and Ingrid Verbauwhede. Elliptic-Curve-Based Security Processor for RFID. *IEEE Transactions on Computers*, pages 1514–1527, 2008.

[124] Yung-Cheng Lee. Two ultralightweight authentication protocols for low-cost RFID tags. *Applied Mathematics and Information Sciences*, 6(2S):425–431, May 2012.

[125] LG OPTIMUS 4X HD P880. Technical Specifications. `http://www.lg.com/uk/mobile-phones/lg-P880/technical-specifications`, 2013.

[126] Tieyan Li. Employing lightweight primitives on low-cost RFID tags for authentication. In *68th Vehicular Technology Conference – VTC 2008*, pages 1–5, September 2008.

[127] Tieyan Li and Robert H. Deng. Vulnerability Analysis of EMAP - An Efficient RFID Mutual Authentication Protocol. In *Second International Conference on Availability, Reliability and Security – AReS 2007*, Vienna, Austria, April 2007.

[128] Tieyan Li and Guilin Wan. SLMAP - a secure ultra-lightweight RFID mutual authentication protocol. In *Advances in Cryptology – CHINACRYPT'07*, Lecture Notes in Computer Science, pages 19–22, Cheng Du, China, October 2007. Springer.

[129] Tieyan Li and Guilin Wang. Security Analysis of Two Ultra-Lightweight RFID Authentication Protocols. In Hein Venter, Mariki Eloff, Les Labuschagne, Jan Eloff, and Rossouw Von Solms, editors, *IFIP TC-11 22nd International Information Security Conference – SEC 2007*, volume 232 of *IFIP*, pages 109–120, Sandton, Gauteng, South Africa, May 2007. IFIP, Springer.

[130] Libnfc. The free/libre near field communication (nfc) library. `http://nfc-tools.org/`.

[131] Chae Hoon Lim and Tymur Korkishko. mcrypton–a lightweight block cipher for security of low-cost rfid tags and sensors. In *Information Security Applications*, pages 243–258. Springer, 2006.

[132] Chae Hoon Lim and Taekyoung Kwon. Strong and Robust RFID Authentication Enabling Perfect Ownership Transfer. In Peng Ning, Sihan Qing, and Ninghui Li, editors, *International Conference on Information and Communications Security – ICICS'06*, volume 4307 of *Lecture Notes in Computer Science*, pages 1–20, Raleigh, North Carolina, USA, December 2006. Springer.

[133] Jie Ling and Jinwei Shen. New defending ultra-lightweight RFID authentication protocol against DoS attacks. In *3rd International Conference on Consumer Electronics, Communications and Networks – CECNet 2013*, pages 423–426, Xianning, China, November 2013.

[134] Li Lu, Jinsong Han, Lei Hu, Yunhao Liu, and Lionel Ni. Dynamic Key-Updating: Privacy-Preserving Authentication for RFID Systems. In *International Conference on Pervasive Computing and Communications – PerCom 2007*, pages 13–22, New York City, NY, USA, March 2007. IEEE, IEEE Computer Society.

[135] George Marsaglia. The marsaglia random number CDROM including the diehard battery of tests of randomness (1995). *`http://www.stat.fsu.edu/pub/diehard`*, 2008.

[136] James L. Massey. Guessing and entropy. In *International Symposium on Information Theory, ISIT 1994*, Trondheim, Norway, June 1994. IEEE.

[137] Stephen M. Matyas, Carl H. Meyer, and Jonathan Oseas. Generating strong one-way functions with cryptographic algorithm. *IBM Technical Disclosure Bulletin*, 27(10A):5658–5659, 1985.

[138] Nele Mentens, Lejla Batina, Bart Preneel, and Ingrid Verbauwhede. Cracking Unix passwords using FPGA platforms. SHARCS - Special Purpose Hardware for Attacking Cryptographic Systems, February 2005.

[139] Aikaterini Mitrokotsa, Melanie R. Rieback, and Andrew S. Tanenbaum. Classifying RFID Attacks and Defenses. *Information Systems Frontiers*, July 2009.

[140] David Molnar and David Wagner. Privacy and Security in Library RFID: Issues, Practices, and Architectures. In Vijayalakshmi Atluri, Birgit Pfitzmann, and Patrick Drew McDaniel, editors, *Conference on Computer and Communications Security – ACM CCS'04*, pages 210–219, Washington, DC, USA, October 2004. ACM, ACM Press.

[141] Jorge Munilla and Alberto Peinado. HB-MP: A further step in the HB-family of lightweight authentication protocols. *Computer Networks*, 51(9):2262–2267, 2007.

[142] Arvind Narayanan and Vitaly Shmatikov. Fast dictionary attacks on passwords using time-space tradeoff. In *ACM Conference on Computer and Communications Security, CCS 2005*, Alexandria, VA, USA, November 2005. ACM.

[143] Huansheng Ning, Hong Liu, and Chen Yang. Ultralightweight RFID authentication protocol based on random partitions of pseudorandom identifier and pre-shared secret value. *Chinese Journal of Electronics*, 20(4):701–707, October 2011.

[144] Yasunobu Nohara and Sozo Inoue. A Secure and Scalable Identification for Hash-based RFID Systems Using Updatable Pre-computation. In Susanne Wetzel, Cristina Nita-Rotaru, and Frank Stajano, editors, *Proceedings of the 3rd ACM Conference on Wireless Network Security – WiSec'10*, pages 65–74, Hoboken, New Jersey, USA, March 2010. ACM, ACM Press.

[145] Yasunobu Nohara, Sozo Inoue, and Hiroto Yasuura. A secure high-speed identification scheme for RFID using bloom filters. In *Third International Conference on Availability, Reliability and Security – AReS 2008*, pages 727–722, Barcelona, Spain, March 2008.

[146] Karsten Nohl and David Evans. Quantifying Information Leakage in Tree-Based Hash Protocols. In Peng Ning, Sihan Qing, and Ninghui Li, editors, *International Conference on Information and Communications Security – ICICS'06*, volume 4307 of *Lecture Notes in Computer Science*, pages 228–237, Raleigh, North Carolina, USA, December 2006. Springer.

[147] Philippe Oechslin. Making a faster cryptanalytic time-memory trade-off. In Dan Boneh, editor, *Advances in Cryptology – CRYPTO'03*, volume 2729 of *Lecture Notes in Computer Science*, pages 617–630, Santa Barbara, California, USA, August 2003. Springer.

[148] Philippe Oechslin. The ophcrack password cracker. `http://ophcrack.sourceforge.net/`, 2013.

[149] Miyako Ohkubo, Koutarou Suzuki, and Shingo Kinoshita. Cryptographic Approach to "Privacy-Friendly" Tags. In *RFID Privacy Workshop*, MIT, MA, USA, November 2003.

[150] Miyako Ohkubo, Koutarou Suzuki, and Shingo Kinoshita. Efficient Hash-Chain Based RFID Privacy Protection Scheme. In *International Conference on Ubiquitous Computing – Ubicomp, Workshop Privacy: Current Status and Future Directions*, Nottingham, England, September 2004.

[151] Khaled Ouafi and Raphael C.-W. Phan. Privacy of Recent RFID Authentication Protocols. In Liqun Chen, Yi Mu, and Willy Susilo, editors, *4th International Conference on Information Security Practice and Experience – ISPEC 2008*, volume 4991 of *Lecture Notes in Computer Science*, pages 263–277, Sydney, Australia, April 2008. Springer.

[152] Liaojun Pang, Huixian Li, Liwei He, Ali Alramadhan, and Yumin Wang. Secure and efficient lightweight RFID authentication protocol based on fast tag indexing. *International Journal of Communication Systems*, March 2013.

[153] Pedro Peris-Lopez, Julio C. Hernandez-Castro, Juan M. Estevez-Tapiador, Tieyan Li, and Jan C.A. van der Lubbe. Weaknesses in Two Recent Lightweight RFID Authentication Protocols. In *Workshop on RFID Security – RFIDSec'09*, Leuven, Belgium, July 2009.

[154] Pedro Peris-Lopez, Julio C. Hernandez-Castro, Juan M. Estevez-Tapiador, and Arturo Ribagorda. EMAP: An Efficient Mutual Authentication Protocol for Low-Cost RFID Tags. In *OTM Federated Conferences and Workshop: IS Workshop – IS'06*, volume 4277 of *Lecture Notes in Computer Science*, pages 352–361, Montpellier, France, November 2006. Springer.

[155] Pedro Peris-Lopez, Julio C. Hernandez-Castro, Juan M. Estevez-Tapiador, and Arturo Ribagorda. LMAP: A Real Lightweight Mutual Authentication Protocol for Low-cost RFID tags. In *Workshop on RFID Security – RFIDSec'06*, Graz, Austria, July 2006. Ecrypt.

[156] Pedro Peris-Lopez, Julio C. Hernandez-Castro, Juan M. Estevez-Tapiador, and Arturo Ribagorda. M2AP: A Minimalist Mutual-Authentication Protocol for Low-cost RFID Tags. In Jianhua Ma, Hai Jin, Laurence Tianruo Yang, and Jeffrey J. P. Tsai, editors, *International Conference on Ubiquitous Intelligence and Computing – UIC'06*, volume 4159 of *Lecture Notes in Computer Science*, pages 912–923, Wuhan and Three Gorges, China, September 2006. Springer.

[157] Pedro Peris-Lopez, Julio C. Hernandez-Castro, Juan M. Estevez-Tapiador, and Arturo Ribagorda. Advances in Ultralightweight Cryptography for Low-cost RFID Tags: Gossamer Protocol. In Kyo-Il Chung, Kiwook Sohn, and Moti Yung, editors, *Workshop on Information Security Applications – WISA'08*, volume 5379 of *Lecture Notes in Computer Science*, pages 56–68, Jeju Island, Korea, September 2008. Springer.

[158] Pedro Peris-Lopez, Julio C. Hernandez-Castro, Juan M. Estevez-Tapiador, and Jan C. A. van der Lubbe. Security Flaws in a Recent Ultralightweight RFID Protocol. In *Workshop on RFID Security – RFIDSec Asia'10*, volume 4 of *Cryptology and Information Security*, pages 83–93, Singapore, Republic of Singapore, February 2010. IOS Press.

[159] Pedro Peris-Lopez, Julio C. Hernandez-Castro, Raphael C.-W. Phan, Juan M. E. Tapiador, and Tieyan Li. Quasi-Linear Cryptanalysis of a Secure RFID Ultralightweight Authentication Protocol. In *6th China International Conference on Information Security and Cryptology – Inscrypt'10*, Shanghai, China, October 2010. Springer.

[160] Alexander Peslyak. The John the Ripper password cracker. `http://www.openwall.com/john/`, 2014.

[161] Raphael C.-W. Phan. Cryptanalysis of a New Ultralightweight RFID Authentication Protocol - SASI. *IEEE Transactions on Dependable and Secure Computing*, 99(1), 2008.

[162] Raphael C.-W. Phan, Jiang Wu, Khaled Ouafi, and Douglas R. Stinson. Privacy analysis of forward and backward untraceable rfid authentication schemes. *Wirel. Pers. Commun.*, 61(1):69–81, November 2011.

[163] Cai Qingling, Zhan Yiju, and Wang Yonghua. A Minimalist Mutual Authentication Protocol for RFID System & BAN Logic Analysis. In *ISECS International Colloquium on Computing, Communication, Control, and Management – CCCM'08.*, volume 2, pages 449–453, August 2008.

[164] Robert Rice and James Plaunt. Adaptive variable-length coding for efficient compression of spacecraft television data. *Communication Technology, IEEE Transactions on*, 19(6):889–897, 1971.

[165] Ronald Rivest. The RC5 encryption algorithm. In Bart Preneel, editor, *Proceedings of the Second International Workshop on Fast Software Encryption – FSE 1994*, volume 1008 of *Lecture Notes in Computer Science*, pages 86–96, Leuven, Belgium, December 1995. Springer, Springer.

[166] Samad Rostampour, Mojtaba Eslamnezhad Namin, and Mehdi Hosseinzadeh. A novel mutual rfid authentication protocol with low complexity and high security. *International Journal of Modern Education and Computer Science*, 2014.

[167] Andrew Rukhin, Juan Soto, James Nechvatal, Miles Smid, and Elaine Barker. A statistical test suite for random and pseudorandom number generators for cryptographic applications. Technical report, DTIC Document, 2001.

[168] Stuart J. Russell and Peter Norvig. *Artificial intelligence: a modern approach*, volume 2. Pearson Education, 2003.

[169] Markku-Juhani Olavi Saarinen. A time-memory tradeoff attack against LILI-128. In *Fast Software Encryption*, volume 2365, pages 231–236, Leuven, Belgium, February 2001.

[170] Shankland, Stephen. Google's Android parts ways with Java industry group. CNET News. Retrieved 2012-02-15, November 12, 2007.

[171] François-Xavier Standaert, Gilles Piret, Neil Gershenfeld, and Jean-Jacques Quisquater. Sea: A scalable encryption algorithm for small embedded applications. In *Smart Card Research and Advanced Applications*, pages 222–236. Springer, 2006.

[172] Hung-Min Sun, Shuai-Min Chen, and King-Hang Wang. Cryptanalysis on the RFID ACTION protocol. In *International Conference on Security and Management – SAM 2011*, Las Vegas, Nevada, USA, July 2011.

[173] Hung-Min Sun, Wei-Chih Ting, and King-Hang Wang. On the Security of Chien's Ultra-Lightweight RFID Authentication Protocol. *IEEE Transactions on Dependable and Secure Computing*, 99(PrePrints), 2009.

[174] Deepak Tagra, Musfiq Rahman, and Srinivas Sampalli. Technique for Preventing DoS Attacks on RFID Systems. In *18th International Conference on Software Telecommunications and Computer Networks – SoftCOM'10*, Bol, Island of Brac, Croatia, September 2010. IEEE, IEEE Computer Society.

[175] D.R. Thompson, N. Chaudhry, and C.W. Thompson. RFID security threat model. In *Conf. on Applied Research in Information Technology*. Citeseer, 2006.

[176] Yun Tian, Gongliang Chen, and Jianhua Li. A new ultralightweight RFID authentication protocol with permutation. *IEEE Communications Letters*, 16(5):702–705, May 2012.

[177] Wiem Tounsi, Nora Cuppens-Boulahia, Joaquin Garcia-Alfaro, Yannick Chevalier, and Frdric Cuppens. Kedgen2: A key establishment and derivation protocol for {EPC} gen2 {RFID} systems. *Journal of Network and Computer Applications*, 39(0):152 – 166, 2014.

[178] Gene Tsudik. YA-TRAP: Yet Another Trivial RFID Authentication Protocol. In *International Conference on Pervasive Computing and Communications – PerCom 2006*, pages 640–643, Pisa, Italy, March 2006. IEEE, IEEE Computer Society.

[179] Gene Tsudik. A Family of Dunces: Trivial RFID Identification and Authentication Protocols. In Nikita Borisov and Philippe Golle, editors, *Workshop on Privacy Enhancing Technologies – PET 2007*, volume 4776 of *Lecture Notes in Computer Science*, pages 45–61, Ottawa, Canada,, June 2007. Springer.

[180] Harshal Tupsamudre, Shikha Bisht, and Debdeep Mukhopadhyay. Differential fault analysis on the families of simon and speck ciphers. Technical report, Cryptology ePrint Archive, Report 2014/267, 2014., 2014.

[181] Ton van Deursen, Sjouke Mauw, and Saša Radomirović. Untraceability of RFID Protocols. In Jose Antonio Onieva, Damien Sauveron, Serge Chaumette, Dieter Gollmann, and Constantinos Markantonakis, editors, *Workshop on Information Security Theory and Practice – WISTP'08*, volume 5019 of *Lecture Notes in Computer Science*, pages 1–15, Sevilla, Spain, May 2008. Springer.

[182] Tri Van Le, Mike Burmester, and Breno de Medeiros. Universally Composable and Forward-secure RFID Authentication and Authenticated Key Exchange. In Feng Bao and Steven Miller, editors, *ACM Symposium on Information, Computer and Communications Security – ASIACCS 2007*, pages 242–252, Singapore, Republic of Singapore, March 2007. ACM, ACM Press.

[183] Serge Vaudenay. RFID Privacy Based on Public-Key Cryptography (Abstract). In Min Surp Rhee and Byoungcheon Lee, editors, *International Conference on Information Security and Cryptology – ICISC 2006*, volume 4296 of *Lecture Notes in Computer Science*, pages 1–6, Busan, Korea, November–December 2006. Springer.

[184] Serge Vaudenay. On Privacy Models for RFID. In Kaoru Kurosawa, editor, *Advances in Cryptology – Asiacrypt 2007*, volume 4833 of *Lecture Notes in Computer Science*, pages 68–87, Kuching, Malaysia, December 2007. Springer.

[185] John Walker. ENT, a pseudorandom number sequence test program. *Fourmilab, Oct*, 1998.

[186] Weijia Wang, Yong Li, Lei Hu, and Li Lu. Storage-awareness: RFID private authentication based on sparse tree. In *Security, Privacy and Trust in Pervasive and Ubiquitous Computing, 2007. SECPerU 2007. Third International Workshop on*, pages 61–66. IEEE, 2007.

[187] Stephen Weis. Security and Privacy in Radio-Frequency Identification Devices. Master thesis, Massachusetts Institute of Technology (MIT), MIT, Massachusetts, USA, May 2003.

[188] David J Wheeler and Roger M Needham. Tea, a tiny encryption algorithm. In *Fast Software Encryption*, pages 363–366. Springer, 1995.

[189] Lin Yang, Meiqin Wang, and Siyuan Qiao. Side channel cube attack on PRESENT. In Juan A. Garay, Atsuko Miyaji, and Akira Otsuka, editors, *8th International Conference on Cryptology And Network Security – CANS'09*, volume 5888 of *Lecture Notes in Computer Science*, pages 379–391, Kanazawa, Japan, December 2009. Springer.

[190] Qingsong Yao, Yong Qi, Jinsong Han, Jizhong Zhao, Xiangyang Li, and Yunhao Liu. Randomizing RFID private authentication. In *Pervasive Computing and Communications, 2009. PerCom 2009. IEEE International Conference on*, pages 1–10. IEEE, 2009.

[191] Kuo-Hui Yeh, Naiwei Lo, and Enrico Winata. An Efficient Ultralightweight Authentication Protocol for RFID Systems. In *Workshop on RFID Security – RFIDSec Asia'10*, volume 4 of *Cryptology and Information Security*, Singapore, Republic of Singapore, February 2010. IOS Press.

[192] Kuo-Hui Yeh and N.W. Lo. Improvement of Two Lightweight RFID Authentication Protocols. *Information Assurance and Security Letters – IASL 2010*, 1:6–11, 2010.