Tampering in Wonderland

PhD Candidate: Daniele Venturi

Promoter: Andrea Baiocchi

Thesis submitted in partial fulfilment of the PhD in
Information and Communication Engineering (XXIV ciclo)

SAPIENZA
UNIVERSITÀ DI ROMA

DANIELE VENTURI

TAMPERING IN WONDERLAND

SAPIENZA
UNIVERSITÀ DI ROMA

PHD THESIS

Advisor: Prof. Andrea Baiocchi

SAPIENZA Università di Roma
Facoltà di Ingegneria dell'Informazione Informatica e Statistica
DIET
December 2011

Dedicato ad Anna ed alla sua dolce bellezza

ABSTRACT

Cryptography (originally meaning *"the art of secret communication"*) is an ancient discipline born in the past to satisfy some needs of the human being. Today the world "cryptography" has a much broader meaning. In fact, the development of the *digital era* (mainly due to the introduction of the *digital calculators*) and later on of the *Internet* have created new scenarios of communications, generating intriguing challenges for modern "cryptographers".

In the past (say before the 80's) the development of a "secure" system was mostly based on intuition and experience. This approach led to a cut-and-mouse game in which new schemes were proposed and (immediately after) new attacks against them were found. A milestone in the development of a solid mathematical theory—turning cryptography from an art into a real science—is an approach due to Goldwasser and Micali called *"provable security"*. The goal of this approach is to state rigorous definitions of what "secure" means in a given context and build cryptographic primitives able to satisfy these definitions in a provable way. This is indeed a very strong guarantee.

It should be clear that the way one defines security is crucial for the results being meaningful. In particular our mathematical model of reality should be as close as possible to the real world, catching all possible attacks out there. Unfortunately, it turned out that this is not always the case. There are in fact attacks—so called side-channel attacks—which can be applied to an actual implementation of a device and not to its mathematical abstraction. These attacks have been shown to be very powerful, completely compromising security of otherwise provably secure schemes.

A modern trend in theoretical cryptography is to try filling this gap between theory and practice, building primitives which maintain their provably secure guarantees even in the presence of a powerful adversary able to apply side-channels. This area of research is evolving continuously and very quickly. The thesis you are reading deals with some of these challenges and summarises recent achievements in this topic.

PUBLICATIONS

The main contributions presented in this thesis are taken from two papers I wrote during my PhD. The first paper [FPV11] has been published in ICALP 2011. The second paper [Kil+11] got the best paper award in Eurocrypt 2011.

Besides these two projects, I have been involved in several others in several areas of cryptography, such as leakage resilience, traffic analysis, zero knowledge, cloud storage and quantum cryptography.

ACKNOWLEDGEMENTS

On my way to the PhD, I met several people and I own a big "Grazie!" to all of them.

My first "Grazie!" is to Andrea Baiocchi for being the advisor of this thesis: I know I was not an "easy" student to deal with, but Andrea always gave me his unconditional support, letting me free to follow my research interests.

I am in debt with Stefan Dziembowski and Krzysztof Pietrzak: they have been my mentors in Rome and in Amsterdam (respectively) and all I know about cryptography started from them. Thanks for being always so nice and helpful and for answering to all my questions (even the trivial ones), and sharing ideas.

During my PhD I had the wonderful opportunity to visit CWI in Amsterdam for a whole year. Needless to say, this experience made me a much better person, giving me also the opportunity to get in touch with many interesting people and friends. It has been really a great experience! Thanks to Ronald Cramer for hosting me and to Eike Kiltz, Sebastian Faust, Nacho Cascudo, Otto Johnston, Niek Bouman, Alp Bassa, Giannicola Scarpa, Christian Schaffner and Abhishek Jain for making my time in Amsterdam so pleasant!. Besides Amsterdam, I also visited three other groups in Europe: TU Darmstadt in Germany, ESAT/COSIC in Belgium and Aarhus University in Denmark. Thanks to Marc Fischlin, Bart Preneel and Ivan Damgård for hosting me, and to Cristina Onete, Özgür Dagdelen, Andreas Peter, Christina Brzuska, Sebastian Faust for taking care of me. Among all these people, Abhishek, Sebastian and Özgür are the ones with whom I spent more time: we always had amazing (and fruitful) discussions and a lot of fun together. It's very nice to have them as friends besides as co-authors...

Other people that I worked/discussed with include: Eike Kiltz, David Cash, Francesco Davì, Kenny Paterson, Michel Abdalla, Giorgia Marson, Giuseppe Ateniese, Pierpaolo Salvo, Anna Abbagnale and all my other office-mates from "La Sapienza". Last but not least I want to thank my family, for their unconditional support and Anna, for her wonderful smile and for staying always close to me during the last three years.

"Grazie" to all of you, without you this thesis would not have been the same!

Rome, december 2011 *Daniele Venturi*

CONTENTS

1 INTRODUCTION

The White Rabbit put on his spectacles. «Where shall I begin, please your Majesty?» he asked. «Begin at the beginning,» the King said gravely, «and go on till you come to the end: then stop.»

Lewis Carroll, Alice's Adventures in Wonderland [Car65]

CONTENTS

Cryptography (originally meaning *"the art of secret communication"*) is an ancient discipline born in the past to satisfy some needs of the human being. A concrete example is in the military context, where two or more parties (far away from each other) want to communicate secretly over an insecure channel (eventually) controlled by the "enemy". Solutions to this kind of problems have a long and rich history, taking us back at the time of the Roman emperor Giulio Cesare and even earlier.

Today the world "cryptography" has a much broader meaning. In fact, the development of the *digital era* (mainly due to the introduction of the *digital calculators*) and later on of the *Internet* has created new scenarios of communications, generating intriguing challenges for modern "cryptographers".

In the past (say before the 80's) the development of a "secure" system was mostly based on intuition and experience. This approach led to a cut-and-mouse game in which new schemes were proposed and (immediately after) new attacks against break them were discovered. A milestone in the development of a solid mathematical theory—turning cryptography from an art into a real science—is an approach due to Goldwasser and Micali called *"provable security"*. The goal of this approach is to state rigorous definitions of what "secure" means in a given context and build cryptographic primitives able to satisfy these definitions in a provable way. This is indeed a very strong guarantee.

It should be clear that the way one defines security is crucial for the results being meaningful. In particular our mathematical model of reality' should be as close as possible to the real world, catching all possible attacks out there. Unfortunately, it turned out that this is not always the case. There are in fact attacks—so called side-channel attacks—which can be applied to an actual implementation of a device and not to its mathematical abstraction. These attacks have been shown to be very powerful, completely compromising security of otherwise provably secure schemes.

A modern trend in theoretical cryptography is to try filling this gap between theory and practice, building primitives which maintain their provably secure guarantees even in the presence of a powerful adversary able to apply side-channels.

This area of research is evolving continuously and very quickly. The thesis you are reading deals with some of these challenges and summarises recent achievements in this topic.

READER'S GUIDE. In Section 1.1 we will give a brief overview of the modern approach to cryptography, which will be also the starting point for the rest of the thesis. Section 1.2 explains the principles behind side-channel attacks. Finally, Section 1.3 emphasizes the contributions of this thesis.

1.1 A BEAUTIFUL THEORY

Many applications today need to satisfy strict security requirements. Concrete examples are credit cards, online banking and electronic voting. However, the construction of a "secure" scheme is a difficult task. In fact, we would like our scheme to be able to resist all *known* attacks. Ideally, we would actually like that it will be hard to break the scheme also in the future, when *new* attacks are discovered. How can this be possible? What makes the task of cryptographers difficult, is that we should be able to predict how a malicious adversary will abuse of our system in order to break it.

In the early days cryptographers were almost guided by experience and intuition. Sometimes the consequences of this approach have been devastating, leading to the massive use of presumably secure schemes that were violated after some time. Concrete examples can be found in the GSM telephony system [BBK03] and in the standard PKCS #1.5 as used in RSA [Ble98].

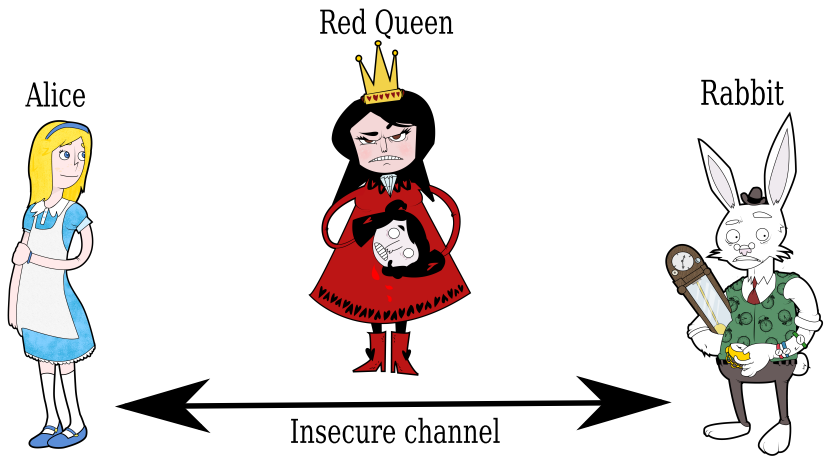

Figure 1.1: Alice and the Rabbit want to communicate over an insecure channel controlled by the Red Queen.

CRYPTOGRAPHY IN WONDERLAND. To change the above state of affairs, cryptographers started to change radically the way to develop a secure system. A cornerstone in this direction, is an approach of Goldwasser and Micali [GM82] called *provable security*. The basic idea is to give a mathematical proof that a given system is secure against an as large as possible class of attacks: This way we don't need to care about future attacks, our system will be able to face them as long as the new attacks fall in the class of attacks against which we have proven security.

Metaphorically, in this thesis we will identify the realm of provable security with Lewis Carroll's "Wonderland" [Car65]. The reason for this is that we should always remember that we are dealing with a mathematical abstraction of reality. Even if we believe that our abstraction represents reality quite well, we must remember it is an abstraction. In particular, a result in the realm of provable security maintains its validity only inside our "Wonderland" and we have to be extremely careful whenever we decide to deploy a scheme in the real world. As we will see in Section 1.2, any inconsistency between the real world and our abstraction could have dramatic consequences, completely invalidating all our "provable security" guarantees.

The most basic[1] scenario in Wonderland sees Alice and the Rabbit exchanging messages over an insecure channel for the purpose of communication. The

1 Let us mention that there are a lot of different security requirements in Wonderland (and thus also in the real world). However, mentioning all of them is by far out of the scope of this thesis.

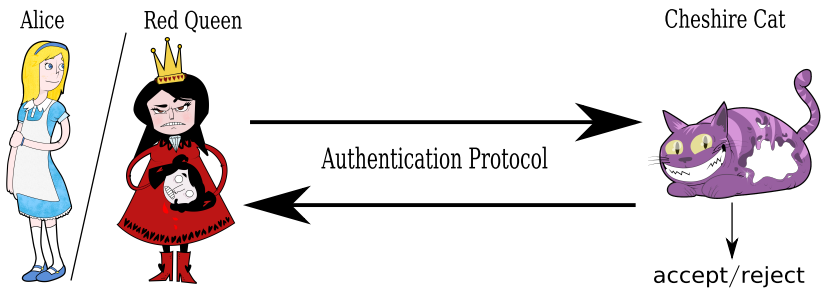

Figure 1.2: Alice wants to prove her identity to the Cheshire Cat. The Red Queen aims at impersonating Alice.

channel is controlled by the Red Queen, as shown in Figure 1.1. In this context we might think of the following security requirements:

1. *Confidentiality.* Alice wishes to send a message m to the Rabbit in such a way that the message content is somehow hidden: The Rabbit must be able to read the message, but the Red Queen should not learn anything about m's content. (This is essentially the original purpose of cryptography, i.e. secret communication.) The primitive used to achieve this task is called an *encryption scheme* (cf. Section 2.2).

2. *Message integrity.* Alice wishes to send a message m to the Rabbit in such a way that the message is received unchanged: The Rabbit must be convinced that the message m is exactly the original message sent by Alice, and the Red Queen should not be able to replace m with a different message m^* without the Rabbit being able to notice it. (This is important when, for instance, the message m contains informations about a money transfer from Alice to the Rabbit.) The primitive used to achieve this task is a *Message Authentication Code* (MAC) or a *digital signature* scheme (cf. Section 2.2).

A somewhat different scenario is the one of *authentication*. In this case Alice would like to convince the Cheshire Cat of her identity. After exchanging messages over an insecure channel, the Cheshire Cat should be able to either accept or reject Alice as "authentic" (cf. Figure 1.2). In the meanwhile, the Red Queen is eavesdropping the channel: Her purpose is to replace Alice in the authentication process, without the Cheshire Cat being able to notice it. A secure authentication scheme is a one where this is hard. (This is important for instance in the context of electronic identity cards.)

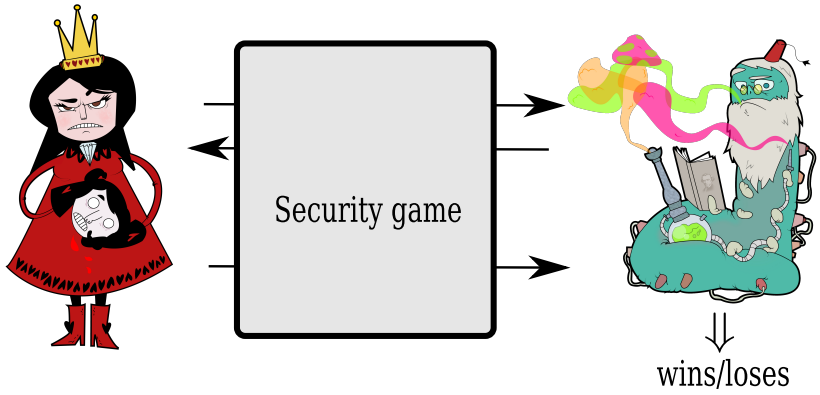

Figure 1.3: A security game formally defines security of a given cryptographic primitive.

The techniques used in the realm of provable security to meet the above requirements, fall into two main categories: *symmetric* techniques and *asymmetric* techniques. In the former, one assumes that the two honest parties share a secret key K. (How this is done is usually considered as a different problem.) In the latter the honest parties do not share any a priori information. (However other setup assumptions are necessary, cf. Section 2.2).

THE RECIPE OF PROVABLE SECURITY. On a very high level, the structure of a proof that a given scheme is indeed "secure" is as follows:

1. Abstract reality away, defining an accurate mathematical model of the real world in which the scheme is used.
2. Define rigorously what "secure" means in this model.
3. Prove that no adversary belonging to a certain class, acting following the rules of the model defined in (1), is able to break the security definition given in (2).

The way to obtain (1) and (2) is by introducing a security "game" between an adversary and a challenger. In Wonderland, these are the Red Queen and the Caterpillar: The Caterpillar behaves like an oracle abstracting away how the Red Queen can attack the system (cf. Figure 1.3). At the end of this interaction, the Red Queen attempts to "break"[2] the system and the Caterpillar determines

2 Of course the precise meaning of what "break" means depends on the primitive considered, cf. Section 2.2 for concrete examples.

whether she has won the game (i.e. she is successful) or not. We will thus say that a scheme is secure when winning the security game for that scheme is in some sense "hard".[3]

The way point (3) is obtained, depends on the class of adversaries against which we are willing to protect our scheme. There are two main classes:

- *Computationally unbounded attackers.* In this case we speak of *perfect* or *unconditional* security, a notion dating back originally to Shannon [Sha49]. There is no bound on the computational resources available to the adversary. Roughly speaking the adversary is not able to break the system because she has never enough "information" to do so.

- *Computationally bounded attackers.* In this case we speak of *computational* security. There is a precise bound on the computational resources available to the adversary. The main idea is to rely on the fact that there are computational problems which are hard to solve with limited resources; thus one shows that a successful attack to the system can be "reduced" to an efficient solution of the problem, which is impossible.

THE REDUCTIONIST APPROACH. Mathematics is rich of hard problems. Famous examples are integer factorisation and the computation of discrete logarithms (see for instance [Ven09, Chapter 6 and Chapter 7]). Roughly speaking these problems cannot be solved "efficiently".[4] In the world of computational security the goal is to protect the system against "efficient" adversaries (To formalise the meaning of "efficient" we will rely on computational complexity, cf. Section 2.2.) An efficient adversary can only perform efficient attacks; in this sense she is resource-bounded. The idea is to base security of a cryptographic scheme Π on the hardness of a computational problem P, for which no efficient solution exists. Then, in order to prove security, one is required to build a "reduction" showing that any successful efficient attack against Π can be used to define an efficient algorithm solving P: as long as we believe that P is hard this is a contradiction, thus showing that our system is indeed secure.

In Wonderland, the reductionist approach is explained as follows (cf. also Figure 1.4). The White Queen is given an instance of the problem P. Assume that the Red Queen is able to win the security game defined for Π (cf. Figure 1.3).

3 The meaning of "hard" is still intentionally vague. We will formalise this later on in the thesis (cf. Section 2.2).

4 It is not known whether an efficient solution exists at all. An answer to this question is ultimately related to an unproven conjecture in complexity theory, cf. Section 2.2.

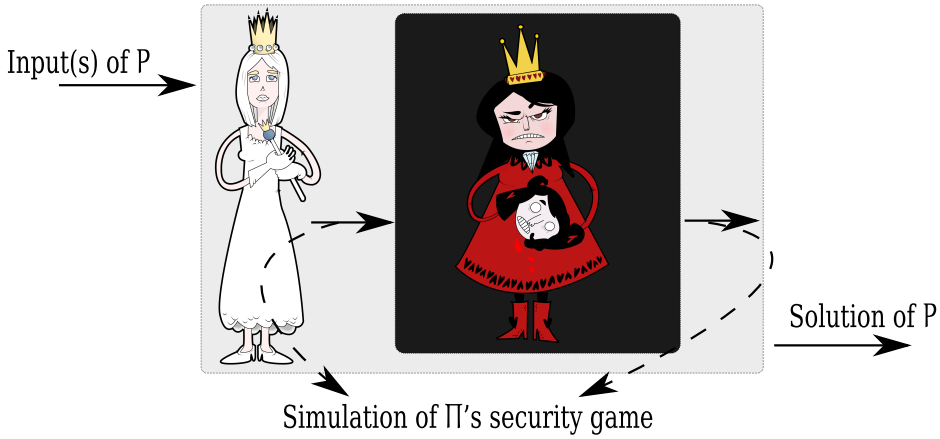

Figure 1.4: Cryptographic reductions.

The White Queen will attempt to solve P using the Red Queen as a black-box: She needs somehow to use the fact that the Red Queen is able to break Π to produce a solution for P. In order to do so, she has to perfectly simulate the "environment" for the Red Queen (who is expecting to attack Π). Once this is done, we have an efficient algorithm for solving P. Since P is hard, we have reached a contradiction. Thus we can conclude that it must be impossible to break Π and security is proven.

Note that the approach of computational security leads to schemes that could be broken in theory, disposing of enough time/computational power. One might ask why we should rely on computational security if it is possible to achieve the stronger notion of unconditional security. The reason for this is that, most of the times, unconditionally secure schemes are inherently inefficient (and sometimes even impossible). The approach of computational security is a natural way to relax the security requirements in a meaningful way, yielding constructions with reasonable efficiency and strong security guarantees.

A RUNNING EXAMPLE: THE RSA CRYPTOSYSTEM. To give a flavour of a cryptographic scheme, we briefly recap here the case of RSA [RSA78] for both (public key) encryption and digital signatures. (The reader can refer to Section 2.2 for an informal overview of these concepts.) Let $N = pq$ be the product of two primes, each, say, $\kappa/2$ bits long. The scheme works over $\mathbb{Z}_N^*$, i.e. over the

set of integers which are invertible[5] modulo N. There is a *public* exponent e and a *secret* exponent d which is part of the secret key of a given user (say the Rabbit) together with the primes p and q. The exponents have to satisfy $e \cdot d = 1 \bmod (p-1)(q-1)$; the RSA modulus N is public. (We don't specify here how the public key and the secret key are generated, however it is possible to do this efficiently.)

To encrypt a message m for the Rabbit, Alice computes $c = \mu(m)^e \bmod N$ using the Rabbit's public value e, and sends the result over to the channel. The function μ is an invertible encoding mapping the message space to $\mathbb{Z}_N^*$; concrete examples can be found in [BR96; Coro2]. The Rabbit can recover $\mu(m) = c^d \bmod N$ using the secret parameter d, and thus m inverting μ.[6] Security of the scheme follows from the hardness of factorising the integer N.[7].

To sign a message m, the Rabbit[8] computes $\sigma = \mu(m)^d \bmod N$ and sends the pair (m, σ) to Alice. Alice can use the public exponent e (of the Rabbit) to verify the signature, as $\mu(m) = \sigma^e \bmod N$. When the verification succeeds the message can be considered as authentic. Again, the security of this scheme follows from the hardness of factorisation.

1.2 THE CRUEL REALITY

As we have briefly mentioned in the previous section, a result in the realm of provable security maintains its validity only in Wonderland: If we are willing to use a cryptographic scheme in practice, we need to ensure that all the assumptions we made in order to prove security are still satisfied!

Unfortunately, as people have noticed in the past, this is not always the case. An example is given by the context of *side-channel attacks*. The main assumption underlying the realm of provable security is that—in the security game played by the Red Queen and the Caterpillar (cf. Figure 1.3)—the adversary has only

5 An integer x is invertible modulo N if there exists another integer x^{-1} such that $x^{-1}x = 1 \bmod N$. It is not hard to show that $x \in \mathbb{Z}_N$ is invertible if and only if $\gcd(x, N) = 1$.

6 Correctness follows from Euler's theorem, i.e. from the fact that for any integer x in $\mathbb{Z}_N^*$ we have $x^{(p-1)(q-1)} = 1 \bmod N$ when $N = pq$.

7 This is not completely exact. Of course if one is able to factorise N, there is no hope of security. However the other direction is not known to be true. One can prove that RSA is secure if it is difficult to compute e-th roots modulo N, but no direct reduction to the problem of factorising N is known, at least at the moment. (Though some result goes into that direction [AM09].)

8 It must be the Rabbit, because in our running example the parameters N, p, q, e, d are referred to the Rabbit.

black-box access to the primitive Π: she can specify an input X and get back the corresponding output Y, but all the secrets stored "inside the box" are completely oblivious to the Red Queen. For example, if Π depends on a secret key K, this assumption means that no information about K is leaked to the Red Queen.

In the 90's, it turned out that this assumption is too strong and in fact *not* satisfied in reality (at least not in a general sense). The main problem comes from the fact that in practice the system is not a black-box, but a *physical device* (e.g. a smart-card). Attacking such a device is significantly different than just observing its input-output behaviour. There are two main distinction, discussed below with some detail:

1. A *passive* adversary leaking informations on the secrets stored into the device.

2. An *active* adversary injecting faults into the device and then obtaining the output of the "tampered" computation.

PASSIVE ATTACKS. During the late 90's Kocher published two seminal papers showing the devastating power of side-channels. In the first paper [Koc96] he showed the vulnerability of many implementations against so-called *timing attacks*. These attacks exploits the dependency between the secret key and the time needed to perform certain cryptographic operations. If the time discrepancies can be measured efficiently and reasonably accurately, they often allow to recover the complete secret key at low cost.

A simple example of this fact can be found in the case of modular exponentiation as used in RSA. The RSA decryption algorithm is based on modular exponentiation modulo an integer N, where the exponent d is part of the secret key. For efficiency reasons, modular exponentiation is often performed using the "square-and-multiply" algorithm, as follows. Let $d = (d_{n-1}, \dots, d_0)_2$ be the base-2 representation of the integer d. Suppose we want to compute $c^d \bmod N$. We can write:

$$c^d \equiv c^{\sum_{i=0}^{n-1} d_i 2^i} \equiv \prod_{i=0}^{n-1} \left(c^{2^i}\right)^{d_i} \equiv \left(\left(c^{d_{n-1}}\right)^2 c^{d_{n-2}}\right)^2 \dots c^{d_0} \pmod{N}.$$

In this way one needs to perform at most $n \approx \log(d)$ modular multiplications. It is not difficult to see that the running time of the above algorithm increases linearly with the number of 1's in the exponent. For RSA decryption this implies that an adversary may easily learn the Hamming weight of the secret exponent. In his paper, Kocher extended this attack and presented a method to learn the

complete key when exponentiation is done with many different bases. Since in RSA the base c of the modular exponentiation represents the ciphertext, to perform Kocher's attacks one just needs to run the decryption process on many different ciphertexts.

In 1999 Kocher discovered another attack, based on *power analysis* [KJJ99]. Here the idea is to get some leakage on the secret key by exploiting the dependency between the latter and the power consumption of a device. In the most basic case this is done by visual inspection of a power trace. (This only works if the adversary has detailed knowledge about the implementation of the target device.) Assume that the order in which the instructions are performed depends on the secret key. Then identification of operations in the power trace may allow to learn the order of operations, and in turn allows to recover the secret key. For example in the "square-and-multiply" the order of operations (and the type of operations) is different depending on whether a key bit is 0 or 1. Hence, if the power traces allow to identify the type and order of operations, the key can be recovered.

Even when the detailed description of the implementation is not given, it is still possible to exploit power consumption in a *differential* power analysis attack. Differential power analysis is a more advanced form of power analysis which can be carried out even with very little information on the target implementation. The details of this technique are outside the scope of this thesis and are therefore omitted.

After Kocher's seminal work, numerous other side-channel attacks have been discovered, measuring for instance *electromagnetic radiation* [GMO01; QS01] and even the *sound* [ST] emitted by a device.

ACTIVE ATTACKS. Until this point we have only considered the case where an adversary obtains some leakage on the secret key by measuring some quantities related to a specific implementation of a cryptographic device. What about active adversaries? Imagine an attacker injecting faults into a specific implementation: After she has tampered with the device, she can query it on input X thus receiving the *modified* output Y' (potentially different from the normal output Y because of the tampering).

One might hope that the output of a tampered device is not very useful. However, as shown in a seminal paper by Boneh, De Millo and Lipton [BDL01] this is not quite true. Consider a smartcard producing RSA signatures. Let $N = pq$ be the RSA modulus. To sign a message m one has to compute $\sigma = \mu(m)^d \bmod N$ where d is part of the secret key. There is a trick mainly used to improve ef-

ficiency based on the Chinese Reminder Theorem[9] (CRT): To sign m one can compute $\sigma_p = \mu(m)^d \bmod p$ and $\sigma_q = \mu(m)^d \bmod q$, and finally recover σ as $\sigma = a\sigma_p + b\sigma_q \bmod N$ where $a = q(q^{-1} \bmod p)$, $b = p(p^{-1} \bmod q)$ are predetermined constants. Since $\sigma_p = \mu(m)^d \bmod p = \mu(m)^{d \bmod p-1} \bmod p$, using the CRT is much faster, because d is of the order of N whereas $d \bmod p - 1$ is of the order of p.

The attack of [BDL01] is as follows. Let $\sigma = \mu(m)^d \bmod N$ be a signature of m computed using RSA with CRT thorough the values σ_p and σ_q. Let σ' be a faulty signature of the same message m. Note that σ' is computed through values σ'_p and σ'_q. Assume that the adversary has tampered with the smartcard in such a way that an error occurs during the computation of only *one* of σ'_p and σ'_q, e.g. $\sigma'_p \not\equiv \sigma_p \bmod p$ but $\sigma'_q = \sigma_q$. (We note that this is very easy to achieve, provided that the implementation of the signature generation mechanism is known.) This implies $\sigma \equiv \sigma' \bmod q$, but $\sigma \not\equiv \sigma' \bmod p$. Thus

$$\gcd(\sigma - \sigma', N) = q,$$

and N is factorised.

A simple modification of the attack, first pointed out by Lenstra [Len96], shows that the correct signature is not needed when the message m is known to the adversary. In fact $\mu(m) \equiv (\sigma')^e \bmod q$—where e is the public exponent in RSA—but $\mu(m) \not\equiv (\sigma')^e \bmod p$. Since $\sigma^e \equiv \mu(m) \bmod N$, we have

$$\gcd\left(\mu(m) - (\sigma')^e, N\right) = q.$$

The main result of [BDL01] is an extension of the above attack to work even in the case where RSA is used without CRT. (For this to work more than a single faulty signature on m is required.) Several other attacks are known by tampering with the RSA public modulus [Sei05; Bri+06; Mui06; BCG08; Ber+09; BCDG10; Bri+11].

For an overview of tampering attacks on other systems different from RSA and how they can be realised in practice, we refer the reader to Section 3.1.

9 The Chinese Reminder Theorem is an old theorem about linear congruences which appeared for the first time in a Chinese third-century AD book by Sun Tzu. We will not give the exact statement here.

1.3 THESIS CONTRIBUTIONS

It is a recent trend in theoretical cryptography—inspired by seminal ideas of Micali and Reyzin [MR04]—to extend the realm of provable security in such a way that one can prove strong security guarantees for a cryptosystem, even in the presence of an adversary applying (an as large as possible class of) side-channels. This effort tries to close the gap between theory and practice in cryptography, providing formal security definitions better describing reality.

In the last few years a lot of results have been achieved in this sense. We will review some of them in Section 2.2 (in the case of leakage) and in Section 3.5 (in the case of tampering). The results in this thesis fall exactly in this stream. Moreover our purpose will always be to provide *efficient* constructions, so that they can be employed in the real world. In this way theory and practice can benefit from each other.

FIRST CONTRIBUTION. The first question we look at is a very general one. Consider a Boolean circuit C (just think of it as a smartcard) and imagine an adversary injecting faults into the circuit and thus maintaining black-box access to the faulty implementation. What kind of circuits can be protected? Which class of adversaries allows for positive results? In [FPV11], building on an earlier result of Ishai, Prabhakaran, Sahai and Wagner [Ish+06], we prove the following result:

Theorem 1 (Informal)**.** *It is possible to* efficiently *transform* any *Boolean circuit* C *into another circuit* $\widehat{C}$ *(with the same functionality) in such a way that* $\widehat{C}$ *is secure against an all-powerful adversary tampering with an unlimited number of wires, provided that: (i) every attack fails* independently *with some probability* $\delta \geqslant 0$*, (ii) the original circuit is still secure given a* logarithmic *amount of leakage on its secret state.*

Our result exhibits a general paradigm showing that it is possible to "trade" a small amount of leakage to achieve a strong form of tamper resilience. The good news is that—as we will see in a later chapter—there are cryptosystems tolerating up to a *constant* fraction of leakage on the secret state. Our proof—similarly to [Ish+06]—relies on the axiom that there exist *small*, *stateless* and *computation independent* tamper-proof components. (This means that the adversary is allowed to tamper only with the inputs and the outputs of these components, but not to touch their internals.) In essence, we reduce the problem of shielding arbitrarily complex circuits to the problem of shielding a few simple components. (Exactly two components in our transformation.)

SECOND CONTRIBUTION. Our second contribution shows that developing security definitions which better represent the real world, does not only help to fill the gap between theory and practice, but may also lead to new techniques of independent interest in other areas of cryptography. In particular, we will show a surprising connection between the area of tamper-resilience and the area of efficient authentication.

Starting from 2001, with the seminal work of Hopper and Blum [HB01], the problem of efficient authentication received a lot of attention. Besides relevant effort, an efficient round-optimal (i.e. with only 2 rounds) authentication protocol meeting the strongest notion of security (so called Man-in-the-Middle, MiM) was still missing.

The main ingredient that makes the HB-style family of protocols efficient is that security is based on a computational assumption which is very simple (and well studied): The Learning Parity with Noise (LPN) assumption. Roughly speaking, the LPN problem asks to distinguish several "noisy" inner products of a secret vector with random vectors from a truly random element. (Cf. Section 2.3 for a precise definition). All the protocols from this family only require to compute inner products of bit strings, and are thus very suitable for applications with bounded resources (e.g. RFID technologies). In essence, LPN-based authentication is able to provide a theoretical improvement in terms of provable security in addition to providing better efficiency than approaches based on more classical symmetric techniques that are not related to hard problems. Usually we trade one benefit for the other, but here we hope to get the best of both worlds.

Recently, Pietrzak [Pie10] has shown that LPN is robust against a powerful adversary tampering with the secret vector and the random vectors used in the definition of the LPN problem. This led to a variant of the LPN which is seemingly more general, but actually equivalent to the standard LPN problem. In [Kil+11], building on Pietrzak's result, we answer some open questions in the area of efficient authentication, developing a new family of protocols meeting MiM security. Our approach shows a completely different approach how to build an authentication protocol from LPN; in this sense our protocols are *non*-HB style.

Theorem 2. *(Informal) Under the LPN assumption there exist truly efficient round-optimal authentication protocols meeting MiM security.*

To achieve the above we provide the *first* practical construction of a MAC from LPN.

STRUCTURE. The structure of the thesis is as follows:

IN THE SECOND CHAPTER we review some of the basic tools of cryptography and some mathematical background on which we rely. This is almost for the sake of completeness. A reader experienced in the field can easily skip this chapter.

IN THE THIRD CHAPTER we start by giving an overview of how to apply tampering attacks in practice. We then give an overview of the results in [Ish+06] and explain the "trading leakage" paradigm. Finally we describe our circuit transformation.

IN THE FOURTH CHAPTER we start by describing Pietrzak's result in more detail. We thus show how to use it to get an efficient 2-round protocol achieving MiM security.

APPENDICES A–D contain some proofs missing from the main body. This is just to make the thesis self-contained.

2 PRELIMINARIES

«Manners are not taught in lessons,» said Alice. «Lessons teach you to do sums, and things of that sort.» «And you do Addition?» the White Queen asked. «What's one and one and one and one and one and one and one and one and one and one?» «I don't know,» said Alice. «I lost count.» «She can't do Addition,» the Red Queen interrupted.

LEWIS CARROLL, THROUGH THE LOOKING GLASS [CAR71]

CONTENTS

In this chapter, we take a detour to introduce the basic tools and techniques that will be used in the thesis. Some of the tools come, for instance, from probability theory and computational complexity. The material presented here is by far not complete and should be intended as a basic introduction mainly intended for the inexperienced reader. The experienced reader may in fact easily skip this part.

READER'S GUIDE. We introduce the basic notation in Section 2.1. Then, in Section 2.2, we recall the basic security definitions for the primitives used in the context of confidentiality and message integrity over an insecure channel. We also give a brief overview of the area of leakage-resilient cryptography; this quick overview is needed to understand the main result presented in Chapter 3. Section 2.3 introduces a class of computational assumptions with broad use in cryptography (and beyond), namely the class of "hard learning" problems; the computational assumptions introduced here are a basic building block for the

schemes presented in Chapter 4. We conclude this chapter, by recalling some basic inequality in probability theory in Section 2.4.

2.1 NOTATION

BASIC NOTATION. We let $\mathbb{N}$, $\mathbb{Z}$ and $\mathbb{R}$ denote (respectively) the set of natural numbers, the set of integers and the set of real numbers. Usually all the variables represented by the letters i, j, k, l, m, n are integers. If $n \geqslant 1$, then we write $[n]$ for $\{1, \ldots, n\}$. Given a real value $x \in \mathbb{R}$, we write $\lfloor x \rfloor$ (resp. $\lceil x \rceil$) for the smallest integer $n \in \mathbb{Z}$ such that $n \leqslant x$ (resp. $n \geqslant x$). The absolute value of $x \in \mathbb{R}$—denoted $|x|$—is x if x is positive and $-x$ otherwise. The absolute value satisfies the triangle inequality, namely for every $x, y \in \mathbb{R}$ we have $|x + y| \leqslant |x| + |y|$.

We always write $\log(\cdot)$ for the binary logarithm and $\ln(\cdot)$ for the natural logarithm.

Matrices and vectors are in boldface. Given a vector $\mathbf{r}$, we write $\mathbf{r}[i]$ for the element of $\mathbf{r}$ in position i. Vectors are intended as column vectors. The transpose of a vector is denoted T. (If $\mathbf{M}$ is a matrix, $\mathbf{M}[i]$ is the i-th column of $\mathbf{M}$.) Given two vectors $\mathbf{x}, \mathbf{y}$ of equal length, their inner product is $\mathbf{x}^{\mathsf{T}} \cdot \mathbf{y}$. Given two binary vectors $\mathbf{x}, \mathbf{y} \in \mathbb{Z}_2^n$ their Hamming distance, $0 \leqslant d_H(\mathbf{x}, \mathbf{y}) \leqslant n$, is the number of positions in which $\mathbf{x}$ and $\mathbf{y}$ differ. The Hamming weight $w_H(\mathbf{x})$ of the string $\mathbf{x} \in \mathbb{Z}_2^n$ is the number of 1's in $\mathbf{x}$, i.e. $w_H(\mathbf{x}) = d_H(\mathbf{x}, 0^n)$.

If $\mathcal{S}$ is a finite set, a string in $\mathcal{S}$ is an ordered tuple of elements in $\mathcal{S}$. We will mainly focus on $\mathcal{S} = \{0, 1\}$, i.e. the binary alphabet. For any integer $n \geqslant$, the set $\mathcal{S}^n$ is the set of length-n strings over $\mathcal{S}$. We write $\mathcal{S}^*$ for the set of all strings, i.e. $\mathcal{S}^* = \bigcup_{n \geqslant 0} \mathcal{S}^n$. If x and y are strings, $x \| y$ is the string obtained by concatenating x and y. If x is a string and $\kappa \geqslant 0$ is an integer, we write x^κ for the concatenation of κ copies of x. The length of x is $|x|$. When $\mathcal{S}$ is the binary alphabet, $x \oplus y$ is the xor of the strings $x, y \in \{0, 1\}$.

Events are denotes in small caps shape. If Event is an event, we write $\mathbb{P}[\text{Event}]$ for the probability of Event; the negation of Event is denoted $\neg$Event. If $\mathcal{S}$ is a set, we write $s \in \mathcal{S}$ for an element of $\mathcal{S}$. The cardinality of $\mathcal{S}$ is $\#\mathcal{S}$. If S is a distribution over a set $\mathcal{S}$, then $s \leftarrow S$ means a random variable s is drawn from S (if S has no distribution specified, then $s \xleftarrow{\$} S$ denotes a random variable with uniform distribution over S). If S is an algorithm, then $y \leftarrow S(x)$ means that y is the output of S on input x; in particular when S is probabilistic, y is a random variable. We write $\mathscr{A}^{\mathscr{O}(\cdot)}$ to denote an algorithm $\mathscr{A}$ with oracle access to $\mathscr{O}(\cdot)$.

RANDOM VARIABLES. Given a random variable X defined over domain $\mathcal{X}$, we write $\mathbb{P}[X = x]$ for the probability of X taking the value $x \in \mathcal{X}$. The expected value of X is $\mathbb{E}[X] = \sum_{x \in \mathcal{X}} x\mathbb{P}[X = x]$.

Given two random variables X, X' with finite domain $\mathcal{X}$, their statistical distance is $\Delta(X, X') = \sum_{x \in \mathcal{X}} |\mathbb{P}[X = x] - \mathbb{P}[X' = x]|$. We say X and X' are ϵ-close, if their statistical distance $\Delta(X, X')$ is at most ϵ.

The min-entropy of a random variable X is $H_\infty(X) = -\log(\max_x \mathbb{P}[X = x])$. Note that this implies $\max_x \mathbb{P}[X = x] = 2^{-H_\infty(X)}$.

ASYMPTOTIC NOTATION. The following notation will be very useful in the sequel.

Definition 2.1 (Asymptotic notation). Let $f, g : \mathbb{R} \to \mathbb{R}$ be two arbitrary functions. We say that:

(i) $f = O(g)$ if there is a constant $c > 0$ and a constant κ_0, such that for every $\kappa > \kappa_0$ we have $f(\kappa) \leqslant c \cdot g(\kappa)$. (Intuitively, f is bounded by g from above, up to a constant factor.)

(ii) $f = \Omega(g)$ if $g = O(f)$. (Intuitively g is bounded by f from above, up to a constant factor.)

(iii) $f = \Theta(g)$ if there are constants $c_1, c_2 > 0$ and a constant κ_0, such that for every $\kappa > \kappa_0$ we have $c_1 \cdot g(\kappa) \leqslant f(\kappa) \leqslant c_2 \cdot g(\kappa)$. (In other words, $f = O(g)$ and $g = O(f)$, that is f is bounded by g both from above and from below.)

(iv) $f = o(g)$ if *for every* $c > 0$ there is a constant κ_0 such that $|f(\kappa)| \leqslant c \cdot |g(\kappa)|$. (Intuitively, f is dominated by g asymptotically.)

(v) $f = \omega(g)$ if $g = o(f)$. (Intuitively, g is dominated by f asymptotically.)

We say a function $\mathrm{negl} : \mathbb{N} \to \mathbb{R}$ is negligible if it goes to zero faster than the inverse of any polynomial, that is for every $c > 0$ there exists a constant κ_0 such that for every $\kappa > \kappa_0$ we have $\mathrm{negl}(\kappa) < \kappa^{-c}$. A function $\mathrm{poly} : \mathbb{N} \to \mathbb{R}$ is polynomial if for every κ there exists a constant c such that $\mathrm{poly}(\kappa) < O(\kappa^c)$.

It is easy to check the validity of the following relations:

$$\mathrm{poly}(\kappa) + \mathrm{poly}(\kappa) = \mathrm{poly}(\kappa) \qquad \mathrm{poly}(\kappa) \cdot \mathrm{poly}(\kappa) = \mathrm{poly}(\kappa)$$
$$\mathrm{negl}(\kappa) + \mathrm{negl}(\kappa) = \mathrm{negl}(\kappa) \qquad \mathrm{negl}(\kappa) \cdot \mathrm{poly}(\kappa) = \mathrm{negl}(\kappa).$$

2.2 BASIC CRYPTOGRAPHY

As we have mentioned in Section 1.1, the strongest security guarantee for a cryptographic primitive is the notion of *unconditional security*, where there is no limit on the computational power of the adversary attacking the system. Even though there are schemes satisfying this strong notion (e.g., the well-known One-Time Pad), such schemes are usually inherently inefficient. This is the main reason why cryptographers stick to the weaker notion of *computational security*. The main idea is to somehow limit the computational power of the adversary attacking the system. This is possible by borrowing some ideas from the field of computational complexity.

COMPLEXITY THEORY & COMPUTATIONAL SECURITY. Computational complexity addresses the question of which problems are *efficiently* and *feasibly* computable. One might think that once we know that something is computable, it does not matter whether it takes 5 seconds or 10 seconds to compute. However, this conclusion would not be so obvious in the case of 5 seconds versus 5^{5^5} seconds! Such quantitative gaps are the main focus of complexity theory.

More precisely, complexity theory asks how the resources needed to solve a problem scale with some measure κ of the problem size: "reasonably" (like say κ or κ^2), or "unreasonably" (like 2^κ or $\kappa!$)? A good example is the case of integer multiplication. Whereas it is possible to multiply κ-digit numbers very efficiently (in $\approx \kappa^2$ computational steps using the grade-school method), the fastest algorithm to factor a κ-digit number into primes takes $\approx 2^{\kappa^{1/3}}$ computational steps.

Theoretical computer science calls an algorithm "efficient" (a.k.a. "polynomial-time") if its running time can be upper-bounded by any polynomial function of κ. On the other hand an algorithm is said "inefficient" if its running time can be lower-bounded by any exponential function of κ. Going further, the class **P** (Polynomial-Time) is, roughly speaking, the class of all computational problems that can be solved by a polynomial-time algorithm. The class **NP** (Nondeterministic Polynomial-Time) is the class of all computational problem for which a solution *can be verified* in polynomial-time, even though to find such a solution might be a very hard problem.[1] (Think again at the problem of integer factorisation).

1 Let us stress that **NP** does not stand for "Non-Polynomial". There are problems that require more than polynomial time, but the **NP** problems are not among those.

Clearly $\mathbf{P} \subseteq \mathbf{NP}$, since every problem solvable in polynomial-time is also verifiable in polynomial-time. However, whether the inclusion is strict it is not known and in particular the **P** vs. **NP** problem is one of the biggest open problems in mathematics.[2] If $\mathbf{P} = \mathbf{NP}$, then the ability to check the solutions to puzzles efficiently would imply the ability to find a solution efficiently. Since the above scenario is very unintuitive, complexity theorist (and cryptographers) proceed on the assumption that $\mathbf{P} \neq \mathbf{NP}$. This allows us to prove conditional statements, based on the hardness of certain computational problems (e.g., integer factorisation).

Because of this assumption, in all the security definition (in the context of computational security), the adversary is modelled as a PPT (Probabilistic Polynomial-Time) Turing Machine. We will not give a precise definition of Turing Machine, it suffices to know that a Turing machine is a theoretical device that manipulates symbols on a strip of tape according to a table of rules. Despite its simplicity, a Turing machine can be adapted to simulate the logic of any algorithm. A Turing Machine is PPT if it uses some randomness as part of its logic (i.e., it is probabilistic) and if its running time is polynomial in the length of its input.

SYMMETRIC ENCRYPTION. We now introduce the main cryptographic primitives to satisfy the requirements of confidentiality and message integrity (informally) introduced in Section 1.1. All the definitions will be given in the symmetric setting, where Alice and the Rabbit share a secret key K belonging to a set $\mathcal{K}$ of all possible keys.[3]

Suppose Alice wants to transmit over an insecure channel (controlled by the Red Queen) a message m belonging to a set $\mathcal{M}$ of all possible messages. To do so, she first applies to the message m a transformation (possibly depending on the secret key K) which turns the plaintext into a ciphertext c belonging to a space $\mathcal{C}$ of all possible encrypted messages. In the decryption process the plaintext m is recovered trough the secret key K and the ciphertext c. The formal definition follows.

Definition 2.2 (Symmetric encryption scheme). Let $\mathcal{M}$ be the space of all possible messages, $\mathcal{C}$ be the space of all possible ciphertexts and $\mathcal{K}$ be the space of all possible keys. A symmetric encryption scheme is a triple of PPT algorithms $\Pi_{\mathsf{SKE}} = (\mathsf{KGEN}, \mathsf{ENC}, \mathsf{DEC})$ defined as follows:

2 In fact this is one of the seven million-dollar Clay Millennium Prize Problems, see www.claymath.org/millennium/P_vs_NP/.

3 Of course such a key must be distributed in some way, but this is another issue.

- Key Generation: Algorithm KGEN takes as input the security parameter $\kappa \in \mathbb{N}$ (where κ represents the concrete security of the scheme), and outputs a key $\mathsf{K} \leftarrow \mathsf{KGEN}(1^\kappa)$—with $\mathsf{K} \in \mathcal{K}$—shared between Alice and the Rabbit.
- Encryption: Algorithm $\mathsf{ENC} : \mathcal{K} \times \mathcal{M} \to \mathcal{C}$ takes as input a message $m \in \mathcal{M}$ and a key $\mathsf{K} \in \mathcal{K}$ and outputs a ciphertext $c = \mathsf{ENC}_\mathsf{K}(m)$, with $c \in \mathcal{C}$.
- Decryption: Algorithm $\mathsf{DEC} : \mathcal{K} \times \mathcal{C} \to \mathcal{M}$ takes as input a ciphertext $c \in \mathcal{C}$ and a key $\mathsf{K} \in \mathcal{K}$ and outputs a plaintext $m = \mathsf{DEC}_\mathsf{K}(c)$, with $m \in \mathcal{M}$.

Note that usually an encryption scheme is randomised; we write this as $c \leftarrow \mathsf{ENC}_\mathsf{K}(m)$. Thus, encrypting a message m twice results in different ciphertexts (with overwhelming probability). We say that Π_{SKE} has completeness error α if for every $\kappa \in \mathbb{N}$, for every $\mathsf{K} \leftarrow \mathsf{KGEN}(1^\kappa)$ and for every $m \in \mathcal{M}$, we have

$$\mathbb{P}\left[m = \mathsf{DEC}_\mathsf{K}(c) : \; c \leftarrow \mathsf{ENC}_\mathsf{K}(m), \mathsf{K} \leftarrow \mathsf{KGEN}(1^\kappa), m \in \mathcal{M}\right] \geqslant 1 - \alpha.$$

The standard security notion for symmetric encryption is Indistinguishability against Chosen-Chiphertext Attacks (IND-CCA), where an adversary $\mathscr{A}$ attacking Π_{SKE} with respect to a key $\mathsf{K} \in \mathcal{K}$ is given oracle access to both $\mathsf{ENC}_\mathsf{K}(\cdot)$ and $\mathsf{DEC}_\mathsf{K}(\cdot)$. At some point the adversary chooses two messages $m_0, m_1 \in \mathcal{M}$ with the same length and is given an encryption with key K of one of the two messages. The goal of the adversary is to guess which one. This is formalised trough the following experiment.

Experiment $\mathbf{Exp}^{\mathrm{indcca}}_{\Pi_{\mathsf{SKE}}}(\mathscr{A}, \kappa)$

1. $\mathsf{K} \leftarrow \mathsf{KGEN}(1^\kappa)$
2. $m_0, m_1 \leftarrow \mathscr{A}^{\mathsf{ENC}_\mathsf{K}(\cdot), \mathsf{DEC}_\mathsf{K}(\cdot)}(1^\kappa)$
3. $b \xleftarrow{\$} \{0,1\}$, $c_b \leftarrow \mathsf{ENC}_\mathsf{K}(m_b)$
4. $b' \leftarrow \mathscr{A}^{\mathsf{ENC}_\mathsf{K}(\cdot), \mathsf{DEC}_\mathsf{K}(\cdot)}(c_b)$
5. The experiment outputs 1 if and only if (i) $b' = b$, (ii) $|m_0| = |m_1|$ and (iii) c_b is never asked to decryption oracle $\mathsf{DEC}_\mathsf{K}(\cdot)$

(A weaker definition is Indistinguishability in the presence of Chosen-Plaintext Attacks—IND-CPA—where the adversary is not allowed to ask decryption queries.)

We say that Π_{SKE} is (t, Q, ϵ)-IND-CCA-secure if for any adversary $\mathscr{A}$ running in time t and asking a total of Q queries, we have

$$\mathbb{P}\left[\mathbf{Exp}^{\mathrm{indcca}}_{\Pi_{\mathsf{SKE}}}(\mathscr{A}, \kappa) = 1\right] \leqslant \epsilon.$$

MESSAGE AUTHENTICATION. Another basic security requirement is the one of message integrity. Intuitively, here Alice would like to ensure that the message m she sends over the channel reaches the Rabbit unchanged. To do so, she uses the shared secret key to compute a tag ϕ on the message m and sends the pair (m, ϕ). The main idea is that the Rabbit can now verify if the pair (m, ϕ) is valid trough the secret key K.

Definition 2.3 (Message authentication code). Let $\mathcal{M}$ be the space of all possible messages, Φ be the space of all possible tags and $\mathcal{K}$ be the space of all possible keys. A message authentication code (MAC) is a triple of PPT algorithms $\Pi_{MAC} = (KGEN, TAG, VRFY)$ defined as follows:

- Key Generation: Algorithm KGEN takes as input the security parameter $\kappa \in \mathbb{N}$ (where κ represents the concrete security of the scheme), and outputs a key $K \leftarrow KGEN(1^\kappa)$ — with $K \in \mathcal{K}$ — shared between Alice and the Rabbit.
- Tagging: Algorithm $TAG : \mathcal{K} \times \mathcal{M} \to \Phi$ takes as input a message $m \in \mathcal{M}$ and a key $K \in \mathcal{K}$ and outputs a tag $\phi \leftarrow TAG_K(m)$, with $\phi \in \Phi$.
- Verification: Algorithm $VRFY : \mathcal{K} \times \mathcal{M} \times \Phi \to \{accept, reject\}$ takes as input a pair $(m, \phi) \in \mathcal{M} \times \Phi$ and a key $K \in \mathcal{K}$ and outputs a value $d \leftarrow VRFY_K(m, \phi)$, with $d \in \{accept, reject\}$.

Note that when the verification algorithm is deterministic, it outputs 1 if and only if $\phi = TAG_K(m)$. Otherwise VRFY must be defined explicitly. We say that Π_{MAC} has completeness error α if for every $\kappa \in \mathbb{N}$, for every $K \leftarrow KGEN(1^\kappa)$ and for every $m \in \mathcal{M}$, we have

$$\mathbb{P}\left[VRFY_K(m, \phi) = reject : \ \phi \leftarrow TAG_K(m), K \leftarrow KGEN(1^\kappa), m \in \mathcal{M}\right] \geqslant 1 - \alpha.$$

The standard security notion for symmetric encryption is universal unforgeability against chosen meessage attacks (ufcma), where an adversary $\mathscr{A}$ attacking Π_{MAC} with respect to a key $K \in \mathcal{K}$ is given oracle access to both $TAG_K(\cdot)$ and $VRFY_K(\cdot, \cdot)$. At some point the adversary outputs a pair (m^*, ϕ^*) and she wins if ϕ^* is a valid tag for m^*, and the message m^* is fresh, i.e. it has never been queried to the $TAG_K(\cdot)$ oracle. This is formalised trough the following experiment.

Experiment $\mathbf{Exp}^{ufcma}_{\Pi_{MAC}}(\mathscr{A}, \kappa)$

1. $K \leftarrow KGEN(1^\kappa)$

2. $(m^*, \phi^*) \leftarrow \mathscr{A}^{\mathsf{TAG}_K(\cdot),\mathsf{VRFY}_K(\cdot,\cdot)}(1^\kappa)$
3. The experiment outputs 1 if and only if (i) $\mathsf{VRFY}_K(m^*, \phi^*) = \mathsf{accept}$ and (ii) m^* is never asked to the $\mathsf{TAG}_K(\cdot)$ oracle

We say that Π_{MAC} is (t, Q, ϵ)-ufcma-secure if for any adversary $\mathscr{A}$ running in time t and asking a total of Q queries, we have

$$\mathbb{P}\left[\mathbf{Exp}^{\mathrm{ufcma}}_{\Pi_{\mathsf{MAC}}}(\mathscr{A}, \kappa) = 1\right] \leqslant \epsilon.$$

ASYMMETRIC TECHNIQUES. The main problem in the context of symmetric cryptography is key distribution: every pair of users in the system must share some key $K \in \mathcal{K}$. How to distribute such a key is not discussed further here. (But see for instance [NPR99; CKS05]). An alternative setup is the public key setting, introduced for the first time by Diffie and Hellman [DH76]. Even though we will not focus much on public key cryptography, we sketch briefly the main ideas below.

The idea is to let every party hold *two* keys: say Alice has (pk_A, sk_B) and the Rabbit has (pk_B, sk_B). The two keys (pk_A, pk_B) are public, whereas (sk_A, sk_B) must kept secret. In this context, when Alice wants to send a message $m \in \mathcal{M}$ to the Rabbit over an insecure channel, she will encrypt the message using pk_B. The Rabbit can thus recover m using its secret key sk_B. (The RSA cryptosystem sketched in Section 1.1 is an example of an encryption scheme in the public key setting.) In a similar fashion, to guarantee integrity of a message $m \in \mathcal{M}$, Alice will create a "digital signature" σ on m using her secret key sk_A. Everybody having the corresponding public key pk_A can thus verify the validity of the pair (m, σ).[4]

Note that in this case the parties do not need to share any a priori secret, so that the problem of key distribution is inherently solved. However, others set-up assumption are needed, since, for example, Alice needs to be sure that the Rabbit's public key is authentic (see for instance [Bol+07]).

LEAKAGE RESILIENCE. As we have mentioned in the introduction, side-channel attacks are a powerful means against (otherwise "provably" secure) cryptosystems. In the passive case, an adversary can leak information on the secret key by observing some characteristic of an implementation of a given primitive.

4 Note that in this sense digital signatures are transferable (whereas MACs are not). In fact, given a valid tag (m, ϕ) only a party knowing the shared key K can verify the MAC.

A large and growing body of research — in the so called area of leakage resilient cryptography — is trying to fill this gap, by providing cryptographic systems which remain secure even in the presence of (passive) physical attacks. The main idea is to "ease" the goal of an adversary attacking a cryptographic primitive, by giving her access to a leakage oracle revealing some information on the secret key. Roughly, such a leakage oracle can be queried on input a function f and outputs the result of this function applied to the secret key. The first ideas can be found in some papers on exposure-resilient cryptography [Can+00; DSS01; ISW03], where one considers simple leakage functions that reveal just a subset of the bits of the secret key. In this context, Ishai, Sahai and Wagner [ISW03] show how to "compile" any cryptographic algorithm into one that tolerates such types of leakage. In contrast to these works, that consider leakage functions acting locally, the focus of later works has been on more powerful leakage functions that can perform some global computation on the secret key. (As it will be clear in a moment, some restriction on the nature of the leakage functions is necessary.) The work of Micali and Reyzin on "physically observable" cryptography [MR04] was the first proposing to study formal models that capture general types of leakage. This study has led to several models allowing for positive results, described below.

- *Bounded-leakage.* In this model the adversary can submit to the leakage oracle functions f_i, as long f_i is polynomially computable. However, the range of every function is bounded by a parameter λ_i and the *total* amount of information learned by the adversary is bounded. In particular, for a secret key K, it must be $\sum_i \lambda_i = \lambda < L = |K|$. Note that the functions f_i can be chosen adaptively. There are both encryption schemes [AGV09; NS09; BG10; Dod+10a; Cho+10] and signature schemes [KV09] tolerating up to $\lambda = (1 - o(1)) \cdot L$ bits of leakage.

- *Continual-leakage.* The main limitation of the bounded-leakage model, is that, over time, it is not very reasonable to assume a bound on the leakage. To overcome this problem, some papers [Dod+10a; Bra+10] allow the secret key of the system to be refreshed (and the old one erased), while the public key remains fixed (i.e., those who try to encrypt need not be aware of the refreshes). The bound on the amount leakage, instead of being the overall one throughout the lifetime of the system, is only between refreshes. There is no bound on the total amount of information that can be leaked during the lifetime of the system. In this model, it is possible to construct encryption schemes and signature schemes tolerating up to $(1 - o(1)) \cdot L$

bits of leakage between each refresh and a logarithmic (or even super-logarithmic [LLW11]) amount of leakage during the refreshing phase.

- *Noisy leakage.* Another option (used for instance in [NS09; Fau+10b]) is to ask that the leakage is not of bounded length, but it is guaranteed that the secret key is still unpredictable given the leakage. This motivates a realistic generalization that allows the adversary to learn any random variable (representing the leakage information) that does not decrease the min-entropy of the secret key too much.

- *The auxiliary input model.* A more general model is obtained by only assuming that the secret key cannot be efficiently recovered given the leakage. This approach was put forward by Dodis, Tauman Kalai, and Lovett [DKL09] who studied the security of symmetric-key encryption schemes in the presence of leakage of the form $f(K)$, where K is the secret key and f is any exponentially-hard one-way function. Public key encryption schemes are also known [Dod+10b].

- *Other restrictions.* Other (less general) models require that "only computation leaks information" [DP08; Pie09; Fau+10a; JV10; GR10] (i.e. only the active part of the memory can leak), that the momory is diveded into two parts and each of these parts leak individually [DDV10; KP10; DF11; Dod+11], or that the leakage is computed by a space-bounded function [DKW11b; DKW11a].

2.3 HARD LEARNING PROBLEMS

In this section we introduce a class of computational problems with several applications in cryptography and machine learning. Fix a string $\mathbf{x} \in \mathbb{Z}_2^\kappa$. Given a sequence of randomly chosen vectors $\mathbf{r}_1, \ldots, \mathbf{r}_n$, along with the inner products $z_i = \mathbf{r}^\mathsf{T} \cdot \mathbf{x}$, it is an easy task to recover $\mathbf{x}$ by Gaussian elimination. (In fact, polynomially many samples are sufficient to have κ linearly independent equations in $\mathbf{x}[i]$.) In the presence of noise, i.e. when each value z_i is flipped independently with probability $0 < \tau < 1/2$, the problem above is much more complicated and is called the learning parity with noise (LPN) problem.

Let $\mathbf{x} \in \mathbb{Z}_2^\kappa$ be a random secret and denote with Bern_τ the Bernoulli distribution with parameter $0 < \tau < 1/2$ (i.e. $\mathbb{P}[e] = \tau$ if $e \xleftarrow{\$} \mathsf{Bern}_\tau$). We define $\Lambda_{\tau,\kappa}(\mathbf{x})$ to be the distribution over $\mathbb{Z}_2^{\kappa+1}$ obtained by choosing $\mathbf{r} \xleftarrow{\$} \mathbb{Z}_2^\kappa$, $e \xleftarrow{\$} \mathsf{Bern}_\tau$ and re-

turning $(\mathbf{r}, \mathbf{r}^\intercal \cdot \mathbf{x} \oplus e)$. The (computational) $\mathsf{LPN}_{\tau,\kappa}$ problem requires to compute $\mathbf{x}$ given access to $\Lambda_{\tau,\kappa}(\mathbf{x})$. The (decisional) $\mathsf{LPN}_{\tau,\kappa}$ problem requires to distinguish $\Lambda_{\tau,\kappa}(\mathbf{x})$ from the uniform distribution $\mathcal{U}_{\kappa+1}$ over $\mathbb{Z}_2^{\kappa+1}$.

We say that the LPN problem is (t, Q, ϵ)-hard if for all distinguishers $\mathscr{D}$ running in time t and asking Q oracle queries, we have

$$\left| \mathbb{P}\left[\mathscr{D}^{\Lambda_{\tau,\kappa}(\mathbf{x})} = 1 : \mathbf{x} \xleftarrow{\$} \mathbb{Z}_2^\kappa \right] - \mathbb{P}\left[\mathscr{D}^{\mathcal{U}_{\kappa+1}} = 1 \right] \right| \leqslant \epsilon.$$

In what follows, we will refer only to the decisional version of the problem. In fact, the two versions are known to be equivalent, as shown by the following lemma.

Lemma 2.1 ([KS06; KSS10], Lemma 1). *Say there exists an algorithm $\mathscr{D}$ making Q oracle queries, running in time t, and such that*

$$\left| \mathbb{P}\left[\mathscr{D}^{\Lambda_{\tau,\kappa}(\mathbf{x})} = 1 : \mathbf{x} \xleftarrow{\$} \mathbb{Z}_2^\kappa \right] - \mathbb{P}\left[\mathscr{D}^{\mathcal{U}_{\kappa+1}} = 1 \right] \right| \geqslant \epsilon.$$

Then there exists an algorithm $\mathscr{A}$ making $Q' = O(Q \cdot \epsilon^{-2} \log \kappa)$ queries, running in time $t' = O(t \cdot \kappa \cdot \epsilon^{-2} \log \kappa)$, and such that

$$\mathbb{P}\left[\mathscr{A}^{\Lambda_{\tau,\kappa}(\mathbf{x})} = \mathbf{x} : \mathbf{x} \xleftarrow{\$} \mathbb{Z}_2^\kappa \right] \geqslant \epsilon/4.$$

The best (known) algorithms for the LPN problem take exponential time $2^{\kappa/\log\kappa}$ when $\tau > 0$ is treated as a constant [BKW00; BKW03; LF06].

LEARNING WITH ERRORS. A natural generalization of the above problem is the learning with errors (LWE) problem [Reg05]. The most appealing characteristic of this problem is that it enjoys for certain parameters a worst-case hardness guarantee [Reg05; Pei09].[5] We informally recall the LWE problem below. Let $q \geqslant 2$ be a prime and denote with $\mathsf{Gau}_{q,\tau}$ the so called "discretised normal error" distribution parametrised by some $\tau \in]0, 1[$. This distribution is obtained by drawing $x \in \mathbb{R}$ from the Gaussian distribution of width τ (i.e., x is chosen with probability $\frac{1}{\tau}\exp(-\pi x^2/\tau^2)$) and outputting $\mathit{round}(q \cdot x) \bmod q$. For a random secret $\mathbf{s} \in \mathbb{Z}_q^\kappa$, the (decisional) $\mathsf{LWE}_{q,\tau,\kappa}$ problem is to distinguish samples of the form $(\mathbf{r}, \mathbf{r}^\intercal \cdot \mathbf{s} + e)$ from uniformly random samples in $\mathbb{Z}_q^\kappa \times \mathbb{Z}_q$, where $\mathbf{r} \xleftarrow{\$} \mathbb{Z}_q^\kappa$, $e \xleftarrow{\$} \mathsf{Gau}_{q,\tau}$ and all the operations are performed modulo q.

5 It is a well establish fact that cryptography requires problems that are hard to solve on the average, so that when a cryptographic key is chosen at random, the corresponding function is hard to break with high probability. Before a famous result of Ajtai and Dwork [AD97], all known cryptographic functions almost invariably rely on average-case complexity assumptions. In this respect, lattice problems are exceptional in their ability to provide provably secure cryptographic functions from worst-case complexity assumptions.

2.4 TAIL INEQUALITIES

Tail inequalities are a powerful tool in probability theory, with several applications in cryptography and beyond. Intuitively, these inequalities bound the probability that a random variable with a bell-shaped distribution takes a value in the tails of the distribution, far away from the mean.

Perhaps the simplest tail inequality was named after the Russian mathematician Andrey Markov.

Lemma 2.2 (Markov's inequality). *For every positive random variable X, we have*

$$\mathbb{P}[X \geqslant n] \leqslant \frac{\mathbb{E}[X]}{n}.$$

Proof. Note that

$$\begin{aligned}\mathbb{E}[X] = \sum_{x \geqslant 0} \mathbb{P}[X = x]\, x &\geqslant \sum_{0 \leqslant x < n} \mathbb{P}[X = x] \cdot 0 + \sum_{x \geqslant n} \mathbb{P}[X = x] \cdot n \\ &= \mathbb{P}[X \geqslant n] \cdot n.\end{aligned}$$

□

The above inequality can be generalised to the case where the random variable X is the sum of n independent random variables. This yields the so called Chernoff bound, which will be used repeatedly in Chapter 4.

Theorem 2.3 (Chernoff bounds). *Let $X_1, X_2, \ldots, X_n$ be independent random variables over $\{0,1\}$ and let $\mu = \sum_{i=1}^{n} \mathbb{E}[X_i]$. We have:*

(⋆) $\forall\, \delta > 0$

$$\mathbb{P}\left[\sum_{i=1}^{n} X_i \geqslant (1+\delta)\mu\right] \leqslant \left(\frac{e^{\delta}}{(1+\delta)^{1+\delta}}\right)^{\mu}$$

(⋆⋆) $\forall\, 0 < \delta \leqslant 1$

$$\mathbb{P}\left[\sum_{i=1}^{n} X_i \geqslant (1+\delta)\mu\right] \leqslant e^{-\mu\delta^2/3}$$

(⋆⋆⋆) $\forall\, c \geqslant 6\mu$

$$\mathbb{P}\left[\sum_{i=1}^{n} X_i \geqslant c\right] \leqslant 2^{-c}.$$

Proof. See Appendix A. □

3 TAMPER-PROOF CIRCUITS

«Be what you would seem to be»—or if you'd like it put more simply—«Never imagine yourself not to be otherwise than what it might appear to others that what you were or might have been was not otherwise than what you had been would have appeared to them to be otherwise.»

LEWIS CAROL, ALICE'S ADVENTURES IN WONDERLAND [CAR65]

CONTENTS

We have already seen the devastating power of some tampering attacks against RSA, in Chapter 1. In this chapter we will study tampering from a much more general point of view, referring to arbitrary Boolean circuits and not to specific constructions.

Our goal is to develop new countermeasures using the techniques of the realm of provable security. In a nutshell we would like to address the following question.

> *Q: Is it possible to design an* efficient *procedure to transform* any *circuit into another one (with the same input-output behaviour) provably resisting a large (and meaningful) class of tampering attacks?*

READER'S GUIDE. The take-home message from years of experience in the area of provable security is that for any theoretical model to be meaningful, it has to describe reality quite precisely. Following this lesson, we start Section 3.1 by describing how tampering attacks are applied in the real world. This will help us to understand the power of the adversaries we need to consider. Then, in Section 3.2 we describe the result of Ishai, Prabhakaran, Sahai and Wagner [Ish+06]

and its limitations. To overcome these limitations, in Section 3.3 we introduce a new framework for defining tamper-resilience and describe our transformation. The security proof is given in Section 3.4. Some extensions and open problems are finally considered in Section 3.5.

3.1 TAMPERING IN PRACTICE

Cryptographic devices are usually made of two components: a *digital circuit* carrying out the required functionality (e.g. signing) and *memory cells* that contain the data to be processed and the secret key. Digital circuits are built from logical cells, the smallest atomic unit in the circuit. They operate on Boolean values (i.e. the values in $0, 1$) and compute basic logical operations such as NOT, NAND and so on.

In practice we need a representation for the Boolean values $0, 1$. The value 1 is typically defined by the voltage supply level V_{DD}; the value 0 is defined by the ground voltage level V_{SS}.

CMOS TECHNOLOGY. Logical cells are built from transistors. A transistor is a semiconductor[1] device used to amplify and switch electronic signals. There are different types of transistors; however, today most of the digital integrated circuit are built using *Field Effect Transistor* (FET) in the so-called *Complementary Metal-Oxide-Semiconductor* (CMOS) technology. The words "complementary-symmetry" refer to the fact that the typical digital design style with CMOS uses complementary and symmetrical pairs of p-type and n-type metal oxide semiconductor FETs for building logical functions. The main attractive of CMOS technology is that it has high noise immunity and low static power consumption.

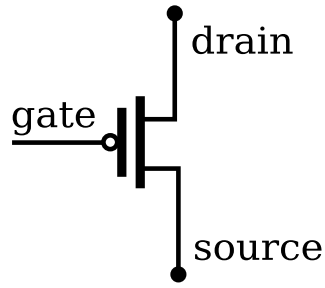

Figure 3.1: NMOS.

Figure 3.1 shows an NMOS transistor: A small voltage at the gate terminal can control or switch a current between the source and drain terminals. (A PMOS transistor is identical, but without the small empty circle at the end of the gate.) Applying a low level voltage (i.e. a 0) to the gate of a PMOS transistor, generates a low resistance between its source and drain contacts (thus current can flow)

1 A semiconductor is a material that is neither fully isolating (like plastic) nor fully conducting (like metal). Silicon is by far the most commonly used semiconductor material.

and viceversa. On the other hand, a low level voltage (i.e. a 0) to the gate of an NMOS transistor, generates a high resistance between its source and drain contacts and viceversa.

The way CMOS circuits are constructed is such that a PMOS transistor always has either an input from the power supply or from another PMOS transistor. In a similar fashion, all NMOS transistors must have either an input from the ground or from another NMOS transistor. Current reduction is accomplished by complementing every NMOS with a PMOS and connecting both gates and both drains together. In this way, a high voltage on the gates will cause the NMOS to conduct and the PMOS not to conduct while a low voltage on the gates causes the reverse. This arrangement greatly reduces power consumption and heat generation.

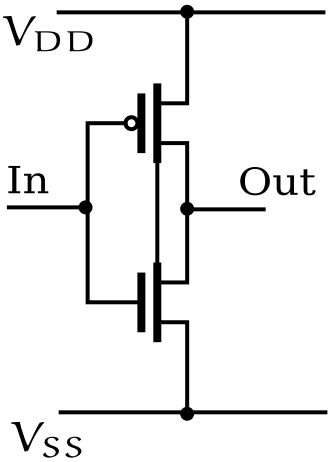

Figure 3.2: CMOS inverter.

Figure 3.2 shows a NOT gate (i.e. an inverter) in CMOS logic. When the input is a 0, the NMOS is not conducting. On the other hand, the PMOS transistor is conducting, resulting in a path from the voltage level to the output, and setting the output to 1. The situation is reversed when the input is a 1: Only the NMOS is conducting in this case, resulting in a path from the output to the ground level and setting the output to 0. In short, the outputs of the PMOS and NMOS transistors are complementary such that when the input is low, the output is high, and when the input is high, the output is low. Because of this behavior of input and output, the CMOS circuits' output is the inversion of the input. The other Boolean gates can be constructed in a similar way, see [Bak10] for details.

MEMORY CELLS. Until now we have only focused on computation. One of the most common technology for storing and managing data is *Static Random Access Memory* (SRAM), shown in CMOS style in Figure 3.3. Each bit in an SRAM is stored using two inverters. The word line (WL) controls the two access transistors T5 and T6. In turn the access transistors control whether the cell should be connected to the bit lines: BL and $\overline{\text{BL}}$. The latter are used to transfer for read/write operations.

An SRAM can be in one of 3 different states: *standby*, *reading* and *writing*. When the word line is not asserted, the cell is isolated from the bit lines and the

SRAM is in standby. Until the power supply is applied the two inverters will continue to store one of the two *stable states*: Either $S = 0$ and $\overline{S} = 1$ (i.e. the cell is storing 0) or $S = 1$ and $\overline{S} = 0$ (i.e. the cell is storing 1). The two inverters configuration is called *flip-flop* and it is widely used in digital electronics.

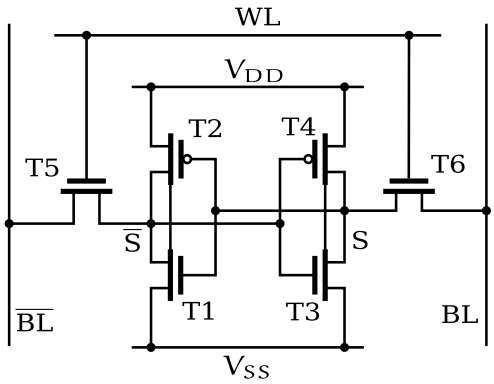

Figure 3.3: SRAM in CMOS style.

To write a value on the memory (say a 1), the value 1 is applied to BL and the opposite is applied to $\overline{BL}$ and the word line is enabled. This will cause the flip-flop to change state, overwriting the previous value.

To read a value, both the bit lines are precharged to 1. If the memory content is 0 (resp. 1), BL will be pulled toward 0 (resp. 1) and $\overline{BL}$ toward 1 (resp. 0).

FAULT INJECTION. By *fault attack* (a.k.a. *tampering attack*), we mean any method/approach/algorithm, which is given a device and which returns secret data. During the attack, an adversary may run the target device several times while inducing faults into memory cells or other structural elements. These faults are induced by some physical attack, exposing the machine to some sort of physical stress. As a reaction, the device malfunctions (e.g., memory cells change their current or structural elements are damaged). All these effect will be referred to as faults. We say that a fault attack induces faults into a device, such that the output of the attacked device is *faulty*, i.e. it has a value different than expected.

The most common techniques to mount a physical attack (and their effects) are described below.

- *High-energy radiation.* Cosmic rays are very high-energy subatomic particles originating in outer space. They are comparable to high-energy protons and neutrons produced by large particle accelerators. It has been experimented that cosmic rays can cause a bit flip [GA03]. However, since the equipment needed to simulate cosmic radiation is not available to the public, to exploit this source an attacker has to wait that a single bit is flipped by a cosmic ray by chance. This is very unlikely.

Other methods of inducing a bit flip exists based on α-particles, β-particles and X-ray radiation. However all these methods are not very effective and difficult to use with standard commercial equipment. Thus, we will not say more about them. The interested reader can refer to [GA03].

- *Heat.* Every electronic device has a range of temperatures under which it works reliably; forcing the device to work outside this range of temperatures can induce faults. A simple attack of this kind against a standard PC has been shown in [GA03] using just a 50 Watt spotlight clip-on lamp. The problem with this approach is that it is difficult to target a single bit with high precision. Moreover, when not finely tuned, the attack can cause the operating system to crash, requiring a complete re-installation.

  The use of heat to inject faults into a device dates back to earlier works on tampering attacks, see [BDL97; Pet97].

- *Optical attacks.* These attacks need the smartcard to be unpacked, i.e. it is necessary that the silicon layer is visible. In this case it is possible to use UV light to destroy individual structures of the chip [KK99].

  However, also non-destructive attacks are possible. The physical principle exploited here, is that semiconductor transistors are sensitive to coherent light (e.g. the light emitted by a laser) in a way similar to the fact they are sensitive to cosmic radiation. Relying on this principle, a simple attack is shown in [SA02] using an intense light source (namely a photoflash lamp), a microscope and aluminium foil. The targeted device is an SRAM cell (cf. Figure 3.3). If the transistor T3 is opened for a very short time by an external stimulus, then it could cause the flip-flop to change state. By exposing the transistor T4 , the state of the cell would be changed to the opposite. Thus one is allowed to set or unset individual chosen bits of the memory.

- *Eddy currents.* It has been known for a long time [AK97; Pet97; KK99] that placing a device in an electromagnetic field may influence the transistors and memory cells behaviour, inducing faults. A more recent attack [QS02], showed that it is possible to generate a magnetic field inducing Eddy currents on the surface of a conducting materials. Eddy currents are present in everyday life and used for a variety of applications such as for the purpose of measurements and for melting materials.

  It is possible to show that Eddy currents can modify the number of electrons inside a transistor's oxide grid. This changes the threshold voltage

of the transistor, such that it cannot be switched anymore. Depending on the actual transistor, this can be used to ensure that a memory cell contains the value 0 or 1.

TAMPERING IN WONDERLAND. To develop a good theoretical model, it is important to abstract away common characteristics of fault attacks. A good characterisation has been proposed in [Ott04]. On a high level, tampering attacks differ in the power to locate and time the induced faults, in the number of bits affected, in the effect of an attack, in the probability of the implied effect of an induced fault and in the duration of an effect. This is specified more carefully below:

- *Location.* Every attack has a certain precision. Some attacks are able to target selected bits very precisely.
- *Number of bits.* Tampering can effect a single bits or more than one bit.
- *Fault type.* A single bit can be *set* to 1, *reset* to 0 or *flipped* (meaning that the value on the targeted wire is changed, whatever the initial value was).
- *Probability.* Usually an attack is not guaranteed to be successful, it is only so most of the time.
- *Duration.* We differentiate between *transient* faults, *permanent* faults and *destructive* faults. Destructive faults cannot be reversed and occur when the adversary destroys part of the physical structure on the chip. On the other hand, permanent and transient faults do not modify the hardware of the device, thus allowing the device to recover from the induced faults after a certain period of time. Permanent faults change an affected variable until that variable is explicitly overwritten, whereas transient faults are faults where after a given amount of time, the effect ceases to exist and the correct value is present again.

We start with an *arbitrary* (probabilistic) Boolean circuit C; a Boolean circuit C is a directed acyclic graph whose vertices are standard Boolean gates and whose edges are the wires. The depth of C, denoted $\mathit{depth}(C)$, is the longest path from an input to an output. We say a circuit is *clocked* if it evolves in clock cycles (or rounds). The input and output values of the circuit C in clock cycle i are denoted by X_i and Y_i, respectively. A circuit is *probabilistic* if it uses internal randomness as part of its logic. We call such probabilistic logic randomness gates and denote them with \$. In each clock cycle \$ outputs a fresh random bit.

As we have seen, additionally, a circuit may contain memory gates. Memory gates, which have a single incoming edge and any number of outgoing edges, maintain state: at any clock cycle, a memory gate sends its current state down its outgoing edges and updates it according to the value of its incoming edge. Any cycle in the circuit graph must contain at least one memory gate. The state of all memory gates at clock cycle i is denoted by M_i, with M_0 denoting the initial state. When a circuit is run in state M_{i-1} on input X_i, the circuit will output Y_i and the memory gates will be in a new state M_i. We will denote this by $(Y_i, M_i) \leftarrow C[M_{i-1}](X_i)$.

Given any (probabilistic) Boolean circuit, we will consider adversaries that can adaptively tamper in Q clock cycles with up to t wires (for some constant $t \in \mathbb{N}$). We are particularly interested in the case where t is unbounded, i.e. the adversary can tamper with an arbitrarily large number of wires in the circuit in every round. For each wire we allow the adversary to choose between the following types of attacks: *set*, i.e. setting a wire to 1, *reset*, i.e. setting a wire to 0 and *toggle*, i.e. flipping the value on the wire. For each wire such an attack fails *independently* with some probability. This is captured by the global parameter δ, where $\delta = 0$ means that the attack succeeds always, and $\delta = 1$ that no tampering takes place. When an attack fails for one wire the computation continues with the original value on that wire. Note that the adversary can tamper the same wire several times, but only once in-between every two invocations. However, as tampering is persistent, after a sufficiently large number of attempts the tampering will succeed almost certainly, i.e. with probability $1 - \delta^l$ after l rounds.

Notice that once a fault is successfully placed it stays permanently. Let us stress that we do allow the adversary to "undo" (with zero error probability) persistent attacks induced in previous rounds (this captures so called transient faults).

We call such an adversary, that can adaptively tamper with a circuit for up to Q clock cycles attacking up to t wires per round, an (t, δ, Q)-adversary and denote the attack strategy for each clock cycle as $\mathcal{W} = \{(w_1, a_1), ..., (w_t, a_t)\}$. The first element in each such tuple specifies which wire in the circuit is attacked and the second element specifies the type of attack (i.e. *set*, *reset* or *toggle*). When the number of faults per clock cycle is unbounded, we will explicitly write $t = \infty$.

3.2 THE RESULT OF ISHAI ET AL.

The question whether an arbitrary (probabilistic) Boolean circuit C can be transformed in such a way that a tampering adversary is defeated with high probability has been addressed for the first time in [Ish+06]. Since our transformation will be based on several ideas of Ishai *et al.*, in this section we take a detour and summarise their result.

The main idea is to rely on a *circuit compiler* Ψ. Such a compiler is just a set of rules how to transform any Boolean circuit C into another (eventually bigger) circuit $\widehat{C}$. Notice that these rules basically need to specify how to transform: (i) The values carried by the wires in C and (ii) the gates in C. (Without loss of generality the NAND gate suffices, since it is well known that NAND is universal.)

More formally, a circuit transformation Ψ takes as input a security parameter κ, a (probabilistic) circuit C and an initial state M_0 and produces a transformed initial state $\widehat{M}_0$ and a transformed (probabilistic) circuit $\widehat{C}$. This is denoted by $(\widehat{C}, \widehat{M}_0) \leftarrow \Psi(C, M_0)$. Let us stress that the transformation itself can be randomised and we let r_Ψ denote the random coins of the transformation.

The first (natural) requirement we want from Ψ is that $\widehat{C}$ maintains the same functionality as the original circuit C: For all C, M_0 and any set of public inputs $X_1, X_2, \ldots, X_Q$ the original circuit C starting with state M_0 and the transformed circuit $\widehat{C}$ starting with state $\widehat{M}_0$ result in an identical output distribution. In this case we say that Ψ is *functionality preserving*. The second requirement we want from Ψ is that it should have some form of tamper-resilience, as discussed below.

THE DEFINITION OF ISHAI ET AL. The model considered by [Ish+06] takes only care of $(t, 0, Q)$-adversaries. Recall this means that the adversary $\mathscr{A}$ is allowed to *set/reset/toggle* the value of up to t wires in the circuit (for some small constant $t \in \mathbb{N}$, so $t \neq \infty$) but attacks are always successful (i.e. $\delta = 0$). The way Ishai *et al.* define tamper-resilience of Ψ is via simulation.[2] The basic idea is to compare two different "worlds". The first one is the real world where the attack against $\widehat{C}$ takes place; thus the first world features an $(t, 0, Q)$-adversary $\mathscr{A}$ tampering with $\widehat{C}$. The second world features a simulator $\mathscr{S}$ having just black-box access to the original circuit C; thus $\mathscr{S}$ applies inputs to C and learns the corresponding outputs, without injecting any faults. Informally, Ψ is tamper-resilient if whatever $\mathscr{A}$ learns in the first world, is in some sense "equivalent" to

2 Simulation-based definitions are a very well-known paradigm how to define security of some cryptographic primitives, e.g. zero-knowledge proof systems [GMR85].

what the simulator $\mathscr{S}$ learns in the second world: This essentially means that tampering is "useless", since what $\mathscr{A}$ can learn by tampering with $\widehat{C}$ can be perfectly "simulated" having only black-box access to the original circuit C.

To make this idea formal we need to introduce two experiments describing what happens in the two worlds sketched above and then to compare the output of the two experiments.

In the real world experiment, the adversary $\mathscr{A}$ can, in each round i, adaptively specify an input X_i and an attack strategy $\mathcal{W}_i$ that is applied to the transformed circuit $\widehat{C}$ when run on input X_i with secret state $\widehat{M}_{i-1}$. The output Y_i resulting from the (possibly) faulty computation is given to the adversary and the state is updated to $\widehat{M}_i$ for the next evaluation. To formally describe such a process we introduce a special oracle, which we call Θ, that can be queried on $(X_i, \mathcal{W}_i)$ to return the result Y_i. More precisely, for any $(t, 0, Q)$-adversary $\mathscr{A}$, any circuit C and any initial state M_0 , consider the following experiment:

Experiment $\mathbf{Exp}^{\text{Real}}_{\Psi}(\mathscr{A}, C, M_0)$
1. $(\widehat{C}, \widehat{M}_0) \leftarrow \Psi(C, M_0)$
2. Output $\mathscr{A}^{\Theta(\widehat{C}, \widehat{M}_0, \cdot)}(C)$

In the second experiment the simulator $\mathscr{S}$ simulates the adversary's view, however, she has to do so without having tampering access to the transformed circuit. More precisely, the simulator only has oracle access to $C[M_0](\cdot)$. For a simulator $\mathscr{S}$ consider the following experiment for any circuit C, any initial state M_0 and any $(t, 0, Q)$-adversary $\mathscr{A}$:

Experiment $\mathbf{Exp}^{\text{Sim}}_{\Psi}(\mathscr{S}, C, M_0)$
1. Output $\mathscr{S}^{C[M_0](\cdot)}$

When the outputs of the two experiments (as random variables depending on the coin tosses of the circuit, of the compiler and of the adversary) are statistically close (cf. Section 2.1) we say Ψ is tamper-resilient. (Note that the adversary here is all-powerful, i.e. we aim at unconditional security.)

Definition 3.1 (Tamper-resilient compiler [Ish+06]). A circuit compiler Ψ is ϵ-tamper-resilient if for any $(t, 0, Q)$-adversary $\mathscr{A}$, for every circuit C and any initial state M_0, there exists a simulator $\mathscr{S}$ such that

$$\Delta(\mathbf{Exp}^{\text{Real}}_{\Psi}(\mathscr{A}, C, M_0), \mathbf{Exp}^{\text{Sim}}_{\Psi}(\mathscr{S}, C, M_0)) \leqslant \epsilon,$$

where the probabilities are taken over all the random coin tosses involved in the experiments.

ACHIEVING THE DEFINITION. To build a compiler Ψ satisfying Definition 3.1, the transformation of [Ish+06] relies on a fundamental axiom, stated below.

Axiom 1 (Tamper-proof gadgets). There exists small, stateless and computation independent tamper-proof gadgets of size linear in the security parameter κ.

The gadgets above are fixed, standard and universal elements that can be added to once standard cell library. This is far better than designing over and over task specific tamper-proof components. The above axiom means that an adversary is not allowed to tamper with the internal of the gadgets, but only with their inputs and outputs. Assuming simple components that withstand active physical attacks has been frequently made in the literature [Gen+04; Fau+10b; GR10; JV10]. Of course, the simpler the components are, the stronger is the result that one gets.

Hence, the main idea is to detect faulty computation combining randomised computation with redundant encodings applied to the values in the circuit. The gates in the original circuit C are replaced by the corresponding tamper-proof gadgets able to compute with the encodings itself. Moreover, if tampering is detected, a self-destruction mechanism is triggered that overwrites the complete state, so that, from there on, regular and tampering queries to the circuit can be trivially simulated. One difficulty that needs to be addressed is that this self-destruction mechanism itself is exposed to tampering attacks. In particular, an adversary could just try to cancel any trigger for self-destruction and from then on apply arbitrary attacks without being in danger of detection. Ishai *et al.* face this problem by spreading and propagating errors that appear during the computation. As discussed later we will use similar techniques in our compiler (cf. Section 3.3).

We can now state the main result of Ishai *et al.* [Ish+06].

Theorem 3.1 (Main theorem of [Ish+06]). *Let $\kappa > 0$ be a security parameter. There exists a compiler Ψ that is ϵ-tamper-resilient against $(t, 0, Q)$-adversaries, where $\epsilon = 2^{-\kappa}$. The compiler blows-up the circuit by a factor $O(\kappa^3 t)$ and requires $O(\kappa^2)$ bits of fresh randomness in every invocation.*

3.3 OUR TRANSFORMATION

The transformation of Ishai *et al.* has some limitations that makes it not practical. The first issue is that their construction is only suitable to resist $(t, 0, Q)$-adversaries. Whereas Definition 3.1 can be easily extended to cover the case of

(∞, δ, Q)-adversaries, as soon as the adversary is allowed to tamper with more than t wires in the compiler of [Ish+06] (in a single invocation), she is able to inject faults without being detected. However in practice it is not clear why there should be an upper bound on the number of wires the adversary can "touch" in every invocation of $\widehat{C}$. Hence the question:

> *Q1: Can we build a compiler secure against adversaries able to tamper with an* unbounded *number of wires in the circuit in* every *invocation?*

Note that an adversary able to tamper successfully with all the wires in C could just "re-program" the circuit in such a way it outputs the secret key, leaving no hope for security. To avoid this attack we will rely on the fact that the adversary is a (∞, δ, Q)-adversary for some $\delta > 0$, i.e. every attack fails (independently) with some probability δ. Recall that this actually describes what happens in a real attack (cf. Section 3.1).

The second issue we want to address is efficiency. In fact the compiler of Ishai *et al.* is not very efficient: The compiled circuit has a big blow-up factor and requires a huge amount of randomness in *each* invocation.

> *Q2: Can we get anything better in terms of efficiency and randomness? Can we use smaller tamper-proof gadgets?*

THE "TADING LEAKAGE" PARADIGM. To address the above questions, our approach is to relax Definition 3.1 by "helping the simulator" as follows. Instead of giving $\mathscr{S}$ just black-box access to $C[M_0]$, we allow it (once, at the beginning of the experiment) to learn a "small" amount of leakage Λ on the secret state M_0. Thus we say that a compiler Ψ is tamper-resilient if whatever an (∞, δ, Q)-adversary learns tampering with $\widehat{C}$ can be simulated having only black-box access to the original circuit *and given the leakage* Λ.[3] Now, if the original circuit C remains secure even given Λ, we can conclude that Ψ is tamper resilient. As we have seen in Section 2.2 there are several cryptosystems which are resilient to an amount of leakage as large as a constant fraction of the secret key. Essentially we are "trading" a small amount of auxiliary information on the secret state to achieve tamper-resilience. As we will see this small amount of leakage improves efficiency drastically.

Let's make the above idea more formal. The real world experiment is exactly the experiment $\mathbf{Exp}^{\mathrm{Real}}_{\Psi}(\mathscr{A}, C, M_0)$ introduced in Section 3.2, where the

3 Note that if $|\Lambda| = |M_0|$, simulation is trivial. In other words, the challenge is to achieve the definition using only a *small* amount of leakage.

adversary $\mathscr{A}$ is now an (∞, δ, Q)-adversary. Moreover, we let the simulator $\mathscr{S}_\lambda$ depend on a parameter λ, where λ will be the amount of auxiliary information given as hint. Besides having black-box access to $C[M_0](\cdot)$, at the beginning of the experiment the simulator additionally chooses an arbitrary function $f : \{0,1\}^* \to \{0,1\}^\lambda$ and learns the result of f evaluated on input the secret state M_0, i.e. $\Lambda = f(M_0)$. For a simulator $\mathscr{S}_\lambda$ we define the following experiment for any circuit C, any initial state M_0 and any (∞, δ, Q)-adversary $\mathscr{A}$:

Experiment $\mathbf{Exp}^{\mathrm{Sim}}_{\Psi}(\mathscr{S}_\lambda, C, M_0, \mathscr{A})$
1. $f \leftarrow \mathscr{S}_\lambda(\mathscr{A}, C)$ where $f : \{0,1\}^* \to \{0,1\}^\lambda$
2. Output $\mathscr{S}_\lambda^{C[M_0](\cdot)}(\Lambda)$ where $\Lambda = f(M_0)$

We say a compiler is tamper-resilient if the outputs of the two experiments are statistically close.

Definition 3.2 (Trading leakage for tamper-resilience). A circuit compiler Ψ is (λ, ϵ)-tamper-resilient if for any (∞, δ, Q)-adversary $\mathscr{A}$, for every circuit C and any initial state M_0, there exists a simulator $\mathscr{S}_\lambda$ such that

$$\Delta(\mathbf{Exp}^{\mathrm{Real}}_{\Psi}(\mathscr{A}, C, M_0), \mathbf{Exp}^{\mathrm{Sim}}_{\Psi}(\mathscr{S}_\lambda, C, M_0, \mathscr{A})) \leqslant \epsilon,$$

where the probabilities are taken over all the random coin tosses involved in the experiments.

We now describe our compiler in detail. Instead of computing with bits the compiled circuit $\widehat{C}$ will operate on redundant and randomised encodings of bits.

ENCODINGS. Our transformation is based on three encoding schemes, where each is used to encode the previous one. The first encoding, so called *Manchester encoding*, can be described by a deterministic function that takes as input a bit $b \in \{0,1\}$ and has output $MC(b) = (b, \overline{b})$. Decoding is done just by outputting the first bit.

The output $(b, \overline{b})$ is given as input to the next level of our encoding procedure, where we use a probabilistic function $mask : \{0,1\}^2 \times \{0,1\}^2 \to \{0,1\}^4$. Such a function uses as input additionally two random bits for masking its output. More precisely, we have $mask(MC(b), (r, r')) = (b \oplus r, r, \overline{b} \oplus r', r')$, with $(r, r') \xleftarrow{\$} \{0,1\}^2$. We denote with $\mathcal{MMC} \subset \{0,1\}^4$ the set of valid masked Manchester encoded bits, and with $\overline{\mathcal{MMC}} = \{0,1\}^4 \setminus \mathcal{MMC}$ the non-valid encodings.

Our final encoding consists of κ independent masked Manchester encodings:

$$Encode(b, \mathbf{r}) = mask(MC(b), (r_1, r'_1)), \ldots, mask(MC(b), (r_\kappa, r'_\kappa)),$$

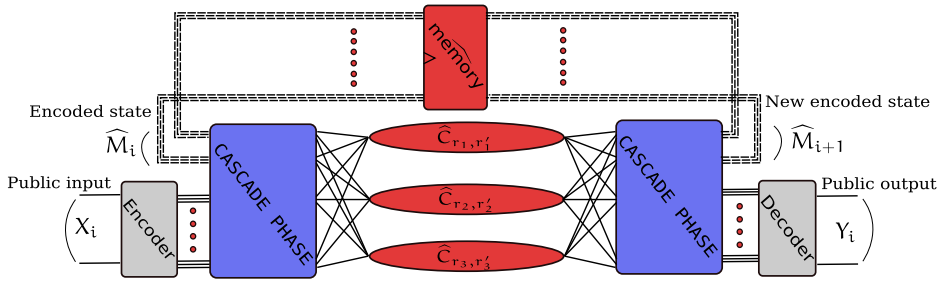

Figure 3.4: A global picture of our compiler in the case $\kappa = 3$. In the red-coloured parts we rely on gadgets of constant size, whereas in the blue-coloured parts gadgets of linear size (in the security parameter κ) are used.

with $\mathbf{r} = (r_1, r'_1, r_2, r_2, \ldots, r_\kappa, r'_\kappa) \in \{0,1\}^{2\kappa}$. Thus it has length 4κ bits and uses 2κ bits of randomness. When the randomness in an encoding is omitted, it is uniformly sampled, e.g. $\mathit{Encode}(b)$ denotes the random variable $\mathit{Encode}(b, \mathbf{r})$ where $\mathbf{r} \xleftarrow{\$} \{0,1\}^{2\kappa}$ is sampled uniformly at random. Again, we denote with $\mathcal{ENC} \subset \{0,1\}^{4\kappa}$ the set of all valid encodings and with $\overline{\mathcal{ENC}} = \{0,1\}^{4\kappa} \setminus \mathcal{ENC}$ the non-valid ones.

THE COMPILER. Consider any (probabilistic) Boolean circuit C that consists of Boolean NAND gates, randomness gates \$ and memory cells. We assume that the original circuit handles fanout[4] through special Copy gates taking one bit as input and outputting two copies. If κ copies are needed, the original value is passed through a subcircuit of $\kappa - 1$ Copy gadgets arranged in a tree structure. Let us first describe the transformation for the secret state. On factory setup 2κ random bits $\mathbf{r}_\Psi = (r_1, r'_1, \ldots, r_\kappa, r'_\kappa)$ are sampled uniformly. Then, each bit of the secret state m_i is encoded as in $\mathit{Encode}(m_i, \mathbf{r}_\Psi)$. Putting all these encodings together we get the initial transformed secret state $\widehat{M}_0$. The encoded secret state will be stored in the memory cells of $\widehat{C}$, but we will discuss this below. Notice that we use the *same* randomness for each encoding.

The global picture of our transformer consists of four different stages: the encoder, the input/output cascade phase, the transformation for the core and the decoder. These stages are connected as shown in Figure 3.4 and are described below.

4 By fanout we mean the maximum number of inputs that can be driven correctly by the output of a logical gate.

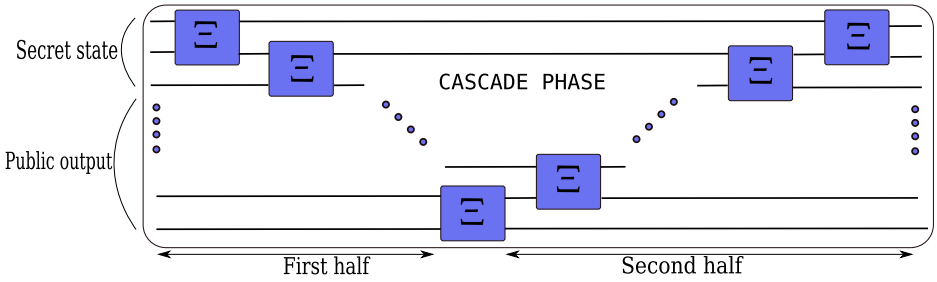

Figure 3.5: The cascade phase for error propagation and self-destruction. Every cascade gadget (in blue) has size linear in the security parameter κ.

- *The encoder and the decoder.* Since the compiled circuit computes with values in encoded form, we need to specify how to encode and decode the public inputs and outputs of $\widehat{C}$. The encoder (which is deterministic and build from copy and negation gates) encodes every bit of the input using randomness $\mathbf{r}_\Psi$:

$$\mathsf{Encoder}(x_1, \ldots, x_n) = (\mathit{Encode}(x_1, \mathbf{r}_\Psi), \ldots, \mathit{Encode}(x_n, \mathbf{r}_\Psi)),$$

  where $x_1, \ldots, x_n \in \{0, 1\}$. The decoding phase Decoder simply outputs the XORs of the first two bits of every encoding.

- *The input and output cascade phases.* For self-destruction we use a tool already introduced by [Ish+06]—the *cascade phase* (cf. Figure 3.5). In our construction we make use of two cascade phases: an *input* cascade phase and an *output* cascade phase. As shown in Figure 3.4 on the preceding page the input cascade phase takes as input the output of the encoder and the encoded secret state. The output cascade phase takes as inputs the output of the core and the updated secret state.[5] As discussed later in Section 3.4, for technical reasons we require that the secret state is always in the top part and the public output is always on the bottom part of the cascade phase. For ease of description we call the part of the cascade phase that takes the inputs as the first half and the part that produces the outputs as the second half. This is shown in Figure 3.5.

5 Notice that the input and the output cascade phases might have a different number of inputs/outputs.

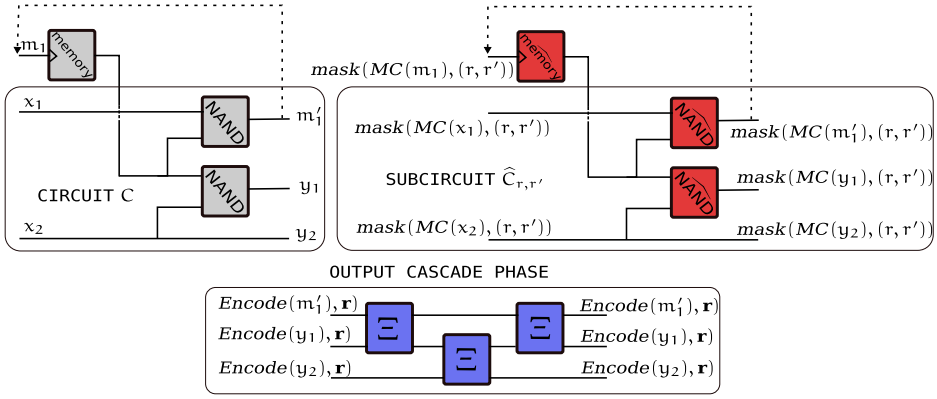

Figure 3.6: The compiler Ψ applied to a concrete circuit C (in the top left picture). The top right picture shows one of the subcircuits $\widehat{C}_{r,r'}$ in the core of the compiled circuit $\widehat{C}$. The gadgets used in $\widehat{C}_{r,r'}$ are all of constant size. Finally the bottom picture shows the output cascade phase.

Inside the cascade phase we make use of special *cascade gadgets* $\Xi : \{0,1\}^{8\kappa} \to \{0,1\}^{8\kappa}$. The gadgets behave like the identity function if the inputs are valid encodings using randomness $\mathbf{r}_\Psi$, and output $0^{8\kappa}$ otherwise, i.e.

$$\Xi(A,B) = \begin{cases} A,B & \text{if } A,B \in \{Encode(0,\mathbf{r}_\Psi), Encode(1,\mathbf{r}_\Psi)\} \\ 0 & \text{otherwise.} \end{cases}$$

The gadgets are assumed to be *tamper-proof*, i.e. the adversary is allowed to tamper with their inputs and outputs, but she cannot modify their internals (cf. Axiom 1).

- *The core.* With $\widehat{\mathsf{NAND}}_{r,r'} : \{0,1\}^{2\times 4} \to \{0,1\}^4$ we define a NAND gate which works on masked Manchester encodings using randomness r, r' (on input and output). If the input contains anything else than a valid masked Manchester encoding, the output is $0^4 \in \overline{\mathcal{MMC}}$. The truth table of these gadgets is given in Table 3.1. Similarly we denote with $\widehat{\mathsf{Copy}}_{r,r'} : \{0,1\}^4 \to \{0,1\}^{2\times 4}$ a Copy gate which takes as input a masked Manchester encoding using randomness r, r' and outputs two copies of it. Whenever the input contains anything else than a masked Manchester encoding using randomness r, r', the output is $0^8 \in \overline{\mathcal{MMC}}$.

$$\widehat{\mathsf{Copy}}_{r,r'}(A) = \begin{cases} A,A & \text{if } A \in \{mask(MC(0,r,r')), mask(MC(1,r,r'))\} \\ 0^8 & \text{otherwise.} \end{cases}$$

1st Input	2nd Input	Output
$mask(MC(0,r,r'))$	$mask(MC(0,r,r'))$	$mask(MC(1,r,r'))$
$mask(MC(0,r,r'))$	$mask(MC(1,r,r'))$	$mask(MC(1,r,r'))$
$mask(MC(1,r,r'))$	$mask(MC(0,r,r'))$	$mask(MC(1,r,r'))$
$mask(MC(1,r,r'))$	$mask(MC(1,r,r'))$	$mask(MC(0,r,r'))$
$\star$	$\star$	0^4

Table 3.1: Truth table of $\widehat{\mathsf{NAND}}_{r,r'} : \{0,1\}^{2\times 4} \to \{0,1\}^4$.

Finally we let $\widehat{\$}_{r,r'}$ denote a randomness gadget outputting a fresh masked Manchester encoded random bit.

With $\widehat{C}_{r,r'}$ we denote the circuit we get by replacing every wire in C with 4 wires (carrying an encoding in $\mathcal{MMC}$ using randomness r, r') and every NAND gate (resp. Copy gate, \$ gate) in C with a $\widehat{\mathsf{NAND}}_{r,r'}$ (resp. $\widehat{\mathsf{Copy}}_{r,r'}$, $\widehat{\$}_{r,r'}$). Similar to the Ξ gadgets, we require the $\widehat{\mathsf{NAND}}_{r,r'}$, $\widehat{\mathsf{Copy}}_{r,r'}$, $\widehat{\$}_{r,r'}$ gadgets to be *tamper-proof*, i.e. the adversary is allowed to tamper with their inputs and outputs, but cannot modify the internals (cf. again Axiom 1). Note that if we want to avoid the use of $\widehat{\$}_{r,r'}$ gadgets we can derandomise the original circuit C replacing the \$ gates with the output of a Pseudo-Random Generator (PRG).[6] The core of the transformed circuit consists of the κ circuits $\widehat{C}_{r_1,r'_1}, \ldots, \widehat{C}_{r_\kappa,r'_\kappa}$ (where the r_i, r'_i are from $\mathbf{r}_\Psi$).

To sum up, our compiled circuit has a blow-up of $O(\kappa)$ and requires only 2κ bits of randomness during production (no randomness at run-time is needed). Also note that we need tamper-proof gadgets of linear size (in κ) only in the cascade phase, whereas the core relies on gadgets of constant size in κ. In contrast, the transformation of [Ish+06] requires tamper-proof gadgets of linear (in κt) size in the entire circuit, albeit simpler ones than we do.[7] See Figure 3.6 for a concrete example how the compiled circuit looks like in a simple case.

6 A PRG is a deterministic algorithm taking a small random seed as input and producing an output which is indistinguishable from random [KL07, Chapter 6].

7 More precisely, in the core they need tamper-proof NAND gadgets taking $4\kappa t$ bits as input.

3.4 SECURITY PROOF

To prove security of our transformation, we construct an efficient simulator $\mathscr{S}_\lambda$ that—having only black-box access to the original circuit C—can simulate access to the transformed circuit $\widehat{C} = \Psi(C)$ including adversarial tampering queries. The main challenge here is consistency, that is, answers computed by $\mathscr{S}_\lambda$ must have the same distribution as an adversary would see when tampering with C.

When $\mathscr{A}$ tampers with C, a subsequent invocation of C can have one of three different outcomes:

1. *Nothing happens:* The invocation goes through as if no tampering did happen (this is e.g. the case if a wire is set to 0, but its value during the invocation is 0 anyway).

2. *Self-destruct:* The redundancy added to C "detects" tampering, and the entire state is deleted.

3. *Successful tampering:* The outcome of C changes as a consequence of the tampering, and this tampering was not detected.

In a first step, we show that case (3) will not happen but with exponentially small probability. To show this, we use the fact that tampering with any particular wire fails with probability δ, and moreover that every bit carried by a wire in C is encoded in $\widehat{C}$ with a highly redundant and randomised encoding. This guarantees that the chance of an adversary to change a valid encoding of a bit to its complement is tiny: either she has to be lucky—in the sense that she tampers with many wires at once and all attacks succeed—or she has to guess the randomness used in the encoding.

As we ruled out case (3), we must only build a simulator $\mathscr{S}_\lambda$ that simulates C as if no tampering has happened (i.e. case (1)). This is easy as $\mathscr{S}_\lambda$ has access to C which is functionally equivalent. Moreover, at some point $\mathscr{S}_\lambda$ has to simulate a self-destruct (i.e. case (2)). Unfortunately there is no way for the simulator to know when the self-destruct happens (as the probability of this event can be correlated with the secret state). To avoid this impasse, we will provide the exact point of failure as auxiliary input to $\mathscr{S}_\lambda$. (As we will see later in the proof, the point of failure can be easily encoded using a logarithmic amount of leakage on the secret state.)

Note that the simulator has to continue simulation even after the self-destruct. This seems easy, as now all the secret state has been deleted. There is one important technicality though. As tampering is permanent, even after self-destruct

the simulator $\mathscr{S}_\lambda$ must simulate a circuit in a way that is consistent with the simulation so far. A priori the simulator only knows which wires the adversary tried to tamper, but recall that each tampering is only successful with probability $1-\delta$. For this reason, we let the simulator choose all the randomness used, including the randomness of the compiler (which generates $\widehat{C}$ from C) and the randomness that determines the success of the tampering attacks. Knowledge of this randomness, allows the simulator to continue simulation after self-destruct. Note that the above-mentioned auxiliary information (i.e. the point at which self-destruct is triggered) can be computed—once and for all at the beginning of the simulated experiment—as a function of this randomness, and the randomness used by the adversary.

THE BAD EVENTS. Consider the experiment $\mathbf{Exp}^{\text{Real}}_{\Psi}(\mathscr{A}, C, M_0)$. We will say that the adversary "triggered" the self-destruct, if in an invocation of $\widehat{C}$ we get (as a consequence of tampering with $\widehat{C}$) an invalid encoding at the input to a cascade gadget Ξ. The high level idea behind our circuit compiler, which will make simulation possible, can be summarised as follows: (i) Any kind of tampering attempt is much more likely to trigger self-destruct than to succeed in changing a valid encoding of b into an encoding of $1-b$ and (ii) Once self-destruct is triggered, the entire secret state of $\widehat{C}$ gets erased with overwhelming probability.

The reason the latter could actually fail is that even though we have an invalid encoding at the input to Ξ, the adversary can also tamper with its output (which will be all zeros), potentially changing it back to a valid encoding. We define two event BAD_1 and BAD_2 which hold if the first or second of the unlikely cases above occurs.

$\text{BAD}_1 = 1$ if in the experiment $\mathbf{Exp}^{\text{Real}}_{\Psi}(\mathscr{A}, C, M_0)$ at some point the encoding $\mathit{Encode}(b, \mathbf{r}_\Psi)$ occurs either at the output of the core of $\widehat{C}$ or within a cascade phase, whereas in the un-tampered circuit this value should be $\mathit{Encode}(1-b, \mathbf{r}_\Psi)$. Moreover this happens before self-destruct was triggered.

$\text{BAD}_2 = 1$ if self-destruct is triggered, but "un-done" before the entire state has been erased. Notice that if the Ξ gadget where self-destruct is triggered lies in the first half of the (input or output) cascade phase, then the entire state is erased if the inputs to all the following Ξ's in this phase are not valid, as in this case the output of the cascade phase is all zeros. If the Ξ gadget lies in the second half of the input (resp. output) cascade phase,

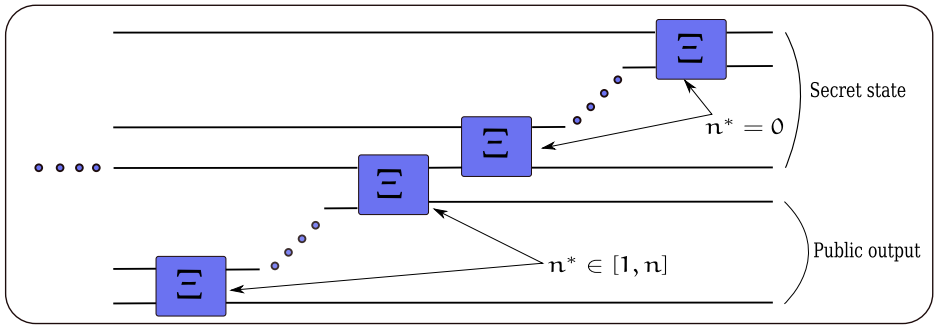

Figure 3.7: Cascade gadgets in the second half of the output cascade phase are assigned a label in $[1, n]$. The auxiliary information Λ includes the label n^* of the first cascade gadget where $0^{8\kappa}$ appears as an input.

then the state is erased if all inputs to the Ξ gadgets in the first half of the next output (resp. input) cascade phase are not valid encodings.

DESCRIPTION OF $\mathscr{S}_\lambda$. The simulator $\mathscr{S}_\lambda$ must "fake" a run of the experiment $\mathbf{Exp}^{\text{Real}}_{\Psi}(\mathscr{A}, C, M_0)$, which basically means to simulate the answers coming from the tampering oracle $\Theta(\widehat{C}, \widehat{M}_0, \cdot, \cdot)$, having only black-box access to $C[M_0](\cdot)$.

The simulator starts by sampling all the coin tosses for the experiment. This includes the following uniform coin tosses: The coins $\mathbf{r}_{\mathscr{A}}$ for the adversary and the coins $\mathbf{r}_\Psi$ for the transformation. Additionally, $\mathscr{S}_\lambda$ samples a sufficiently long string $\mathbf{r}_E$ which will model the failure of each tampering attempt. The bits of $\mathbf{r}_E$ are i.i.d. and each bit is 1 with probability δ ($\mathbf{r}_E[i] = 1$ meaning the i-th tampering attempt fails).

Next, the simulator can define the auxiliary input function $f : \{0,1\}^* \to \{0,1\}^\lambda$. The function f gets M_0 as input, and thus can completely simulate the experiment $\mathbf{Exp}^{\text{Real}}_{\Psi}(\mathscr{A}, C, M_0)$ using the randomness just sampled. This experiment defines the following three values, which are f's output:

- $\textsc{Abort} \in \{0,1\}$ is a predicate which is 1 if either of the predicates $\textsc{Bad}_1$ or $\textsc{Bad}_2$ (as defined above) is 1.
- $Q^* \in [1, Q]$ specifies the round (if there is any) in which a self-destruct is triggered, that is, the first round where an input from $\overline{\mathcal{ENC}}$ appears as input to a cascade gadget Ξ.
- $n^* \in [0, n]$ specifies which cascade gadget this is, as illustrated in Figure 3.7. If this is not one of the gadgets computing the final (public and

secret) output we set $n^* = 1$. If this is one of the gadgets computing the secret output we set $n^* = 0$. Otherwise n^* specifies the gadget exactly.

After getting this auxiliary input the simulator checks if $\textsc{Abort} = 1$, in this case it stops with output "simulation failed". Otherwise $\mathscr{S}_\lambda$ runs the experiment $\mathbf{Exp}^{\text{Real}}_\Psi(\mathscr{A}, C, M_0)$ using $\mathbf{r}_\mathscr{A}$ as $\mathscr{A}$'s randomness. We will show in Lemma 3.2 below that $\mathscr{S}_\lambda$ can do so perfectly. Then we show in Lemma 3.3 that the probability that $\textsc{Abort} = 1$ is very low.

Lemma 3.2. *There exists a simulator $\mathscr{S}_\lambda$ such that whenever $\textsc{Abort} = 0$ the simulation is perfect, i.e. the output is exactly the output of the experiment $\mathbf{Exp}^{\text{Real}}_\Psi(\mathscr{A}, C, M_0)$ (where the experiment uses the same randomness as sampled by $\mathscr{S}_\lambda$).*

Lemma 3.3. *The probability that f returns $\textsc{Abort} = 1$ is at most $3(1-\delta/2)^\kappa$.*

These lemmas, whose proofs we postpone for a moment, imply our main theorem.

Theorem 3.4 (Tamper-resilience against (∞, δ, Q)-adversaries). *Let $0 < \delta < 1/2$, $\kappa > 0$. Consider an arbitrary Boolean circuit with n bits of public output. The compiler Ψ of Section 3.3 is (λ, ϵ)-tamper-resilient, where $\lambda = \log(Q) + \log(n+1) + 1$ and $\epsilon = 3(1-\delta/2)^\kappa$.*

Proof. Let $\mathcal{R}$ be the randomness space of the experiment $\mathbf{Exp}^{\text{Real}}_\Psi(\mathscr{A}, C, M_0)$. For $\mathbf{r} \in \mathcal{R}$ let $\mathbf{Exp}^{\text{Real}}_\Psi(\mathscr{A}, C, M_0)[\,\mathbf{r}\,]$ denote the outcome of the experiment when using randomness $\mathbf{r}$. Similarly let $\mathbf{Exp}^{\text{Sim}}_\Psi(\mathscr{S}_\lambda, C, M_0, \mathscr{A})[\,\mathbf{r}\,]$ denote the outcome of the experiment $\mathbf{Exp}^{\text{Sim}}_\Psi(\mathscr{S}_\lambda, C, M_0, \mathscr{A})$, assuming $\mathscr{S}_\lambda$ initially samples randomness $\mathbf{r}$. For any $\mathbf{r}$ where $\textsc{Abort} = 0$ we have by Lemma 3.2

$$\mathbf{Exp}^{\text{Sim}}_\Psi(\mathscr{S}_\lambda, C, M_0, \mathscr{A})[\,\mathbf{r}\,] = \mathbf{Exp}^{\text{Real}}_\Psi(\mathscr{A}, C, M_0)[\,\mathbf{r}\,].$$

By Lemma 3.3 for a random $\mathbf{r} \in \mathcal{R}$ we have $\textsc{Abort} = 1$ with probability at most ϵ. This implies that for a random $\mathbf{r}$, the output of the two experiments is ϵ-close, i.e.

$$\Delta\left(\mathbf{Exp}^{\text{Sim}}_\Psi(\mathscr{S}_\lambda, C, M_0, \mathscr{A}), \mathbf{Exp}^{\text{Real}}_\Psi(\mathscr{A}, C, M_0)\right) \leqslant \epsilon,$$

as claimed. □

PROOF OF LEMMA 3.2. We have to specify how (in the case $\textsc{Abort} = 0$) the simulator answers $\mathscr{A}$'s queries to the $\Theta(\widehat{C}, \widehat{M_0}, \cdot, \cdot)$ oracle. The answers must be exactly the same as the answers in the experiment $\mathbf{Exp}^{\text{Real}}_\Psi(\mathscr{A}, C, M_0)[\,\mathbf{r}\,]$ when using the same randomness $\mathbf{r} = \{\mathbf{r}_E, \mathbf{r}_\mathscr{A}, \mathbf{r}_\Psi\}$ as chosen by $\mathscr{S}_\lambda$.

- *The first* $Q^* - 1$ *queries.* For $i = 1, \ldots, Q^* - 1$, if $\mathscr{A}$ makes the query $(X_i, \mathcal{W}_i)$ the simulator forwards X_i to $C[M_0](\cdot)$ and gives the output Y_i to $\mathscr{A}$.
- *Queries* $Q^* + 1$ *to* Q. As $\textsc{Abort} = 0$ (and thus $\textsc{Bad}_2 = 0$), in the query Q^* or $Q^* + 1$ there is some point in the evaluation of $\widehat{C}$ where all wires are 0, and the simulator knows this point (from Q^*, n^*). Moreover, as the simulator chooses the randomness $\mathbf{r}$ of the experiment, she knows the state of $\widehat{C}$ exactly, in particular which wires were successfully tampered. Thus from the point where all wires are 0, the simulator can continue to compute the answers of the $\Theta(\widehat{C}, \widehat{M}_0, \cdot, \cdot)$ oracle itself.
- *The* Q^**-th query.* We haven't yet covered the query Q^*. This query is a hybrid between the two cases considered above. If $n^* = 0$ (i.e. the self-destruct is triggered at the end of the output cascade phase after the public output is already computed) we can handle this query exactly like the first $Q^* - 1$ queries. Also if $n^* = 1$ (i.e. self-destruct is triggered early in the query, before the cascade gadgets outputting the final public output were invoked), this query can be handled like in the previous case.

  If $n^* > 1$ the simulator first queries (as in the first $Q^* - 1$ queries) for Y_{Q^*}, but before forwarding this value to $\mathscr{A}$ it must adapt it to take into account that parts of it were deleted: the first $n^* - 1$ bits of Y_{Q^*} remain unchanged, the others are set to 0.
- *Finalising.* Finally $\mathscr{S}_\lambda$ outputs whatever $\mathscr{A}$ outputs.

By inspection, the queries above have exactly the same distribution as the outputs of the $\Theta(\widehat{C}, \widehat{M}_0, \cdot, \cdot)$ oracle in the experiment $\mathbf{Exp}^{\text{Real}}_{\Psi}(\mathscr{A}, C, M_0)[\,\mathbf{r}\,]$. □

PROOF OF LEMMA 3.3. Recall that $\textsc{Abort} = 1$ if either $\textsc{Bad}_1$ or $\textsc{Bad}_2$ (as defined at the beginning of this section) are 1. We upper bound the probability that $\textsc{Bad}_1 = 1$ and $\textsc{Bad}_2 = 1$ below. We will first state some properties of the transformed circuit $\widehat{C}$ which will be used in the proof of the lemma.

We first show (Lemma 3.5 below) that tampering with (a non-empty subset of) four wires holding a masked Manchester encoded value — where tampering with a wire fails independently with probability δ and the adversary may know the encoded value, but not the randomness used in the encoding — will result in an invalid value with probability $\delta/2$.

Let $0 < \delta < 1/2$ and consider the following game.

- An adversary chooses $b \in \{0, 1, \perp\}$ and four functions $f_1, f_2, f_3, f_4 : \{0,1\} \to \{0,1\}$ of which at least one is not the identity function.

- If $b = \perp$ set $(x_1, x_2, x_3, x_4) = (0, 0, 0, 0)$, otherwise sample random $r, r' \in \{0,1\}$ and let $(x_1, x_2, x_3, x_4) = \mathit{mask}(MC(b), (r, r'))$ be the masked Manchester encoding of b.
- For each $i \in \{1, 2, 3, 4\}$, with probability $1 - \delta$ set $y_i = f_i(x_i)$ and $y_i = x_i$ otherwise.

Lemma 3.5 (Tampering with $\mathcal{MMC}$). *The probability that at the end of the experiment just described, $(y_1, y_2, y_3, y_4) \in \overline{\mathcal{MMC}}$ (i.e. it is not a valid masked Manchester encoding) is at least $\delta/2$.*

Proof. We first look at the case where at least one of the f_i's — for concreteness say f_4 — is a toggle function, i.e. $f_4(x_4) = 1 - x_4$. The main observation is that any subset of three bits of a masked Manchester encoding uniquely determine the fourth. In particular, if $(y_1, y_2, y_3, y_4) \in \mathcal{MMC}$ then $y_1 \oplus y_2 \oplus y_3 \oplus 1 = y_4$. Recall that $y_4 = x_4$ with probability δ (i.e. we don't apply f_4) and $y_4 = 1 - x_4$ with probability $1 - \delta$. In one of the two cases we get $(y_1, y_2, y_3, y_4) \in \overline{\mathcal{MMC}}$, thus the encoding is invalid with probability at least δ.

Consider now the case where at least one of the f_i's — for concreteness say f_4 — is a set function, i.e. $f_4(x_4) = 1$ (the proof for the reset case is identical). As $\mathbb{P}[x_4 = 0] = 1/2$ and $y_4 = x_4 = 0$ with probability δ, we get $\mathbb{P}[y_4 = 0] = \delta/2$. Again, only in one of the two cases ($y_4 = 0$ or $y_4 = 1$) we get $(y_1, y_2, y_3, y_4) \in \overline{\mathcal{MMC}}$, thus the encoding is invalid with probability at least $\delta/2$. □

By Lemma 3.5, tampering with the output wires of a $\widehat{\mathsf{NAND}}_{r,r'}$ (or $\widehat{\mathsf{Copy}}_{r,r'}$) gate (which is 0^4 or $\mathit{mask}(MC(b), r, r')$) will result in an invalid encoding with probability at least $\delta/2$. We next show a general *composition lemma*, which essentially states that the property proven in Lemma 3.5 *composes* to an arbitrary subcircuit made of masked Manchester gadgets as used in $\widehat{C}$.

Lemma 3.6 (Tampering with the core of $\widehat{C}$). *The probability that in the experiment $\mathbf{Exp}^{\mathrm{Real}}_{\Psi}(\mathscr{A}, C, M_0)$ we have an encoding $\mathit{Encode}(b, \mathbf{r}_\Psi)$ at the output of the core, where in the un-tampered circuit this values would be $\mathit{Encode}(1 - b, \mathbf{r}_\Psi)$ and this happens before self-destruct is triggered, is at most $(1 - \delta/2)^\kappa$.*

Proof. Recall that the core of $\widehat{C}$ is made of κ independent subcircuits $\{\widehat{C}_{r_i, r'_i}\}$, each using *independent* randomness r_i, r'_i. The output of each subcircuits when no tampering took place, consists of the original output in masked Manchester encoded form (masked using the same randomness).

The first important observation is that if the adversary wants to change one of the encodings at the output of the core, she must tamper with *every* subcircuit

(either in a single round or in different rounds). Consider the very last level where the adversary applies a tampering attack in a given subcircuit, say $1 \leqslant \ell \leqslant \mathit{depth}(\widehat{C}_{r_i,r'_i})$.[8] The value $\ell = 1$ means that the adversary tampers only with the input of the subcircuit, whereas the value $\ell = \mathtt{depth}(\widehat{C}_{r_i,r'_i})$ means that the attack involves also the encoded output. Since any tampering strategy ending at level ℓ involves the output of some masked Manchester gadget, Lemma 3.5 says that the output of that gadget is in $\overline{\mathcal{MMC}}$ with probability at least $\delta/2$. The key observation here is that, since ℓ is the last level where the adversary applies an attack, the invalid encoding will propagate to the output of the subcircuit. Thus the output of that subcircuit has at least one value in $\overline{\mathcal{MMC}}$ with probability at least $\delta/2$. Moreover this argument applies to all the subcircuits since they use *independent* randomness. As a consequence with probability at least $1-(1-\delta/2)^\kappa$ one of the wire bundles at the output carries a value in $\overline{\mathcal{ENC}}$, and thus triggers self-destruct. □

Note that the above lemma does not cover all cases of the $\textsc{Bad}_1 = 1$ case. The other possibility to get $\textsc{Bad}_1 = 1$ is by tampering within the cascade phase, hoping to change $\mathit{Encode}(b, \mathbf{r}_\Psi)$ into $\mathit{Encode}(1-b, \mathbf{r}_\Psi)$. A similar argument as the one used in the proof of Lemma 3.6, shows that in this case the probability of success can also be bounded by $(1-\delta/2)^\kappa$. Taking both cases into account, we get

$$\mathbb{P}\left[\textsc{Bad}_1 = 1\right] \leqslant 2(1-\delta/2)^\kappa.$$

It remains to bound $\mathbb{P}[\textsc{Bad}_2 = 1]$. Recall that $\textsc{Bad}_2 = 1$ if the adversary manages to change the output $0^{8\kappa}$ of a Ξ gadget (which was queried on an invalid input) back to something valid, i.e. $\mathit{Encode}(b, \mathbf{r}_\Psi), \mathit{Encode}(b', \mathbf{r}_\Psi)$ for some bits $b, b' \in \{0,1\}$. Even in the case $\delta = 0$ (i.e. when tampering always succeeds), we can upper bound the probability that the adversary can "un-do" tampering on one particular Ξ gadget by $2^{-2\kappa}$. To see this, assume an adversary changes $0^{8\kappa}$ to a valid encoding with advantage ϵ. In this case we can extract $\mathbf{r}_\Psi$ from its tampering queries, but as $\mathbf{r}_\Psi \in \{0,1\}^{2\kappa}$ is uniform (and the answers from the Θ oracle are independent of $\mathbf{r}_\Psi$), it follows that the adversary has 2κ bits min-entropy about $\mathbf{r}_\Psi$ and thus $\epsilon \leqslant 2^{-2\kappa}$. Taking into account that the adversary can use different guesses for $\mathbf{r}_\Psi$ on the outputs of the different Ξ gadgets, the probability that $\textsc{Bad}_2 = 1$ is $l \cdot 2^{-2\kappa}$ where l is the number of Ξ gadgets in $\widehat{C}$.

8 Since the transformation in the core is gate-by-gate the depth of $\widehat{C}_{r_i,r'_i}$ is the same as the depth of the original circuit C.

Assuming $l \cdot 2^{-2\kappa} \leqslant (1-\delta/2)^{\kappa}$ (which is the case for any interesting range of parameters) we get

$$\mathbb{P}[\text{BAD}_2 = 1] \leqslant (1-\delta/2)^{\kappa}.$$

This finishes the proof. □

3.5 EXTENSIONS AND OPEN PROBLEMS

In this section we discuss some further extensions and open problems in the area of tamper resilience.

THE CASE $\delta = 0$. The "trading leakage" paradigm introduced in this chapter allows for efficient compilers of arbitrary Boolean circuits, resilient against (∞, δ, Q)-adversary. It is a natural question to ask weather it is possible to achieve a similar efficiency improvement for the case where $\delta = 0$ (i.e. attacks are always successful) and the attacker is allowed to attack only up to t wires per invocation. (Recall that this is exactly the model considered in [Ish+06].)

Before we discuss the details of our new compiler Ψ_t, let us outline why the transformation from Section 3.3 is not secure when the failure probability of attacks on individual wires is highly correlated (the model of [Ish+06] is such a case). In a nutshell, the reason is that the gadgets in the core operate on encodings of constant length c. A $(t, 0, Q)$-adversary can successfully tamper with such encodings if $c \leqslant t$ (e.g. she can fix it to a fixed encoding). In this section we outline how to significantly improve the efficiency of the compiler Ψ_{IPSW} when a small amount of leakage is tolerated.

For readers familiar with [Ish+06] we recall the compiler Ψ_{IPSW} here. The transformation Ψ_{IPSW} uses randomisation and redundant encodings and works in two steps. First, using the transformation[9] Ψ_{ISW} from [ISW03] the original circuit $C[M_0]$ is transformed into a randomised circuit $C'[M'_0]$. That is, each wire w in C is represented by a κ-bit wire bundle in C' of the form $(r_1, \ldots, r_{\kappa-1}, r_1 \oplus \ldots \oplus r_{\kappa-1} \oplus w)$. The gates in C are replaced by gadgets that operate on such randomised encodings. The gadgets have to satisfy the invariant that learning up to $\kappa - 1$ wires does not reveal any information about the

9 The transformation Ψ_{ISW} has the property that the compiled circuit $\widehat{C}$ is still secure against a *passive* adversary probing up to t wires in each clock cycle. The main trick used by [ISW03] is to reduce the tampering strategy of a $(t, 0, Q)$-adversary to a probing attack against $\widehat{C}$. Tamper-resilience thus follows from the security of Ψ_{ISW}.

1st Input	2nd Input	Output
0^{2kt}	0^{2kt}	1^{2kt}
0^{2kt}	1^{2kt}	1^{2kt}
1^{2kt}	0^{2kt}	1^{2kt}
1^{2kt}	1^{2kt}	0^{2kt}
$\star$	$\star$	$0^{kt}1^{kt}$

Figure 3.8: Truth table of a $\widehat{\mathsf{NAND}}$ gadget in the case of $(t, 0, Q)$-adversaries.

1st Input	2nd Input	Output
0^{2kt}	0^{2kt}	$(0^{2kt}, 0^{2kt})$
0^{2kt}	1^{2kt}	$(0^{2kt}, 1^{2kt})$
1^{2kt}	0^{2kt}	$(1^{2kt}, 0^{2kt})$
1^{2kt}	1^{2kt}	$(1^{2kt}, 1^{2kt})$
$\star$	$\star$	$(0^{kt}1^{kt}, 0^{kt}1^{kt})$

Figure 3.9: Truth table of a cascade gadget in the case of $(t, 0, Q)$-adversaries.

encoded computation. This first transformation blows up the size of the circuit by $O(\kappa^2)$.

The circuit C' uses standard Boolean gates. In the next step each wire in C' (that is part of a wire bundle) is encoded with a $2\kappa t$ bit long redundant encoding. The encoding that is used maps 0 to $0^{2\kappa t}$ and 1 to $1^{2\kappa t}$. The value $0^{\kappa t}1^{\kappa t}$ is a special invalid state. The Boolean gates in C' are then replaced using the $\widehat{\mathsf{NAND}}$ gadgets given in Figure 3.8. These gadgets compute with the encoding just outlined above. This concludes the description of the core of the transformed circuit $\widehat{C}$. The output of the core is given as input to a cascade phase. This is essentially identical to the one described in Section 3.3 but built with the cascade gadgets shown in Figure 3.9. Notice that the gadgets in Figure 3.8 and 3.9 are *not* atomic. Instead, they can be built from standard Boolean gates and tamper-proof AND gates that take $4\kappa t$ bits as input. The result of the above outlined transformation gives us $\widehat{C}[\widehat{M}_0]$.

If we allow the tampering adversary to learn a small amount of information, then we can eliminate most of the overhead resulting from the first step. Our transformation Ψ_t essentially follows the transformation Ψ_{IPSW} with two changes. First, instead of randomizing each wire in C with $\kappa - 1$ random bits (to achieve statistical security with security parameter κ), in C' we use only a single bit of randomness. More precisely, each wire w_i in C is represented in C' by two wires that carry the values $(w_i \oplus r_i, r_i)$. The computation in C' is done in such a way that learning the value of a *single* wire in C' does not reveal information. This can be done with the techniques introduced in [ISW03] (i.e. replacing each gate in C by gadgets that are constructed from standard Boolean gates and probabilistic gates). Second, the gadgets in Figure 3.8 and 3.9 are tamper-proof. That is, an adversary can tamper with its inputs and outputs but the internals are not subject to attacks. These tamper-proof gadgets can be implemented in size linear in κt.

We give some further details below. For security parameter κ we define the probabilistic encoding scheme $\mathit{Encode}_t : \{0,1\} \to \{0,1\}^{4\kappa t}$ as $\mathit{Encode}_t(b) = (b^{2\kappa t} \oplus r^{2\kappa t}, r^{2\kappa t})$, where $r \xleftarrow{\$} \{0,1\}$. Each bit b of the initial secret state M_0 is then replaced by $\mathit{Encode}_t(b)$. Hence, the memory cells in C are replaced by $4\kappa t$ memory cells.

As in the transformation of Section 3.3, $\widehat{C}$ is structured in three phases: the encoder/decoder, the input/output cascade phases and the core. The encoder encodes every input bit b with Encode_t. The decoder takes the first bit of the encoding and the $(2\kappa t + 1)$-th bit and outputs the XOR. The input/output cascade phase is similar to the one described in Section 3.3, with the difference that it makes use of the tamper-proof cascade gadgets of Figure 3.9.

Essentially, $\widehat{C} \leftarrow \Psi_t(C)$ operates with encodings Encode_t instead of plain bits. To achieve this it proceeds in two steps. First every gate in C is replaced by gadgets (built from Boolean and randomness gates) that operate with masked bits, i.e. $(b \oplus r, r)$. This can for instance be done with the transformation Ψ_{ISW} from [ISW03]. Next each gate in C' is replaced with $\widehat{\mathsf{NAND}}$ gadgets of Figure 3.8. This gives us the transformed circuit $\widehat{C}$. Notice that even if the original circuit C was deterministic, the transformed circuit $\widehat{C}$ will contain randomness gates (this is due to the transformation Ψ_{ISW}). We can use the PRG construction proposed in [Ish+06] to de-randomize $\widehat{C}$.[10]

It is easy to see that changing the transformation Ψ_{IPSW} as outlined above decreases the size of $\widehat{C}$ by a factor of $O(\kappa^2)$. Also, the amount of randomness is decreased from $O(|C| \cdot \kappa^2)$ to $O(|C|)$ (where $|C|$ is the size of C).

We prove security of Ψ_t using the "trading leakage" paradigm of Section 3.3. This requires to show that any attempt to change a valid encoding in the core (or the cascade phases) results with high probability in an invalid encoding at the output of $\widehat{C}$'s core.

Lemma 3.7 (Tampering with the core and/or with the cascade phase of $\widehat{C}$). *Consider any circuit C and its transformation $\widehat{C} = \Psi_t(C)$. Every $(t, 0, Q)$-adversary tampering inside the core of $\widehat{C}$ either lets the computation in $\widehat{C}$ unchanged or results with probability at least $1 - 2^{-2\kappa}$ to an invalid encoding on some wire bundle at the output of the core.*

10 Notice that here we require a special PRG (namely the one from [Ish+06]), while in Section 3.3 we could just use any PRG for de-randomization. The reason for this is that in Section 3.3, if C is deterministic, then so is $\widehat{C}$. On the other hand if C is probabilistic we can make it deterministic using any standard PRG, and only afterwards apply our transformation.

Proof. Since $\widehat{C}$ only uses tamper-proof $\widehat{\mathsf{NAND}}$ and cascade gadgets, the attack strategy of the adversary can only include the inputs and outputs of such gadget. Notice that the minimal Hamming distance between two *valid* encodings is $d_H(0^{2\kappa t}, 1^{2\kappa t}) = 2\kappa t$. Since the adversary is only allowed to tamper with at most t wires in every round, either fixing or changing an encoding will require 2κ rounds. The main observation is that the randomness of the masking is refreshed in each round. Hence, the adversary is successful in each round with probability at most $1/2$. This concludes the proof. □

Similar to the proof of Theorem 3.4 we can use Lemma 3.7 to show tamper-resilience against $(t, 0, Q)$-adversaries.

Theorem 3.8 (Tamper-resilience against $(t, 0, Q)$-adversaries). *Let $\kappa > 0$ be the security parameter and $t \in \mathbb{N}$. Consider an arbitrary Boolean circuit C with n bits of public output. For any $Q = Q(\kappa)$, the compiler Ψ_t described above is (λ, ϵ)-tamper-resilient, where $\lambda = \log(Q) + \log(n+1) + 1$ and $\epsilon = 2^{-2\kappa}$.*

Proof. The proof is along the lines of Theorem 3.4 and is therefore omitted. □

USING ONLY GADGETS OF CONSTANT SIZE. It is an interesting open problem to reduce the size of the tamper-proof gadgets used in the transformation. Even though our compiler uses tamper-proof gadgets of constant size in the core of $\widehat{C}$ (in contrast to [Ish+06]), the cascade phase still relies on tamper-proof gadgets of linear size. One is tempted to "open-up" the gadgets, implementing them using smaller (tamper-proof) gadgets. However, it is not difficult to see that the composition argument we used in the core of $\widehat{C}$ fails in this case.

Another interesting approach would be to rely on the following "babushka doll construction".[11] The main idea is to let every cascade gadget of size $8k$ be replaced by a smaller cascade phase using gadgets of size $4k$. In the same fashion, every such a gadget "contains" a smaller cascade phase relying on gadgets of size $2k$ and so forth, until only gadgets of constant size are used. However, we don't know how to build the simulator for this construction and we can only conjecture its security.

BEYOND (∞, δ, Q)-ADVERSARIES. In this chapter we looked at the general case where the adversary is allowed to tamper with every part of the circuit. A simpler case is when the adversary can only tamper with the memory. Below

11 The name is inspired by the so called matryoshka doll, or babushka doll, a Russian nesting doll which is a set of wooden dolls of decreasing size placed one inside the other.

we sketch some results regarding this model and make a comparison with the general setting considered in this chapter.

- *Algorithmic Tamper-Proof (ATP) security.* Gennaro *et al.* [Gen+04] consider a model where an adversary can attack a circuit C with secret state M; the circuit C could e.g. compute digital signatures where M contains the secret key. The adversary can make regular queries to the cryptosystem, where on input X she receives $C[M](X)$. Additionally she can make "tamper" queries, where she chooses a function f, and the secret state is replaced with $f(M)$.

  The solution they propose is to sign the state. More precisely, they compile the circuit/state pair C, M into $\widehat{C}, \widehat{M}$ where the new state $\widehat{M} = \{M, \sigma\}$ contains a digital signature σ on the memory M. The circuit $\widehat{C}(\widehat{M})$ parses $\widehat{M} = \{M, \sigma\}$, checks if σ is a valid signature on M, and if so, outputs $C(M)$; otherwise it "self-destructs". The main advantage of this solution is that it works for any *efficient tampering function* f, but some problems remain. For example C has to be stateless (i.e. it cannot overwrite its state) as the public $\widehat{C}$ cannot contain the secret signing key. Moreover, as in our work, the attacker can learn some information about M by using tamper-queries (though at most $\log(Q)$ bits where Q is the number of tamper-queries).

- *Non-Malleable (NM) codes.* In [DPW10] the notion of "non-malleable codes" is proposed. It is shown that by encoding the state M (instead of appending a signature) one can avoid most of the problems in [Gen+04], albeit one must settle for more restricted families of tampering functions (not all efficient functions as in [Gen+04]). See [DPW10] for the details.

- *Related-Key Attacks (RKA).* A special case of tampering attacks are "related-key" attacks, where an adversary is allowed to query a primitive not only under the target key, but under other "related" (chosen by her) keys derived from it. A theoretical framework for this setting has been developed for pseudorandom functions and permutations [BK03; Luc04; BC10], one-way functions, hard-core predicates and pseudorandom generators [GL10], symmetric key encryption [AHI11], weak pseudorandom functions, public key encryption, identity-based encryption and signatures [BCM11]. Most of known construction of RKA-secure primitives, assume that the ensemble of tampering function is *claw-free*, this meaning that any two distinct functions in the set disagree on *all* inputs. Note that if the set of admissible tampering functions contains both the function able to set a certain bit of

the key to zero, and, say, the identity function then this set is not claw-free. Moreover, any ensemble containing the function that can set the key to a constant value is also not claw-free. The assumption of claw-freeness is removed in [BCM11], where a more general (but still limited) class of tampering function is allowed.

- *Tamperable and leaky memory.* Kalai, Kanukurthi and Sahai [KKS11], considered a model where (i) all memory is leaky and leakage can be an arbitrarily chosen (efficient) function of the memory, (ii) all memory is tamperable and the tampering can be an arbitrarily chosen (efficient) function applied to the memory. In other words, they consider the case where the memory is both leaky and tamperable (arbitrarily). In this model they construct a signature scheme and an encryption scheme that are provably secure against such attacks, assuming that memory can be updated in a randomised fashion between episodes of tampering and leakage.

There are two fundamental difference between our work and the line of research presented in [BK03; Gen+04; Luc04; BC10; GL10; DPW10; AHI11; KKS11; BCM11]. On the one hand the adversary considered in the latter papers is much stronger, since she can apply tampering attacks from a much larger class of tampering functions (in the work of [Gen+04; KKS11] even arbitrary polynomial time functions). On the other hand, we (as well as [Ish+06]) consider adversaries that tamper with the entire computation. This is in contrast to [BK03; Gen+04; Luc04; BC10; GL10; DPW10; AHI11; KKS11; BCM11], where the adversary is restricted solely to attack the memory, but the computation is assumed to be completely tamper proof.

It is an open problem to build a compiler achieving the best of both worlds, namely security against adversaries able to tamper with the whole circuit applying arbitrary functions to the bits in the circuit.

4 EFFICIENT AUTHENTICATION FROM HARD LEARNING PROBLEMS

«... Now I'll give you something to believe. I'm just one hundred and one, five months and a day.» «I can't believe that!» said Alice. «Can't you?» the Queen said in a pitying tone. «Try again: draw a long breath, and shut your eyes.» Alice laughed. «There's no use trying,» she said: «one can't believe impossible things.»

LEWIS CARROLL, THROUGH THE LOOKING GLASS [CAR71]

CONTENTS

In this chapter, we show that looking at stronger security models (such as leakage-resilience and tamper-resilience) is not only important to fill the gap between theory and practice (thus delivering better security definitions, closer to reality) but could also yield new powerful techniques with broader applications in theoretical cryptography. In this sense, theory and practice can thus benefit from each other.

The context we will focus on is (secret key) authentication (cf. Figure 1.2): Alice (a.k.a. the prover $\mathscr{P}$) wants to authenticate herself to the Cheshire Cat (a.k.a. the verifier $\mathscr{V}$) using a previously shared secret string $\mathbf{s} \in \mathbb{Z}_2^\kappa$. As we will see in this chapter there are very well-defined models and paradigms for this purpose, based on other basic cryptographic primitives such as encryption schemes and digital signatures. However, in practice, there are cases in which devices are resource-constrained; this is e.g. the case of RFID devices.[1] In such

1 Radio-frequency identification (RFID) is a technology that uses radio waves to transfer data from an electronic tag, called RFID tag, attached to an object, through a reader for the purpose of identifying and tracking the object.

cases, it is not possible to rely on the aforementioned paradigms. Hence, the question:

Q: Can we design truly efficient *authentication protocols?*

Starting from a tampering-related question, we will develop new techniques able to address the question above.

An influential paper in the area of efficient authentication is a paper of Hopper and Blum presented for the first time at Asiacrypt 2001 [HB01]. The protocol introduced by Hopper and Blum is based on the LPN problem (cf. Section 2.3) and started an entire line of research trying to construct (highly efficient) authentication protocols meeting stronger and stronger security notions. However, after more than 10 years, a round-optimal protocol based on LPN and meeting the strongest notion of security, was still missing.

In [Kil+11] we have solved the above open problem. The starting point for our contribution is the LPN problem in the presence of a powerful tampering adversary. A recent result of Pietrzak [Pie10] shows that LPN is robust against a certain class of tampering attacks. Building on Pietrzak's result, we are able to construct (truly efficient) authentication protocols meeting the strongest security notion.

READER'S GUIDE. The context of secret key authentication is reviewed in Section 4.1, together with some known paradigms how to construct authentication protocols based on other basic primitives. In Section 4.2 we describe the state of the art for authentication protocols based on LPN. Our protocol is introduced and analysed in Section 4.3, starting from the result of Pietrzak on tampering with LPN. In Section 4.4 we explain what are the difficulties of turning our protocol into a MAC and we explain how to solve them, thus yielding the first MAC based on LPN. We conclude this chapter comparing our protocols to the other known solutions and stating some open problem for future research (cf. Section 4.5).

4.1 SECRET KEY AUTHENTICATION

In an authentication protocol, Alice (in what follows, the prover $\mathscr{P}$) wants to convince the Cheshire Cat (in what follows, the verifier $\mathscr{V}$) of her identity. To achieve this goal, the two parties exchange messages over an insecure channel controlled by the Red Queen (in what follows, the adversary $\mathscr{A}$). See also Fig-

ure 1.2 on page 4 for a graphical representation. In all the schemes considered in this chapter, we focus on computational security (cf. Section 2.2), thus $\mathscr{P}$, $\mathscr{V}$ and $\mathscr{A}$ are all PPT algorithms.

More formally a (secret key) authentication protocol has associated a key generation algorithm KGEN which takes as input the security parameter κ and outputs a secret vector $\mathbf{s} \in \mathbb{Z}_2^\kappa$ shared between $\mathscr{P}$ and $\mathscr{V}$. Then $\mathscr{P}$ and $\mathscr{V}$ start exchanging messages over the channel (possibly depending on the secret $\mathbf{s}$). At the end of the interaction, the verifier outputs a value in $\{\mathsf{accept}, \mathsf{reject}\}$ depending on the fact it accepts or not the prover as authentic.

We say an authentication protocol has completeness error α if, for every secret $\mathbf{s} \leftarrow \mathsf{KGEN}(1^\kappa)$, a honest execution of the protocol ends with the verifier returning accept with probability $1 - \alpha$.

SECURITY NOTIONS. The most basic scenario is the one of a *passive* attack. A passive attacker runs in two phases. In the first phase, the adversary eavesdrops the channel, observing a polynomial[2] number of honest executions of the protocol between $\mathscr{P}$ and $\mathscr{V}$. Then, in the second phase, she tries to convince the verifier replacing the prover in the protocol. We say an authentication protocol is (t, Q, ϵ)-secure against *passive* attacks, if no PPT adversary $\mathscr{A}$ running in time t and observing Q honest execution of the protocol, can convince the verifier with probability better than ϵ.

A slightly more general scenario is what we call an *active* attack. In an active attack, the adversary is additionally given the opportunity to replace the verifier in the first phase. In particular this means that $\mathscr{A}$ can arbitrarily deviate from the protocol while interacting with the prover. In the second phase, she is given a *single* shot to convince the verifier (exactly as in the passive case). We say an authentication protocol is (t, Q, ϵ)-secure against *active* attacks, if no PPT adversary $\mathscr{A}$ running in time t and interacting Q times with the honest $\mathscr{P}$ in the first phase, can convince the verifier with probability better than ϵ.

The strongest security notion for authentication protocols is security against Man-in-the-Middle (MiM) attacks. Here the adversary can initially interact (concurrently) with any number of provers and—unlike in an active attacks—also verifiers. The adversary gets to learn the verifiers accept/reject decisions. (In the concurrent setting, many protocols sessions are executed at the same time, involving many verifiers which may be talking with the same provers simultaneously.)

2 Whenever we say "polynomial", we mean polynomial in the security parameter κ. This is without loss of generality, since $\mathscr{A}$ must be efficient.

THE "CHALLENGE-AND-RESPONSE" PARADIGM. A very well known paradigm to construct authentication protocols is the "challenge-and-response" paradigm. Essentially, in a first message the verifier sends a challenge to the prover. Then the answers from the prover, together with the secret $\mathbf{s}$, is used by the verifier to generate the output in $\{\mathsf{accept}, \mathsf{reject}\}$.

The good news is that following the above paradigm yields provably secure authentication protocols meeting security against MiM attacks [BCK98]. These protocols rely on basic building blocks such as (symmetric key) encryption schemes and MACs (cf. Section 2.2).[3]

In what follows, we discuss a simple solution based on any secure MAC. Let $\Pi_M = (\mathsf{KGEN}, \mathsf{TAG}, \mathsf{VRFY})$ be a message authentication code with message space $\mathcal{M}$ and key space $\mathcal{K}$. The idea is to let $\mathscr{V}$ challenge $\mathscr{P}$ asking the MAC of a randomly chosen message $m \in \mathcal{M}$. A honest prover can compute the tag ϕ corresponding to m, using the shared key $K \in \mathcal{K}$. The protocol is shown in Figure 4.1.

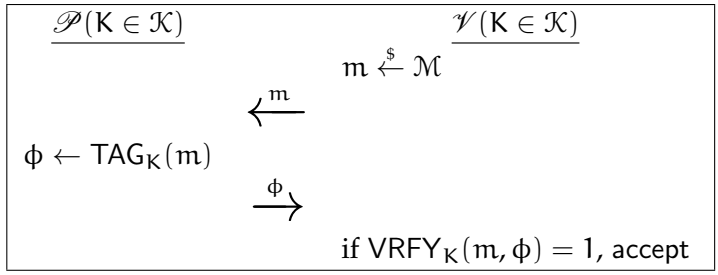

Figure 4.1: Authentication using a MAC.

Theorem 4.1 (Security of protocol in Figure 4.1). *If Π_M is $(t_{MAC}, Q, \epsilon_{MAC})$-secure against uf-cma adversaries, the authentication protocol in Figure 4.1 is (t, Q, ϵ)-secure against* active *attacks, where*

$$t \approx t_{MAC} \qquad \epsilon \leqslant \epsilon_{MAC} + \frac{Q}{\#\mathcal{M}}.$$

Proof. See Appendix B. □

Note that the protocol of Figure 4.1 is not secure against MiM attacks since there is one easy way for the adversary to make the verifier accept: play "man in

3 More precisely, Bellare *et al.* [BCK98] define their protocols in the public-key setting, relying on asymmetric encryption and digital signatures. However, the analysis for the symmetric-key analogues is very similar.

the middle" between the verifier and some prover instance, relaying messages back and forth between them until the verifier accepts. Yet, it is clear that this is not really an attack; there is no harm in the verifier accepting under these conditions since in fact it was actually talking to the prover. A possible solution comes from the idea of using "matching session ids" [BPRoo]: View a session id shared between a prover instance and the verifier as a "connection name", enabling the verifier to differentiate between different prover instances. It is not secret, and in particular will be given to the adversary. (Session ids are public in the sense that the adversary gets to see those created by any instances with which it interacts.) In the absence of an adversary, the session ids output by a prover instance and the verifier at the end of their interaction must be the same, but with high probability no two different prover instances should have the same session id, since otherwise the verifier cannot tell them apart. Victory for the adversary now will correspond to making the verifier accept with a session id not held by any prover instance. Moreover we let the adversary be successful also if she "confuses" the verifier by managing to make two different prover instances output the same session id.

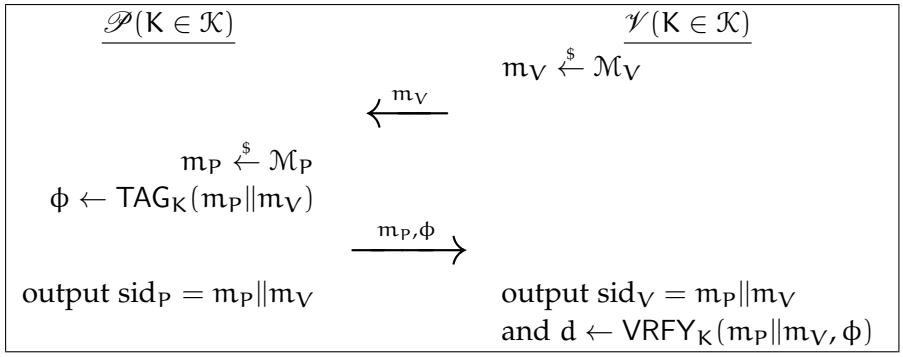

Figure 4.2: MiM-secure authentication using a MAC.

The full protocol is shown in Figure 4.2 and its security is proven below. (The result is known to be folklore and no explicit proof appears in the literature [Bel+01].) Note that now both the prover and the verifier choose a random challenge in the message spaces $\mathcal{M}_P$ and $\mathcal{M}_V$ (respectively); the MAC is the defined over the message space $\mathcal{M} = \mathcal{M}_P \times \mathcal{M}_V$. Recall that the verifier accepts if and only if $d = 1$ and $\mathrm{sid}_P = \mathrm{sid}_V$.

Theorem 4.2 (Security of protocol in Figure 4.2). *If Π_M is $(t_{MAC}, Q, \epsilon_{MAC})$-secure against uf-cma adversaries, the authentication protocol in Figure 4.2 is (t, Q, ϵ)-secure against* MiM *attacks, where*

$$t \approx t_{MAC} \qquad \epsilon \leqslant \epsilon_{MAC} + \frac{Q}{\#\mathcal{M}_V} + \frac{Q^2 - Q}{2 \cdot (\#\mathcal{M}_P)}.$$

Proof. See Appendix B. □

4.2 THE HB FAMILY

Note that the overall efficiency of the protocol in Figure 4.2, in the end, depends on the efficiency of the MAC Π_M. Thus, whether we can use the protocol in practice or not it depends on the context. At some point people started to ask if it is possible to construct *highly efficient* authentication protocols, to be deployed in resource-constrained scenarios.

A milestone in this line of work is a seminal paper of Hopper and Blum [HB01]. Essentially, the main observation is that building an authentication protocol based on the LPN assumption (cf. Section 2.3) yields very efficient schemes, since LPN relies on simple arithmetic modulo 2. On the practical side, LPN-based authentication schemes are strikingly efficient, requiring relatively few bit-level operations. Indeed, in their original proposal, Hopper and Blum suggested that humans (without the help of any calculator) could perform the computation in their provably-secure scheme, even with realistic parameters. The efficiency of LPN-based schemes also makes them suitable for weak devices like RFID tags, where even evaluating a blockcipher may be prohibitive.

For convenience, we briefly recall the (decisional version of the) LPN problem here. Let $\mathbf{x} \in \mathbb{Z}_2^\kappa$ be a random secret and denote with Bern_τ the Bernoulli distribution with parameter $0 < \tau < 1/2$ (i.e. $\mathbb{P}[e] = \tau$ if $e \xleftarrow{\$} \mathsf{Bern}_\tau$). We define $\Lambda_{\tau,\kappa}(\mathbf{x})$ to be the distribution over $\mathbb{Z}_2^{\kappa+1}$ obtained by choosing $\mathbf{r} \xleftarrow{\$} \mathbb{Z}_2^\kappa$, $e \xleftarrow{\$} \mathsf{Bern}_\tau$ and returning $(\mathbf{r}, \mathbf{r}^\mathsf{T} \cdot \mathbf{x} \oplus e)$. The $\Lambda_{\tau,\kappa}$ problem requires to distinguish $\Lambda_{\tau,\kappa}(\mathbf{x})$ from the uniform distribution $U_{\kappa+1}$ over $\mathbb{Z}_2^{\kappa+1}$.

THE HB PROTOCOL. Let $\mathbf{s} \in \mathbb{Z}_2^\kappa$ be the secret shared between the prover and the verifier. The protocol of Hopper and Blum is sufficiently simple to be described in one sentence. The verifier chooses a random vector $\mathbf{r} \xleftarrow{\$} \mathbb{Z}_2^\kappa$ and sends it to $\mathscr{P}$; the prover replies with $z = \mathbf{r}^\mathsf{T} \cdot \mathbf{s} \oplus e$ (where $e \leftarrow \mathsf{Bern}_\tau$). Thus, $\mathscr{V}$ accepts if and only if $z = \mathbf{r}^\mathsf{T} \cdot \mathbf{s}$.

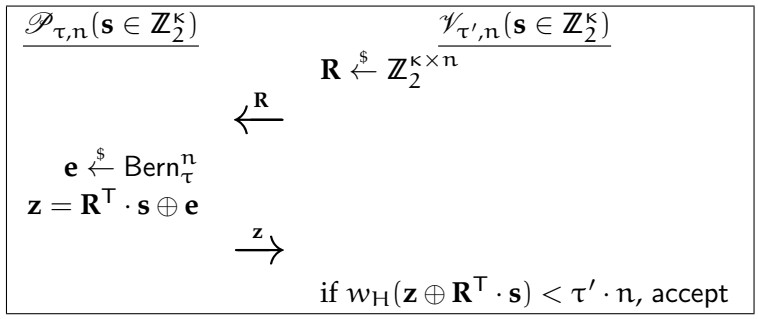

Figure 4.3: The HB protocol, secure against passive adversaries.

In the basic version described above, the protocol has a large *completeness error* (i.e. the probability that a honest $\mathscr{P}$ is rejected) τ and a large *soundness error* (i.e. the probability that a random answer z is accepted) $1/2$. However, the solution to these issues is just to repeat the protocol n times, either sequentially or in parallel. The parallel version (which is the most compact) is denoted HB and is shown in Figure 4.3. Basically, the vector $\mathbf{r}$ is now replaced by a matrix $\mathbf{R} \in \mathbb{Z}_2^{\kappa \times n}$. Similarly, the value z is now a vector $\mathbf{z} = \mathbf{R}^\mathsf{T} \cdot \mathbf{s} \oplus \mathbf{e}$, where $\mathbf{e} \xleftarrow{\$} \mathsf{Bern}_\tau^n$ is a vector of length n in which every component is drawn independently from Bern_τ. The verifier outputs accept if and only if the Hamming weight of the vector $\mathbf{z} \oplus \mathbf{R}^\mathsf{T} \cdot \mathbf{s}$ is less than $\tau' \cdot n$ for a threshold parameter $\tau < \tau' < 1/2$. For concreteness we can set $\tau' = \tau/2 + 1/4$.

The completeness error is the probability $\alpha_{\tau,n}$ that a vector $\mathbf{e} \xleftarrow{\$} \mathsf{Bern}_\tau^n$ has more than a fraction $\tau' \cdot n$ of 1 values. An application of the Chernoff bound (cf. Section 2.4, Theorem 2.3) yields

$$\alpha_{\tau,n} = \mathbb{P}\left[w_H(\mathbf{e}) > \tau' \cdot n : \ \mathbf{e} \xleftarrow{\$} \mathsf{Bern}_\tau^n\right] \leqslant 2^{-c_\tau \cdot n} = 2^{-\Theta(n)}, \tag{4.1}$$

for some constant c_τ only depending on τ. In a similar fashion, the soundness error is the probability $\alpha'_{\tau',n}$ that a random string $\mathbf{y} \xleftarrow{\$} \mathbb{Z}_2^\kappa$ has Hamming weight $\leqslant \tau' \cdot n$. Applying again the Chernoff bound we get

$$\alpha'_{\tau',n} = 2^{-n} \sum_{i=0}^{\lfloor \tau' \cdot n \rfloor} \binom{n}{i} < 2^{-c'_\tau \cdot n} = 2^{-\Theta(n)}, \tag{4.2}$$

for some constant c'_τ depending only on τ.

The security proof for the sequential case is due to Juels and Weis [JW05]. The parallel version was analysed for the first time by Katz and Shin [KS06; KSS10], who proved the following theorem.

Theorem 4.3 (Passive security of HB). *If the* $\mathsf{LPN}_{\tau,\kappa}$ *problem is* $(t_{\mathsf{LPN}}, n \cdot (Q+1), \epsilon_{\mathsf{LPN}})$*-hard, the* HB *protocol is* (t, Q, ϵ)*-secure against* passive *attacks, where*

$$t = O(t_{\mathsf{LPN}}) \qquad \epsilon = \epsilon_{\mathsf{LPN}} + 2^{-\Theta(n)}.$$

Proof. See Appendix C. □

THE HB$^+$ PROTOCOL. It is not hard to show that HB is *not* secure against active attacks. To see this, consider the sequential version of the protocol. An active adversary interacting with $\mathscr{P}$ in the first phase, can just replace the random vector $\mathbf{r}$ with a fixed vector, say $\delta = (1, 0, \ldots, 0)$. Repeating n times and taking majority yields one bit of secret information $\mathbf{s}[1] = \delta^{\mathsf{T}} \cdot \mathbf{s}$.

To overcome the above limitation, a modification of the HB protocol was proposed by Juels and Weis [JW05]. (This is actually the paper where the notion of active adversaries was introduced.) The idea is to rely on two shared secrets $\mathbf{s}_1, \mathbf{s}_2 \in \mathbb{Z}_2^\kappa$. The prover starts the interaction by drawing a random vector $\mathbf{r}_1 \xleftarrow{\$} \mathbb{Z}_2^\kappa$. The verifier chooses a random vector $\mathbf{r}_2 \xleftarrow{\$} \mathbb{Z}_2^\kappa$. Thus, $\mathscr{P}$ computes the answer $z = \mathbf{r}_1^{\mathsf{T}} \cdot \mathbf{s}_1 \oplus \mathbf{r}_2^{\mathsf{T}} \cdot \mathbf{s}_2 \oplus e$, where $e \xleftarrow{\$} \mathsf{Bern}_\tau$ and $\mathscr{V}$ accepts if and only if $z = \mathbf{r}_1^{\mathsf{T}} \cdot \mathbf{s}_1 \oplus \mathbf{r}_2^{\mathsf{T}} \cdot \mathbf{s}_2$.

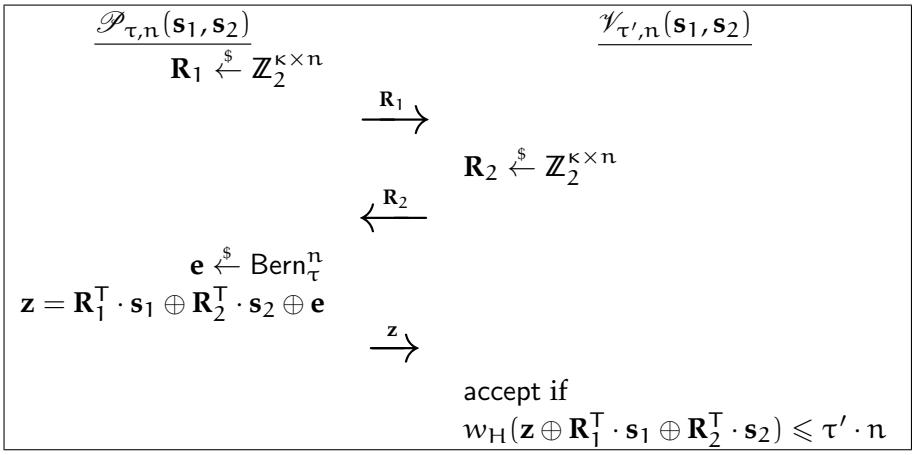

Figure 4.4: The HB$^+$ protocol, secure against active attacks.

Again, the protocol, in its basic form, has a large completeness and soundness error, but they can both be reduced by repetition. The parallel version, denoted HB$^+$, is depicted in Figure 4.4. The soundness and completeness errors for the parallel version are the same as in HB.

Theorem 4.4 (Active security of HB^+). *If the* $\mathsf{LPN}_{\tau,\kappa}$ *problem is* $(t_{\mathsf{LPN}}, n \cdot Q, \epsilon_{\mathsf{LPN}})$*-hard, the* HB^+ *protocol is* (t, Q, ϵ)*-secure against* active *attacks, where*

$$t = O(t_{\mathsf{LPN}}) \qquad \epsilon^2 = \epsilon_{\mathsf{LPN}} + 2^{-\Theta(n)}.$$

Proof. See Appendix C. □

Unfortunately, the HB^+ protocol is not secure against MiM attacks, as shown for the first time in [GRS05]. An attacker can choose a vector $\delta \in \mathbb{Z}_2^\kappa$ and each time a challenge $\mathbf{r}_2$ is sent, she can replace this vector with $\mathbf{r}_2 \oplus \delta$. At the end of the corresponding round, $\mathscr{A}$ has learned the value $z = \mathbf{r}_1^\intercal \cdot \mathbf{s}_2 \oplus (\mathbf{r}_2 \oplus \delta)^\intercal \cdot \mathbf{s}_2 \oplus e$. Hence, the acceptance/rejection decision of $\mathscr{V}$ at the end of the protocol, yields one bit of secret information. In fact, if at the end of the n rounds the verifier outputs accept (resp. reject), with high probability it must be $\delta^\intercal \cdot \mathbf{s}_2 = 0$ (resp. $\delta^\intercal \cdot \mathbf{s}_2 = 1$). Using this trick, we can recover $\mathbf{s}_2$ bit-by-bit changing δ opportunely. Once $\mathbf{s}_2$ is known, we can recover $\mathbf{s}_1$ using the same attack used in the case of HB.

STATE OF THE ART. In spite relevant effort [BCD06; MP07; BC08; GRS08d; GRS08a; GRS08c; Gol+08; LMM; OOV08], an authentication protocol based on LPN, secure against MiM attacks is still missing.

Moreover there are a few open problems also in the case of active security. Note that HB^+ requires 3 rounds (i.e. it is not round-optimal). In principle, since the matrix $\mathbf{R}_2$ is chosen *independently* of $\mathbf{R}_1$, we could drop the first message of the verifier and let $\mathscr{P}$ send the matrix $\mathbf{R}_1$ together with the answer $\mathbf{z}$ in the last message. However, the security proof for HB^+ does not carry over the modified version of the protocol. The reason is that the original proof needs to rewind the adversary at the point she already chose the matrix $\mathbf{R}_2$ (cf. Appendix C for details).

The rewinding approach has also the disadvantage of a non-tight security reduction (this is where we loose the factor $\sqrt{\epsilon}$).[4]

The main contribution of [Kil+11] is a solution to all the above issues.

4.3 NON HB-STYLE AUTHENTICATION

In this section we introduce a new (round optimal) authentication protocol based on LPN. Our protocol has active security and a tight reduction to the LPN prob-

4 Another issue is that proofs based on rewinding do not carry over the quantum setting [VDG98].

lem, thus solving the main issues of HB^+ (cf. the discussion at the end of the previous section). The main ingredient of the new protocol, is a "new" hardness assumption introduced by Pietrzak and showed to be actually equivalent to the LPN assumption in [Pie10]. We give a flavour of Pietrzak's result below.

TAMPERING WITH LPN. We stick only to a special case of the result in [Pie10]; for a more detailed discussion see Appendix D. Imagine an adversary $\mathscr{A}$ tampering with the LPN secret $\mathbf{x} \in \mathbb{Z}_2^\kappa$, as follows. Instead of just seeing LPN samples of the form $(\mathbf{r}, \mathbf{r}^\mathsf{T} \cdot \mathbf{s} \oplus e)$, i.e. samples drawn from $\Lambda_{\tau,\kappa}(\mathbf{x})$, the attacker can now ask for inner products not only with the secret $\mathbf{x}$, but even with the related secret $\mathbf{x}' = \mathbf{A} \cdot \mathbf{x} \oplus \mathbf{b}$ where $\mathbf{A} \in \mathbb{Z}_2^{\kappa \times \kappa}$ and $\mathbf{b} \in \mathbb{Z}_2^\kappa$ can be adaptively chosen, but $\mathbf{A}$ must have sufficiently large rank. This models a tampering adversary able to inject faults into the memory and then interacting with the "machine" generating the LPN samples. Does such an adversary $\mathscr{A}$ gain any advantage in telling apart samples drawn from this modified distribution and samples drawn from the uniform distribution?

More formally, for a minimal dimension $d \leqslant \kappa$, a secret $\mathbf{x} \in \mathbb{Z}_2^\kappa$ and $\mathbf{A} \in \mathbb{Z}_2^{\kappa \times \kappa}$, $\mathbf{b} \in \mathbb{Z}_2^\kappa$, consider the following distribution:

$$\Gamma_{\tau,\kappa,d}(\mathbf{x}, \mathbf{A}, \mathbf{b}) = \begin{cases} \perp & \text{if } \operatorname{rank}(\mathbf{A}) < d \\ \Lambda_{\tau,\kappa}(\mathbf{A} \cdot \mathbf{x} \oplus \mathbf{b}) & \text{otherwise} \end{cases}$$

and let $\Gamma_{\tau,\kappa,d}(\mathbf{x}, \cdot, \cdot)$ be the oracle that on input $\mathbf{A}, \mathbf{b}$ outputs a sample from $\Gamma_{\tau,\kappa,d}(\mathbf{x}, \mathbf{A}, \mathbf{b})$. The Subspace LPN (SLPN) problem, is to distinguish (adaptively chosen) samples from $\Gamma_{\tau,\kappa,d}(\mathbf{x}, \cdot, \cdot)$ from the uniform distribution over $\mathbb{Z}_2^{\kappa+1}$.

Definition 4.1 (Subspace LPN assumption). Let $\kappa, d \in \mathbb{Z}$, where $d \leqslant \kappa$. The (decisional) $\mathsf{SLPN}_{\tau,\kappa,d}$ problem is (t, Q, ϵ)-hard if for every distinguisher $\mathscr{D}$ running in time t and asking Q queries

$$\left| \mathbb{P}\left[\mathscr{D}^{\Gamma_{\tau,\kappa,d}(\mathbf{x},\cdot,\cdot)} : \mathbf{x} \xleftarrow{\$} \mathbb{Z}_2^\kappa \right] - \mathbb{P}\left[\mathscr{D}^{U_{\kappa+1}(\cdot,\cdot)} \right] \right| \leqslant \epsilon,$$

where $U_{\kappa+1}(\cdot, \cdot)$ on input a pair $(\mathbf{A}, \mathbf{b})$ outputs a sample from $U_{\kappa+1}$ when $\operatorname{rank}(\mathbf{A}) \geqslant d$ and $\perp$ otherwise.

The following lemma states that the Subspace LPN problem mapping to dimension $d + g$ is almost as hard as the standard LPN problem with secrets of length d. The hardness gap is exponentially small in g.

Lemma 4.5 (LPN ⇒ SLPN). *For any* $\kappa, d, g \in \mathbb{Z}$ *(where* $\kappa \geqslant d + g$*), if the* $\mathsf{LPN}_{\tau,d}$ *problem is* $(t_{\mathsf{LPN}}, Q, \epsilon_{\mathsf{LPN}})$*-hard then the* $\mathsf{SLPN}_{\tau,\kappa,d+g}$ *problem is* (t, Q, ϵ)*-hard where*

$$t = t_{\mathsf{LPN}} - \mathrm{poly}(\kappa, Q) \qquad \epsilon = \epsilon_{\mathsf{LPN}} + \frac{2Q}{2^{g+1}}.$$

Proof. See Appendix D. □

We also need a special case of the SLPN problem where the adversary does not ask for inner products with $\mathbf{x}' = \mathbf{A} \cdot \mathbf{x} \oplus \mathbf{b}$ for any $\mathbf{A}$ (of rank $\geqslant d$) but only with *subsets* of $\mathbf{x}$ (of size $\geqslant d$). We call this variation the Subset LPN problem. Formally, for $\mathbf{x}, \mathbf{v} \in \mathbb{Z}_2^\kappa$, let $\mathrm{diag}(\mathbf{v}) \in \mathbb{Z}_2^{\kappa \times \kappa}$ denote the zero matrix with $\mathbf{v}$ in the diagonal and define the distribution:

$$\Gamma^*_{\tau,\kappa,d}(\mathbf{x}, \mathbf{v}) = \Gamma_{\tau,\kappa,d}(\mathbf{x}, \mathrm{diag}(\mathbf{v}), 0^\kappa) \begin{cases} \perp & \text{if } w_H(\mathbf{v}) < d \\ \Lambda_{\tau,\kappa}(\mathbf{x} \wedge \mathbf{v}) & \text{otherwise} \end{cases}$$

Definition 4.2 (Subset LPN assumption). Let $\kappa, d \in \mathbb{Z}$, where $d \leqslant \kappa$. The (decisional) $\mathsf{SLPN}^*_{\tau,\kappa,d}$ problem is (t, Q, ϵ)-hard if for every distinguisher $\mathscr{D}$ running in time t and asking Q queries

$$\left| \mathbb{P}\left[\mathscr{D}^{\Gamma^*_{\tau,\kappa,d}(\mathbf{x},\cdot)} : \mathbf{x} \xleftarrow{\$} \mathbb{Z}_2^\kappa \right] - \mathbb{P}\left[\mathscr{D}^{U_{\kappa+1}(\cdot)} \right] \right| \leqslant \epsilon,$$

where $U_{\kappa+1}(\cdot)$ on input a vector $\mathbf{v}$ outputs a sample from $U_{\kappa+1}$ when $w_H(\mathbf{v}) \geqslant d$ and $\perp$ otherwise.

For vectors $\mathbf{x}, \mathbf{v} \in \mathbb{Z}_2^\kappa$, we write $\mathbf{x}_{\downarrow \mathbf{v}}$ to denote the vector in $\mathbb{Z}_2^{w_H(\mathbf{v})}$ obtained by projecting $\mathbf{x}$ on the coordinates where $\mathbf{v}$ is 1. For instance if $\mathbf{x} = (1,0,1,0,0,0,1)$ and $\mathbf{v} = (0,0,1,1,1,0,0)$, then $\mathbf{x}_{\downarrow \mathbf{v}} = (1,0,0)$. Samples from $\Gamma^*_{\tau,\kappa,d}(\mathbf{x}, \cdot)$ are pairs of the form $(\mathbf{r}, \mathbf{r}^\mathsf{T}_{\downarrow \mathbf{v}} \cdot \mathbf{x}_{\downarrow \mathbf{v}} \oplus e) \in \mathbb{Z}_2^{\kappa+1}$, where $e \xleftarrow{\$} \mathrm{Bern}_\tau$. Observe that to compute the value of the inner product, only the vector $\mathbf{r}_{\downarrow \mathbf{v}} \in \mathbb{Z}_2^{w_H(\mathbf{v})}$ is relevant. We will use this observation to improve the communication complexity of our authentication protocol (and of our MACs).

OUR PROTOCOL. In the protocols of the HB family, the prover must compute LPN samples of the form $\mathbf{R}^\mathsf{T} \cdot \mathbf{s} \oplus \mathbf{e}$, where $\mathbf{R}$ is the challenge chosen by the verifier in the first message. We take a different approach. Instead of sending $\mathbf{R}$, we now let the verifier choose a random subset of the bits of $\mathbf{s}$ to act as the "session-key" for this interaction. In other words, $\mathscr{V}$ send to $\mathscr{P}$ a binary vector $\mathbf{v} \in \mathbb{Z}_2$ that acts as a "bit selector" of the secret $\mathbf{s}$. The prover then picks $\mathbf{R}$ by

itself and computes noisy inner products of the form $\mathbf{R}^\mathsf{T} \cdot \mathbf{s}_{\downarrow \mathbf{v}} \oplus \mathbf{e}$. Curiously, allowing the verifier to choose which bits of $\mathbf{s}$ to use in each session is sufficient to prevent active attacks. We only need to add a few sanity-checks that no pathological $\mathbf{v}$ or $\mathbf{R}$ were sent by an active adversary. The protocol is detailed below.

`Public parameters.` The authentication protocol has the following public parameters, where τ, τ' are constants and n depends on the security parameter κ.

$\kappa \in \mathbb{N}$	length of the secret key $\mathbf{s} \in \mathbb{Z}_2^{2\kappa}$
$\tau \in]0, 1/2[$	parameter of the Bernoulli error distribution Bern_τ
$\tau' = 1/4 + \tau/2$	acceptance threshold
$n \in \mathbb{N}$	number of parallel repetitions (we require $n \leqslant \kappa/2$)

`Key Generation.` Algorithm $\mathsf{KGEN}(1^\kappa)$ samples $\mathbf{s} \xleftarrow{\$} \mathbb{Z}_2^{2\kappa}$ and returns $\mathbf{s}$ as the secret key.

`Authentication Protocol.` The 2-round authentication protocol with prover $\mathscr{P}_{\tau,n}$ and verifier $\mathscr{V}_{\tau',n}$ is given in Figure 4.5.

$\mathscr{P}_{\tau,n}(\mathbf{s} \in \mathbb{Z}_2^{2\kappa})$		$\mathscr{V}_{\tau',n}(\mathbf{s} \in \mathbb{Z}_2^{2\kappa})$
	$\xleftarrow{\mathbf{v}}$	$\mathbf{v} \xleftarrow{\$} \{\mathbf{x} \in \mathbb{Z}_2^{2\kappa} : w_\mathrm{H}(\mathbf{x}) = \kappa\}$
if $w_\mathrm{H}(\mathbf{v}) \neq \kappa$ abort		
$\mathbf{R} \xleftarrow{\$} \mathbb{Z}_2^{\kappa \times n}$; $\mathbf{e} \xleftarrow{\$} \mathsf{Bern}_\tau^n$		
$\mathbf{z} = \mathbf{R}^\mathsf{T} \cdot \mathbf{s}_{\downarrow \mathbf{v}} \oplus \mathbf{e} \in \mathbb{Z}_2^n$	$\xrightarrow{(\mathbf{R},\mathbf{z})}$	if $\mathrm{rank}(\mathbf{R}) \neq n$, reject
		if $w_\mathrm{H}(\mathbf{z} \oplus \mathbf{R}^\mathsf{T} \cdot \mathbf{s}_{\downarrow \mathbf{v}}) > n \cdot \tau'$, reject
		else accept

Figure 4.5: Two-round authentication protocol with active security from the LPN assumption.

We first prove that our authentication protocol has a negligible completeness error $\alpha \leqslant \alpha_{\tau,n} + 2^{-\kappa+n} = 2^{-\Theta(n)}$, where the term $\alpha_{\tau,\kappa}$ is given in Eq. (4.1). The verifier performs two checks. In the first verification step, $\mathscr{V}$ outputs reject if the random matrix $\mathbf{R}$ has not full rank. The lemma below shows this happens only with probability $\leqslant 2^{-\kappa+n}$.

Lemma 4.6 (Technical lemma). *For $n, d \in \mathbb{Z}$, let $\wp(n, d)$ denote the probability that a random matrix in $\mathbb{Z}_2^{n+d\times n}$ has rank less than n. Then,*

$$\wp(n, d) \leqslant 2^{-d}.$$

Proof. Assume we sample the columns of a random matrix in $\mathbb{Z}_2^{n+d\times n}$ one by one. For every $i = 1, \ldots, n$ let $\textsc{Indep}_i$ be the event that the first i columns are linearly independent. We have

$$\mathbb{P}\left[\neg\textsc{Indep}_i \mid \textsc{Indep}_{i-1}\right] = \frac{2^{i-1}}{2^{n+d}} = 2^{i-1-n-d},$$

since $\neg\textsc{Indep}_i$ happens if and only if the i-th column (sampled from a space of dimension 2^{n+d}) falls into the space (of size 2^{i-1}) spanned by the first $i-1$ columns. We conclude

$$\wp(n, d) = \mathbb{P}\left[\neg\textsc{Indep}_n\right] = \sum_{i=1}^{n} \mathbb{P}\left[\neg\textsc{Indep}_i \mid \textsc{Indep}_{i-1}\right] = \sum_{i=1}^{n} 2^{i-1-n-d} \leqslant 2^{-d}.$$

□

In the second verification step the verifier needs to check that $w_H(\mathbf{e}) \leqslant n \cdot \tau'$, where $\mathbf{e} = \mathbf{R}^T \cdot \mathbf{s} \oplus \mathbf{z}$ is the noise added by the prover $\mathscr{P}_{\tau,n}$. From Eq. (4.1), we know that this check will fail only with probability $\alpha_{\tau,n}$. This completes the proof of completeness.

We now prove active security of our protocol under the hardness of the $\mathsf{SLPN}^*_{\tau,2\kappa,d}$ problem (and hence under the LPN assumption, cf. Lemma 4.5), where $d = \kappa/(2+\gamma)$ for some constant $\gamma > 0$. (Concretely, $\gamma = 0.1$ should do for all practical purposes.)

Theorem 4.7 (Active security of protocol in Figure 4.5). *For any constant $\gamma > 0$, let $d = \kappa/(2+\gamma)$. If the $\mathsf{SLPN}^*_{\tau,2\kappa,d}$ problem is $(t_{\mathsf{SLPN}}, nQ, \epsilon_{\mathsf{SLPN}})$-hard, then the authentication protocol of Figure 4.5 is (t, Q, ϵ)-secure against* active *adversaries, where for constants $c_\gamma, c_\tau > 0$ that depend only on γ and τ respectively,*

$$t = t_{\mathsf{SLPN}} - \mathrm{poly}(Q, \kappa) \qquad \epsilon = \epsilon_{\mathsf{SLPN}} + Q \cdot 2^{-c_\gamma \cdot \kappa} + 2^{-c_\tau \cdot n} = \epsilon + 2^{-\Theta(n)}.$$

Before going into the proof, we give some intuition. It is quite natural to try reducing the security of the protocol to the hardness of the Subset LPN problem: Given an adversary $\mathscr{A}$ breaking the protocol we want to build a distinguisher $\mathscr{D}$ for the Subset LPN problem. The adversary $\mathscr{D}$ has access to an oracle $\mathscr{O}$ which

is either the $\Gamma^*_{\tau,2\kappa,d}(\mathbf{x},\cdot)$ (for some unknown $\mathbf{x} \xleftarrow{\$} \mathbb{Z}_2^{2\kappa}$) or the uniform oracle $U_{2\kappa+1}(\cdot)$, and it is asked to tell the two oracles apart. To do so, $\mathscr{D}$ will "fake" the environment for the adversary $\mathscr{A}$ who is expecting to attack the protocol as in an active attack.

Note that in principle, when $\mathscr{O}$ is the Subset LPN oracle, we can perfectly simulate the first phase of an active attack as follows. When $\mathscr{A}$ asks a query $\mathbf{v} \in \{\mathbf{y} \in \mathbb{Z}_2^{2\kappa} : w_H(\mathbf{y}) = \kappa\}$, we query $\mathscr{O}$ on input $\mathbf{v}$ for n times. Let $(\widehat{\mathbf{R}} \in \mathbb{Z}_2^{2\kappa\times n}, \mathbf{z} \in \mathbb{Z}_2^n)$ be the n outputs from the oracle. The pair $(\mathbf{R} = \widehat{\mathbf{R}}_{\downarrow\mathbf{v}}, \mathbf{z})$ is distributed exactly as in the protocol, where the secret is $\mathbf{s} = \mathbf{x}$. On the other hand when $\mathscr{O}$ is the uniform oracle, $\mathscr{A}$ sees just uniformly random vectors.

However, in the second phase of the simulation, we need to challenge $\mathscr{A}$ on a vector $\mathbf{v}^*$. Let $(\mathbf{R}^*, \mathbf{z}^*)$ be $\mathscr{A}$'s answer. The idea would be to distinguish the Subset LPN oracle from the uniform oracle based on the verification of this answer. The problem is that we are unable to verify whether this answer is correct or not, because we don't know the secret $\mathbf{x}$.

To solve this problem we build the reduction in such a way that, when $\mathscr{O}$ is a Subset LPN oracle $\Gamma^*_{\tau,2\kappa,d}(\mathbf{x},\cdot)$ with secret $\mathbf{x}$, the simulated answers have exactly the same distribution as the answers of a honest prover $\mathscr{P}_{\tau,n}(\mathbf{s} \in \mathbb{Z}_2^{2\kappa})$ where

$$\mathbf{s} = (\mathbf{x}^* \wedge \mathbf{v}^*) \oplus (\mathbf{x} \wedge \overline{\mathbf{v}}^*), \tag{4.3}$$

for randomly chosen $\mathbf{s}^*$, $\mathbf{v}^*$. Thus one part of $\mathbf{s}$'s bits come from $\mathbf{x}^*$, and the other part is from the unknown secret $\mathbf{x}$ (for which we use the oracle $\mathscr{O}$). In the second phase we give $\mathscr{A}$ the challenge $\mathbf{v}^*$. As $\mathbf{s}_{\downarrow\mathbf{v}^*} = (\mathbf{x}^* \wedge \mathbf{v}^*)_{\downarrow\mathbf{v}^*} = \mathbf{x}^*_{\downarrow\mathbf{v}^*}$ is known, we will be able to verify if $\mathscr{A}$'s answer is correct and to finalise the simulation.

On the other hand, if $\mathscr{O}$ is the uniform oracle $U_{2\kappa+1}(\cdot)$, then after the first phase $\mathbf{x}^* \wedge \mathbf{v}^*$ is information theoretically hidden, and thus $\mathscr{A}$ cannot come up with a valid forgery but with exponentially small probability.

Proof. For a constant γ, let $d = \kappa/(2+\gamma)$ as in the theorem. We define a few terms that we will need later in the security proof. As in Eq. (4.2) we let $\alpha'_{\tau',n}$ denote the probability that a random bitstring $\mathbf{y} \xleftarrow{\$} \mathbb{Z}_2^n$ has Hamming weight $w_H(\mathbf{y}) \leqslant n \cdot \tau'$. Recall that the Chernoff bound (cf. Theorem 2.3) implies that there exists a constant $c_\tau > 0$ (only depending on τ) such that $\alpha'_{\tau',n} \leqslant 2^{-c_\tau \cdot n}$.

Moreover, let $\alpha''_{\kappa,d}$ denote the probability that a random substring of length κ chosen from a string of length 2κ with Hamming weight κ, has a Hamming weight less than d. Using the fact that the expected Hamming weight is $\kappa/2 =$

$d(1+\gamma/2) = d(1+\Theta(1))$, one can show that there exists a constant $c_\gamma > 0$ (only depending on γ), such that

$$\alpha''_{\kappa,d} = \frac{\sum_{i=0}^{d-1} \binom{\kappa}{i}\binom{\kappa}{\kappa-i}}{\binom{2\kappa}{\kappa}} \leqslant 2^{-c_\gamma \cdot \kappa}. \tag{4.4}$$

Since we already discussed the intuition, we now proceed with the formal proof. Given a PPT adversary $\mathscr{A}$ breaking active security of the protocol with advantage ϵ, we build an efficient distinguisher $\mathscr{D}$ for the Subset LPN problem with advantage greater than ϵ_{SLPN}, where ϵ_{SLPN} is as in the theorem statement (contradiction). The adversary $\mathscr{D}$ has access to an oracle $\mathscr{O}$ which is either $\Gamma^*_{\tau,2\kappa,d}(\mathbf{x},\cdot)$ or $U_{2\kappa+1}(\cdot)$ and has to tell the two cases apart. The algorithm for $\mathscr{D}^{\mathscr{O}}$ is given below.

Setup. Initially $\mathscr{D}^{\mathscr{O}}$ samples

$$\mathbf{x}^* \xleftarrow{\$} \mathbb{Z}_2^{2\kappa} \qquad\qquad \mathbf{v}^* \xleftarrow{\$} \{\mathbf{y} \in \mathbb{Z}_2^{2\kappa} : \ w_H(\mathbf{y}) = \kappa\}.$$

First phase. In the first phase $\mathscr{D}^{\mathscr{O}}$ invokes $\mathscr{A}$ who expects access to $\mathscr{P}_{\tau,n}(\mathbf{s} \in \mathbb{Z}_2^{2\kappa})$. We now specify how $\mathscr{D}^{\mathscr{O}}$ samples the answer $(\mathbf{R},\mathbf{z})$ to a query $\mathbf{v} \in \{\mathbf{y} \in \mathbb{Z}_2^{2\kappa} : \ w_H(\mathbf{y}) = \kappa\}$ made by $\mathscr{A}$. Let

$$\mathbf{u}^* = \mathbf{v} \wedge \mathbf{v}^* \qquad\qquad \mathbf{u} = \mathbf{v} \wedge \overline{\mathbf{v}}^*.$$

1. $\mathscr{D}^{\mathscr{O}}$ queries its oracle n times on the input $\mathbf{u}$. If the oracle's output is $\perp$ (which happens if and only if $w_H(\mathbf{u}) < d$), $\mathscr{D}^{\mathscr{O}}$ outputs 0 and stops. Otherwise let $\widehat{\mathbf{R}}_1 \in \mathbb{Z}_2^{2\kappa\times n}, \mathbf{z}_1 \in \mathbb{Z}_2^n$ denote the n outputs of the oracle.
2. Sample $\widehat{\mathbf{R}}_0 \xleftarrow{\$} \mathbb{Z}_2^{2\kappa\times n}$ and set $\mathbf{z}_0 = \widehat{\mathbf{R}}_0^{\mathsf{T}} \cdot (\mathbf{x}^* \wedge \mathbf{u}^*)$.
3. Return $(\mathbf{R} = \widehat{\mathbf{R}}_{\downarrow \mathbf{v}} \in \mathbb{Z}_2^{\kappa\times n}, \mathbf{z} = \mathbf{z}_0 \oplus \mathbf{z}_1 \in \mathbb{Z}_2^n)$, where $\widehat{\mathbf{R}}$ is uniquely determined by requiring $\widehat{\mathbf{R}}_{\downarrow \mathbf{v}^*} = \widehat{\mathbf{R}}_0$ and $\widehat{\mathbf{R}}_{\downarrow \overline{\mathbf{v}}^*} = \widehat{\mathbf{R}}_1$.

Second phase. Eventually, $\mathscr{A}$ enters the second phase of the active attack, expecting a challenge from $\mathscr{V}_{\tau',n}(\mathbf{s} \in \mathbb{Z}_2^{2\kappa})$.

1. $\mathscr{D}^{\mathscr{O}}$ forwards $\mathbf{v}^*$ as the challenge to $\mathscr{A}$.
2. $\mathscr{A}$ answers with some $(\mathbf{R}^*, \mathbf{z}^*)$.
3. $\mathscr{D}^{\mathscr{O}}$ checks if

$$\operatorname{rank}(\mathbf{R}^*) = n \quad \text{and} \quad w_H(\mathbf{z}^* \oplus \mathbf{R}^{*\mathsf{T}} \cdot \mathbf{x}^*_{\downarrow \mathbf{v}^*}) \leqslant n \cdot \tau'. \tag{4.5}$$

The output is 1 if both checks succeed and 0 otherwise.

Of course $\mathscr{D}$ runs in polynomial time $t_{\mathsf{SLPN}} \approx t$. Moreover if $\mathscr{A}$ asks Q queries in the first phase of an active attack, $\mathscr{D}$ queries the oracle nQ times. We distinguish two cases depending on the $\mathscr{O}$ oracle.

Claim 1. *If* $\mathscr{O} = \Gamma^*_{\tau,2\kappa,d}(\mathbf{x},\cdot)$, *then* $\mathscr{D}^{\mathscr{O}}$ *outputs* 1 *with probability* $\epsilon - Q \cdot \alpha''_{\kappa,d}$.

Proof of Claim. We start by showing that $\mathscr{D}$ outputs 1 with probability $\geqslant \epsilon$ if the Subset LPN oracle accepts subsets of arbitrary small size (and does not simply output $\perp$ on inputs $\mathbf{v}$ where $w_H(\mathbf{v}) < d$), i.e.,

$$\mathbb{P}\left[\mathscr{D}^{\Gamma^*_{\tau,2\kappa,0}(\mathbf{x},\cdot)} = 1\right] \geqslant \epsilon. \tag{4.6}$$

Then we'll upper bound the gap between the probability that $\mathscr{D}$ outputs 1 in the above case and the probability that $\mathscr{D}$ outputs 1 when given access to the oracle that we are interested in as:

$$\left|\mathbb{P}\left[\mathscr{D}^{\Gamma^*_{\tau,2\kappa,d}(\mathbf{x},\cdot)} = 1\right] - \mathbb{P}\left[\mathscr{D}^{\Gamma^*_{\tau,2\kappa,0}(\mathbf{x},\cdot)} = 1\right]\right| \leqslant Q \cdot \alpha''_{\kappa,d}. \tag{4.7}$$

Eq. (4.6) holds as:

- The answers $(\mathbf{R},\mathbf{z})$ that $\mathscr{D}^{\Gamma^*_{\tau,2\kappa,0}(\mathbf{x},\cdot)}$ gives to $\mathscr{A}$'s queries in the first phase of the attack have *exactly* the same distribution as what $\mathscr{A}$ would get when interacting with an honest prover $\mathscr{P}_{\tau,n}(\mathbf{s} \in \mathbb{Z}_2^{2\kappa})$ where the "simulated" secret $\mathbf{s}$ is defined in Eq. (4.3).

  To see this, recall that on a query $\mathbf{v}$ from $\mathscr{A}$, the distinguisher $\mathscr{D}^{\Gamma^*_{\tau,2\kappa,0}(\mathbf{x},\cdot)}$ must answer with $(\mathbf{R}, \mathbf{z} = \mathbf{R}^T \cdot \mathbf{s}_{\downarrow\mathbf{v}} \oplus \mathbf{e})$, where $\mathbf{R} = \widehat{\mathbf{R}}_{\downarrow\mathbf{v}}$ is the compressed form of $\widehat{\mathbf{R}}$. In the first step, $\mathscr{D}$ queries its oracle with $\mathbf{u} = \mathbf{v} \wedge \overline{\mathbf{v}}^*$ and obtains noisy inner products $(\widehat{\mathbf{R}}_1, \mathbf{z}_1)$ with the part of $\mathbf{s}_{\downarrow\mathbf{v}}$ that contains only bits from $\mathbf{x}$, i.e.,

  $$\mathbf{z}_1 = \widehat{\mathbf{R}}_1^T \cdot (\mathbf{x} \wedge \mathbf{u}) \oplus \mathbf{e} = \widehat{\mathbf{R}}_1^T \cdot (\mathbf{s} \wedge \mathbf{u}) \oplus \mathbf{e}.$$

  In the second step, $\mathscr{D}$ samples n inner products $(\widehat{\mathbf{R}}_0, \mathbf{z}_0)$ (with no noise) with the part of $\mathbf{s}_{\downarrow\mathbf{v}}$ that contains only bits from the known $\mathbf{x}^*$, i.e.,

  $$\mathbf{z}_0 = \widehat{\mathbf{R}}_0^T \cdot (\mathbf{x}^* \wedge \mathbf{u}^*) = \widehat{\mathbf{R}}_0^T \cdot (\mathbf{s} \wedge \mathbf{u}^*).$$

In the third step, $\mathscr{D}$ then generates $(\widehat{\mathbf{R}}, \widehat{\mathbf{R}}^{\mathsf{T}} \cdot (\mathbf{s} \wedge \mathbf{v}) \oplus \mathbf{e})$ from the previous values where $\widehat{\mathbf{R}}$ is defined by $\widehat{\mathbf{R}}_{\downarrow \mathbf{v}^*} = \widehat{\mathbf{R}}_0$ and $\widehat{\mathbf{R}}_{\downarrow \overline{\mathbf{v}}^*} = \widehat{\mathbf{R}}_1$. Using $\mathbf{v} = \mathbf{u} \oplus \mathbf{u}^*$, we get

$$\begin{aligned}\mathbf{z} &= \mathbf{z}_0 \oplus \mathbf{z}_1 \\ &= \widehat{\mathbf{R}}_0^{\mathsf{T}} \cdot (\mathbf{s} \wedge \mathbf{u}^*) \oplus \widehat{\mathbf{R}}_1^{\mathsf{T}} \cdot (\mathbf{s} \wedge \mathbf{u}) \oplus \mathbf{e} \\ &= \widehat{\mathbf{R}}^{\mathsf{T}} \cdot (\mathbf{s} \wedge \mathbf{v}) \oplus \mathbf{e} \\ &= \mathbf{R}^{\mathsf{T}} \cdot \mathbf{s}_{\downarrow \mathbf{v}} \oplus \mathbf{e}.\end{aligned}$$

- The challenge $\mathbf{v}^*$ sent to $\mathscr{A}$ in the second phase of the active attack is uniformly random (even given the entire view so far), and therefore has the same distribution as a challenge in an active attack.

- $\mathscr{D}^{\Gamma^*_{\tau,2\kappa,0}(\mathbf{x},\cdot)}$ outputs 1 if Eq. (4.5) holds, which is exactly the case when $\mathscr{A}$'s response to the challenge was valid. By assumption this probability is at least ϵ.

It remains to prove Eq. (4.7). Note that $\Gamma^*_{\tau,2\kappa,0}(\mathbf{x},\cdot)$ behaves exactly like $\Gamma^*_{\tau,2\kappa,d}(\mathbf{x},\cdot)$ as long as one never makes a query $\mathbf{v}$ where $w_H(\mathbf{v} \wedge \overline{\mathbf{v}}^*) < d$. Since $\mathbf{v}^* \xleftarrow{\$} \{\mathbf{y} \in \mathbb{Z}_2^{2\kappa} : w_H(\mathbf{y}) = \kappa\}$, for any $\mathbf{v}$, the probability that $w_H(\mathbf{v} \wedge \overline{\mathbf{v}}^*) < d$ is (by definition) $\alpha''_{\kappa,d}$ as defined in Eq. (4.2). Using the union bound, we can upper bound the probability that $w_H(\mathbf{v} \wedge \overline{\mathbf{v}}^*) < d$ for any of the Q different $\mathbf{v}$'s chosen by the adversary as $Q \cdot \alpha''_{\kappa,d}$.

It thus follows from the triangle inequality applied to Eq. (4.6) and Eq. (4.7) that

$$\mathbb{P}\left[\mathscr{D}^{\Gamma^*_{\tau,2\kappa,d}(\mathbf{x},\cdot)} = 1\right] \geqslant \epsilon - Q \cdot \alpha''_{\kappa,d}. \tag{4.8}$$

□

Claim 2. *If $\mathscr{O} = U_{2\kappa+1}(\cdot)$, then $\mathscr{D}^{\mathscr{O}}$ outputs 1 with probability $\leqslant \alpha'_{\tau',n}$.*

Proof of Claim. If $\mathbf{R}^*$ does not have full rank then $\mathscr{D}$ outputs 0 by definition. Therefore, we now consider the case where $\operatorname{rank}(\mathbf{R}^*) = n$. The answers $(\mathbf{R}, \mathbf{z})$ that the adversary $\mathscr{A}$ obtains from $\mathscr{D}^{U_{2\kappa+1}(\cdot)}$ are independent of $\mathbf{x}^*$ (i.e., $\mathbf{z} = \mathbf{z}_0 \oplus \mathbf{z}_1$ is uniform as $\mathbf{z}_1$ is uniform). Since $\mathbf{x}^*_{\downarrow \mathbf{v}^*}$ is uniformly random and $\mathbf{R}^*$ has full rank, the vector

$$\mathbf{y} = \mathbf{R}^{*\mathsf{T}} \cdot \mathbf{x}^*_{\downarrow \mathbf{v}^*} \oplus \mathbf{z}^*$$

is uniformly random over $\mathbb{Z}_2^n$. Thus the probability that the second verification in Eq. (4.5) does not fail is $\mathbb{P}[w_H(\mathbf{y}) \leqslant n \cdot \tau'] = \alpha'_{\tau',n}$. We conclude

$$\mathbb{P}\left[\mathscr{D}^{U_{2\kappa+1}(\cdot)} = 1\right] \leqslant \alpha'_{\tau',n}. \tag{4.9}$$

□

Putting together Eq. (4.8) and Eq. (4.9) we get

$$\left|\mathbb{P}\left[\mathscr{D}^{\Gamma^*_{\tau,2\kappa,d}(\mathbf{x},\cdot)} = 1\right] - \mathbb{P}\left[\mathscr{D}^{U_{2\kappa+1}(\cdot)} = 1\right]\right| \geqslant \epsilon - Q \cdot \alpha''_{\kappa,d} - \alpha'_{\tau',n} = \epsilon_{\mathsf{SLPN}},$$

against the assumption that the $\Gamma^*_{\tau,2\kappa,d}$ problem is $(t_{\mathsf{SLPN}}, nQ, \epsilon_{\mathsf{SLPN}})$-hard. □

$\mathscr{P}_{\tau,n}(\mathbf{s}, \mathbf{b}^{\mathbf{v}}, \mathbf{b}^{\mathbf{z}})$		$\mathscr{V}_{\tau',n}(\mathbf{s}, \mathbf{b}^{\mathbf{v}}, \mathbf{b}^{\mathbf{z}})$
	$\xleftarrow{\mathbf{v}}$	$\mathbf{v} \xleftarrow{\$} \mathbb{Z}_2^{2\kappa}$
$\mathbf{R} \xleftarrow{\$} \mathbb{Z}_2^{2\kappa \times n}$; $\mathbf{e} \xleftarrow{\$} \mathsf{Bern}_\tau^n$		
$\mathbf{z} = \mathbf{R}^\mathsf{T} \cdot (\mathbf{s} \wedge (\mathbf{v} \oplus \mathbf{b}^{\mathbf{v}})) \oplus \mathbf{b}^{\mathbf{z}} \oplus \mathbf{e}$	$\xrightarrow{\mathbf{z},\mathbf{R}}$	if $w_H(\mathbf{R}^\mathsf{T} \cdot (\mathbf{s} \wedge (\mathbf{v} \oplus \mathbf{b}^{\mathbf{v}})) \oplus \mathbf{b}^{\mathbf{z}}) > n \cdot \tau'$ reject otherwise accept

Figure 4.6: By blinding the values $\mathbf{v}, \mathbf{z}$ with secret random vectors $\mathbf{b}^{\mathbf{v}}, \mathbf{b}^{\mathbf{z}} \in \mathbb{Z}_2^{2\kappa}$ we can avoid checking whether $w_H(\mathbf{v}) = \kappa$ and $\mathrm{rank}(\mathbf{R}) = n$ as in the protocol from Figure 4.5.

AVOID CHECKING. One disadvantage of the protocol in Figure 4.5, compared to HB style protocols, is the necessity to check whether the messages exchanged have the right from: the prover checks if $\mathbf{v}$ has weight κ, while the verifier must make the even more expensive check whether $\mathbf{R}$ has full rank. Eliminating such verification procedures can be particularly useful if for example the prover is an RFID chip where even the simple verification that a vector has large weight is expensive. We note that it is possible to eliminate these checks by blinding the exchanged messages $\mathbf{v}$ and $\mathbf{z}$ using random vectors $\mathbf{b}^{\mathbf{v}} \in \mathbb{Z}_2^{2\kappa}$ and $\mathbf{b}^{\mathbf{z}} \in \mathbb{Z}_2^n$ respectively, as shown in Figure 4.6. The security and completeness of this protocol is basically the same as for the protocol in Figure 4.5. The security proof is also very similar and is therefore omitted.

4.4 A MAC FROM LPN

In this section we show how to adapt the protocol from Figure 4.5 to get a MAC. Recall that this immediately implies a round optimal authentication protocol secure against MiM attacks (cf. Theorem 4.2). Notably, this is also the first construction of a MAC from the LPN assumption.[5]

As a first attempt, let us try to view our authentication protocol as a MAC. That is, a MAC tag is of the form $\phi = (\mathbf{R}, \mathbf{z} = \mathbf{R}^\mathsf{T} \cdot f_\mathbf{s}(\mathbf{m}) \oplus \mathbf{e})$, where the secret key derivation function $f_\mathbf{s}(\mathbf{m}) \in \mathbb{Z}_2^\kappa$ first uniquely encodes the message $\mathbf{m}$ into $\mathbf{v} \in \mathbb{Z}_2^{2\kappa}$ of weight κ and then returns $\mathbf{s}_{\downarrow\mathbf{v}}$ by selecting κ bits from secret $\mathbf{s}$, according to $\mathbf{v}$. However, this MAC is not secure: given a MAC tag $\phi = (\mathbf{R}, \mathbf{z})$ an adversary can ask verification queries where it sets individual rows of $\mathbf{R}$ to zero until verification fails: if the last row set to zero was the i-th, then the i-th bit of $f_\mathbf{s}(\mathbf{m})$ must be 1.[6] (In fact, the main technical difficulty to build a secure MAC from LPN is to make sure the secret $\mathbf{s}$ does not leak from verification queries.) Our solution is to randomise the mapping f, i.e. use $f_\mathbf{s}(\mathbf{m}, \boldsymbol{\omega})$ for some randomness $\boldsymbol{\omega}$ and compute the tag as $\phi = \pi(\mathbf{R}, \mathbf{R}^\mathsf{T} \cdot f_\mathbf{s}(\mathbf{m}, \boldsymbol{\omega}) \oplus \mathbf{e}, \boldsymbol{\omega})$, where π is a pairwise independent permutation (contained in the secret key). We can prove that if LPN is hard then this construction yields a secure MAC. (The key argument is that, with high probability, all non-trivial verification queries are inconsistent and hence lead to reject.) However, the security reduction to the LPN problem is quite loose since it has to guess the value $\mathbf{v}$ from the adversary's forgery. (In the context of identity-based encryption (IBE) a similar idea has been used to go from selective-ID to full security using "complexity leveraging" [BB04].) In our case, however, this still leads to a polynomial security reduction when one commits to the hardness of the LPN problem at the time of the construction.

To get a strictly polynomial security reduction (without having to commit to the hardness of the LPN problem), in our second construction we adapt a technique originally used by Waters [Wat05] in the context of IBE schemes that has been applied to lattice based signature [Boy10] and encryption schemes [ABB10]. Concretely, we instantiate the above MAC construction with a different secret key derivation function $f_\mathbf{s}(\mathbf{m}, \boldsymbol{\omega}) = \mathbf{s}_0 \oplus \bigoplus_{i:\mathbf{v}[i]=1} \mathbf{s}_i$ (where $\mathbf{v} = h(\mathbf{m}, \boldsymbol{\omega})$ and $h(\cdot)$ is a pairwise independent hash). The drawback of our second construc-

5 Prior to our work, the only known way to construct an LPN-based MAC was via a relatively inefficient generic transformation [GGM86] (that works with any pseudorandom generator). See also Section 4.5 for a discussion on this point.

6 Note that a similar strategy yields a MiM attack against the protocol from the previous section, showing that this protocol is not MiM-secure.

tion is the larger key-size. Our security reduction uses a technique from [Boy10; ABB10] based on encodings with full-rank differences (FRD) by Cramer and Damgard [CD09].

FIRST CONSTRUCTION. Our first MAC is based on the 2-round authentication protocol from Section 4.3. We prove that if the LPN problem is ϵ-hard, then no adversary asking Q queries can forge a MAC with probability more than $\Theta(\sqrt{\epsilon} \cdot Q)$.

Recall the 2-round authentication protocol from Figure 4.5. In the protocol the verifier chooses a random challenge subset $\mathbf{v}$. To turn this interactive protocol into a MAC, we will compute this $\mathbf{v}$ from the message $\mathbf{m}$ to be authenticated as $\mathbf{v} = C(h(\mathbf{m}, \boldsymbol{\omega}))$, where h is a pairwise independent hash function,[7] $\boldsymbol{\omega} \in \mathbb{Z}_2^\nu$ is some fresh randomness and C is some encoding scheme. The code C is fixed and public, while the function h is part of the secret key.

The authentication tag ϕ is computed in the same manner as the prover's answer in the authentication protocol. That is, we sample a random matrix $\mathbf{R} \in \mathbb{Z}_2^{\kappa \times n}$ and compute a noisy inner product $\mathbf{z} = \mathbf{R}^\intercal \cdot \mathbf{s}_{\downarrow \mathbf{v}} \oplus \mathbf{e}$, where $\mathbf{e} \xleftarrow{\$} \mathrm{Bern}_\tau^n$. As observed before, using $(\mathbf{R}, \mathbf{z})$ as an authentication tag would not be secure, and we need to blind these values. This is done by applying an (almost) pairwise independent permutation (PIP) π—which is part of the secret key—to $(\mathbf{R}, \mathbf{z}, \boldsymbol{\omega}) \in \mathbb{Z}_2^{\kappa \times n + n + \nu}$.

The message authentication code $\Pi_M^1 = \{\mathsf{KGEN}, \mathsf{TAG}, \mathsf{VRFY}\}$ with associated message space $\mathcal{M}$ is defined as follows.

- `Public parameters.` Π_M^1 has the following public parameters.

κ, τ, τ', n	as in the authentication protocol from Figure 4.5
$\mu \in \mathbb{N}$	output length of the hash function
$\nu \in \mathbb{N}$	length of the randomness
$C: \mathbb{Z}_2^\mu \to \mathbb{Z}_2^{2\kappa}$	encoding, where $\forall\, \mathbf{y} \neq \mathbf{y}' \in \mathbb{Z}_2^\mu$ we have $w_H(C(\mathbf{y})) = \kappa$ and $w_H(C(\mathbf{y}) \oplus C(\mathbf{y}')) \geqslant 0.9\kappa$.

7 A family $\mathcal{H} = \{h : \mathcal{D} \to \mathcal{R}\}$ is a pairwise independent hash function family if $\forall\, x, y \in \mathcal{D}$ (with $x \neq y$) and $\forall\, \alpha, \beta \in \mathcal{R}$

$$\mathbb{P}[h(x) = \alpha \wedge h(y) = \beta] = \frac{1}{(|\mathcal{R}|)^2}$$

For example, when $\mathcal{D} = \mathcal{R} = \mathbb{Z}_{2^\kappa}$ the following is a family of pairwise independent permutation (PIP)

$$\mathcal{H} = \{\pi_{a,b}(x) = a \cdot x + b : a, b \xleftarrow{\$} \mathbb{Z}_{2^\kappa}\}.$$

Truncating the output at any $\kappa' < \kappa$ we have a pairwise independent hash function.

- Key generation. Algorithm $\mathsf{KGEN}(1^\kappa)$ samples $\mathbf{s} \xleftarrow{\$} \mathbb{Z}_2^{2\kappa}$, an (almost) pairwise independent hash function $h : \mathcal{M} \times \mathbb{Z}_2^\nu \to \mathbb{Z}_2^\mu$ and a pairwise independent permutation π over $\mathbb{Z}_2^{\kappa \times n+n+\nu}$. It returns $K = (\mathbf{s}, h, \pi)$ as the secret key.

- Tagging. Given secret key $K = (\mathbf{s}, h, \pi)$ and message $\mathbf{m} \in \mathcal{M}$, algorithm TAG proceeds as follows.
  1. $\mathbf{R} \xleftarrow{\$} \mathbb{Z}_2^{\kappa \times n}$, $\boldsymbol{\omega} \xleftarrow{\$} \mathbb{Z}_2^\nu$, $\mathbf{e} \xleftarrow{\$} \mathsf{Bern}_\tau^n$
  2. $\mathbf{v} = C(h(\mathbf{m}, \boldsymbol{\omega})) \in \mathbb{Z}_2^{2\kappa}$
  3. Return $\phi = \pi(\mathbf{R}, \mathbf{R}^\mathsf{T} \cdot \mathbf{s}_{\downarrow \mathbf{v}} \oplus \mathbf{e}, \boldsymbol{\omega})$

- Verification. On input a secret-key $K = (\mathbf{s}, h, \pi)$, message $\mathbf{m} \in \mathcal{M}$ and tag ϕ, algorithm VRFY proceeds as follows.
  1. Parse $\pi^{-1}(\phi)$ as $(\mathbf{R} \in \mathbb{Z}_2^{\kappa \times n}, \mathbf{z} \in \mathbb{Z}_2^n, \boldsymbol{\omega} \in \mathbb{Z}_2^\nu)$. If $\mathrm{rank}(\mathbf{R}) \neq n$, then return reject
  2. $\mathbf{v} = C(h(\mathbf{m}, \boldsymbol{\omega}))$
  3. If $w_H(\mathbf{z} \oplus \mathbf{R}^\mathsf{T} \cdot \mathbf{s}_{\downarrow \mathbf{v}}) > n \cdot \tau'$ return reject, otherwise return accept

The code C can be constructed as follows. We first sample a random matrix $\mathbf{C} \in \mathbb{Z}_2^{\mu \times \kappa}$ and map $\mathbf{y} \in \mathbb{Z}_2^\mu$ in $C(\mathbf{y}) = (\mathbf{c} \in \mathbb{Z}_2^\kappa, \mathbf{c}' \in \mathbb{Z}_2^\kappa)$ where $\mathbf{c} = \mathbf{C}^\mathsf{T} \cdot \mathbf{y}$ and $\mathbf{c}'$ is the "inverse" of $\mathbf{c}$. A random code $\mathbf{C}$ has high distance with high probability and $C(\mathbf{y}) = (\mathbf{c}, \mathbf{c}')$ has weight exactly κ.

Using an argument similar to the one used to derive the completeness error of the protocol in Figure 4.5, it is easy to show that Π_M^1 has completeness error $2^{-c_\tau \cdot n}$, where $c_\tau > 0$ depends only on τ.

Theorem 4.8 (Security of Π_M^1). *For $\mu = \nu \in \mathbb{N}$, a constant $\gamma > 0$ and $d = \kappa/(2+\gamma)$, if the $\mathsf{SLPN}^*_{\tau,2\kappa,d}$ problem is $(t_{\mathsf{SLPN}}, nQ, \epsilon_{\mathsf{SLPN}})$-hard then Π_M^1 is (t, Q, ϵ)-secure against uf-cma adversaries, where*

$$t \approx t_{\mathsf{SLPN}} \qquad \epsilon_{\mathsf{SLPN}} = \min\left\{ \frac{\epsilon}{2} - \frac{Q^2}{2^{\mu-2}} \,,\, \frac{\epsilon}{2^{\mu+1}} - 2^{-\Theta(n)} \right\}.$$

We now give an intuition for the proof of Theorem 4.8. Every query $(\mathbf{m}, \phi)$ to VRFY and query $\mathbf{m}$ to TAG defines a subset $\mathbf{v}$ (as computed in the second step in the definitions of both VRFY and TAG). We say that a forgery $(\mathbf{m}, \phi)$ is "fresh" if the $\mathbf{v}$ contained in $(\mathbf{m}, \phi)$ is different from all $\mathbf{v}$'s contained in all the previous VRFY and TAG queries. The proof makes a case distinction and uses a different reduction for the two cases where the forgery found by the adversary is more

likely to be fresh, or more likely to be non-fresh. In both cases we consider a reduction $\mathscr{D}^{\mathcal{O}}$ which has access to either a uniform oracle $\mathcal{O} = U$ or a Subset LPN oracle $\mathcal{O} = \Gamma^*$. The distinguisher $\mathscr{D}^{\mathcal{O}}$ uses an adversary $\mathscr{A}$ who can find forgeries for the MAC to distinguish those cases and thus break the Subset LPN assumption.

In the first case, where the first forgery is likely to be *non-fresh*, we can show (using the fact that a pairwise independent permutation is used to blind the tag) that if $\mathscr{D}^{\mathcal{O}}$'s oracle is $\mathcal{O} = U$, even a computationally unbounded $\mathscr{A}$ cannot come up with a message/tag pair $(\mathbf{m}, \phi)$ that contains a non-fresh $\mathbf{v}$. Thus we can distinguish the cases $\mathcal{O} = U$ and $\mathcal{O} = \Gamma^*$ by just observing if $\mathscr{A}$ ever makes a VRFY query $(\mathbf{m}, \phi)$ that contains a non-fresh $\mathbf{v}$ (even without being able to tell if $(\mathbf{m}, \phi)$ is valid or not).

If the forgery found by $\mathscr{A}$ is more likely to be *fresh*, we can use a similar argument as in the proof of our authentication protocol in Section 4.3. An additional difficulty here is that the reduction has to guess the fresh $\mathbf{v} \in \mathbb{Z}_2^{\mu}$ contained in the first forgery and cannot choose it as in the protocol. This is the reason why the reduction looses a factor 2^{μ}.

Proof. As in the theorem statement, we set $\mu = \nu$ (but for clarity we will keep the different letters μ for the range of h and ν for the length of the randomness). Let $\mathscr{A}$ be an adversary that (t, Q, ϵ)-breaks the uf-cma security of Π_M^1. Let Q_{tag} and Q_{vrfy} denote the number of queries that $\mathscr{A}$ makes to the oracles $\mathsf{TAG}_K(\cdot)$ and $\mathsf{VRFY}_K(\cdot,\cdot)$ respectively, such that $Q = Q_{\mathsf{tag}} + Q_{\mathsf{vrfy}}$. We assume that $\mathscr{A}$ never makes the same VRFY query twice (since VRFY is deterministic, repeating queries gives no additional information to $\mathscr{A}$) and also that she never makes a VRFY query $(\mathbf{m}, \phi)$ where ϕ was received as the output from TAG on input $\mathbf{m}$. Since the completeness error of Π_M^1 is $2^{-\Theta(n)}$, this is basically without loss of generality (as the answer would almost certainly be accept).

Every query $(\mathbf{m}, \phi)$ to VRFY and query $\mathbf{m}$ to TAG defines a subset $\mathbf{v}$ (as computed in the second step in the definitions of both VRFY and TAG). By definition, in the uf-cma experiment, with probability ϵ the adversary $\mathscr{A}$ at some point makes a VRFY query $(\mathbf{m}, \phi)$ where: (i) ϕ was not received as output on a TAG query $\mathbf{m}$, and (ii) $\mathsf{VRFY}_K(\mathbf{m}, \phi) = \mathsf{accept}$. We say that such a forgery $(\mathbf{m}, \phi)$ is "fresh" if the $\mathbf{v}$ defined by $(\mathbf{m}, \phi)$ is different from all $\mathbf{v}$'s defined by all the previous VRFY and TAG queries.

Let FRESH denote the event that $\mathscr{A}$ finds a fresh forgery. As $\mathscr{A}$ finds a forgery with probability ϵ and every forgery must be either fresh or not, we have that:

$$\mathbb{P}[\text{FRESH}] + \mathbb{P}[\neg\text{FRESH}] = \epsilon.$$

We will consider the two cases where $\mathbb{P}[\text{FRESH}] > \epsilon/2$ and $\mathbb{P}[\text{FRESH}] \leqslant \epsilon/2$ separately.

THE CASE $\mathbb{P}[\text{FRESH}] \leqslant \epsilon/2$. Given $\mathscr{A}$, we will construct an adversary $\mathscr{D}_1^{\mathscr{O}}$ who can distinguish $\mathscr{O} = \Gamma^*_{\tau,2\kappa,d}(\mathbf{x},\cdot)$ from $\mathscr{O} = U_{2\kappa+1}(\cdot)$ (as in Definition 4.2) with advantage[8]

$$\epsilon/2 - \frac{Q^2}{2^{\mu-2}}. \tag{4.10}$$

Setup. The adversary $\mathscr{D}_1^{\mathscr{O}}$ samples π, h (but not $\mathbf{s}$) as defined by KGEN. Next, it invokes $\mathscr{A}$ (who expects to attack Π^1_M with a key $(\mathbf{s}, h, \pi)$) answering its queries as follows.

Tag queries. If $\mathscr{A}$ makes a TAG query $\mathbf{m}$, then $\mathscr{D}_1^{\mathscr{O}}$ does the following:

1. Sample $\boldsymbol{\omega} \xleftarrow{\$} \mathbb{Z}_2^v$ and compute $\mathbf{v} = C(h(\mathbf{m}, \boldsymbol{\omega}))$.
2. Query the oracle $\mathscr{O}$ for n times on input $\mathbf{v}$. For $i = 1, \ldots, n$ let $(\mathbf{R}[i], \mathbf{z}[i]) \xleftarrow{\$} \mathscr{O}(\mathbf{v})$.
3. Return $\phi = \pi(\mathbf{R}, \mathbf{z}, \boldsymbol{\omega})$ where $\mathbf{R} = (\mathbf{R}[1], \ldots, \mathbf{R}[n])$ and $\mathbf{z} = (\mathbf{z}[1], \ldots, \mathbf{z}[n])$ to $\mathscr{A}$.

Verification queries. If $\mathscr{A}$ makes a VRFY query $(\mathbf{m}, \phi)$, then $\mathscr{D}_1^{\mathscr{O}}$ simply answers with reject.

If any TAG or VRFY query contains a $\mathbf{v}$ which has appeared in a previous query, $\mathscr{D}_1^{\mathscr{O}}$ outputs 1 and 0 otherwise. (Note that $\mathscr{D}_1^{\mathscr{O}}$ can compute the value $\mathbf{v}$ in a VRFY query as it knows π, h.)

Claim 3. *If $\mathscr{O} = \Gamma^*_{\tau,2\kappa,d}(\mathbf{x},\cdot)$, then $\mathscr{D}_1^{\mathscr{O}}$ outputs 1 with probability $\geqslant \epsilon/2$.*

Proof of Claim. The answers to the TAG queries of $\mathscr{A}$ computed by $\mathscr{D}_1^{\mathscr{O}}$ have exactly the same distribution as in the uf-cma experiment (where the secret key is $K = (\mathbf{s}, h, \pi)$ and $\mathbf{s} = \mathbf{x}$ is the secret of the Subset LPN oracle.). The answers to the VRFY queries (which are always reject) are correct as long as $\mathscr{A}$ does not query a valid forgery. From our assumption, the probability that $\mathscr{A}$ finds a valid forgery that is *not* fresh is $> \epsilon/2$, which is thus a lower bound on the probability that $\mathscr{D}_1^{\mathscr{O}}$ outputs 1. □

Claim 4. *If $\mathscr{O} = U_{2\kappa+1}(\cdot)$, then $\mathscr{D}_1^{\mathscr{O}}$ outputs 1 with probability $< Q^2/2^{\mu-2}$.*

8 In this case where $\mathbb{P}[\text{FRESH}] \leqslant \epsilon/2$, we can even distinguish a $\Gamma^*_{\tau,2\kappa,\kappa}$ oracle (i.e., for $d = \kappa$) from $U_{2\kappa+1}(\cdot)$.

Proof of Claim. The answers that $\mathscr{A}$ obtains on a TAG query $\mathbf{m}$ from $\mathscr{D}_1^{U_{2\kappa+1}(\cdot)}$ (i.e. $\pi(\mathbf{R}, \mathbf{z}, \boldsymbol{\omega})$ where $\mathbf{R}, \mathbf{z}, \boldsymbol{\omega}$ are sampled uniformly) are uniformly random, and in particular independent of h or π. The answers to VRFY queries are always reject, and thus contain no information about h, π either. Then, we have that $\mathbf{v}_i = \mathbf{v}_j$ (where $\mathbf{v}_i = C(h(\mathbf{m}_i, \boldsymbol{\omega}_i))$ is defined by the i-th TAG or VRFY query) if and only if $h(\mathbf{m}_i, \boldsymbol{\omega}_i) = h(\mathbf{m}_j, \boldsymbol{\omega}_j)$.

Recall that $\mathscr{A}$ makes a total of Q queries. Assume that up to the $(i-1)$-th query, all the $\mathbf{v}$'s were distinct. If the i-th query is a TAG query, a fresh $\boldsymbol{\omega}_i$ is sampled which will be distinct from all previous $\boldsymbol{\omega}_j$ (for any $j < i$) with probability $1 - (i-1)/2^{\nu}$. Assuming this is the case, the probability that $h(\mathbf{m}_i, \boldsymbol{\omega}_i) = h(\mathbf{m}_j, \boldsymbol{\omega}_j)$ for any $j < i$ can be upper bounded by $i/2^{\mu}$ (here we use the fact that the answers that $\mathscr{A}$ gets from $\mathscr{D}_1^{U_{2\kappa+1}(\cdot)}$ are uniformly random, and thus $\mathscr{A}$ has no information about h).

If the i-th query is a VRFY query $(\mathbf{m}_i, \phi_i)$, then using the fact that π is a pairwise independent permutation (and $\mathscr{A}$ has no information about it) we can show that the probability that ϕ_i contains an $\boldsymbol{\omega}_i$ which is equal to some $\boldsymbol{\omega}_j$ (s.t. $\phi_j \neq \phi_i$) is $\leqslant i/2^{\nu+1}$. If this is the case then $(\mathbf{m}_i, \boldsymbol{\omega}_i) \neq (\mathbf{m}_j, \boldsymbol{\omega}_j)$ for all $j < i$ with overwhelming probability.[9] As in the previous case, we can then upper bound the probability that $h(\mathbf{m}_i, \boldsymbol{\omega}_i) = h(\mathbf{m}_j, \boldsymbol{\omega}_j)$ for any $j < i$ by $i/2^{\mu}$.

Using the union bound over all i, for $1 \leqslant i \leqslant Q$, we get the bound $Q^2/2^{\nu-2} = Q^2/2^{\mu-2}$ (recall that $\mu = \nu$) as claimed. □

THE CASE $\mathbb{P}[\text{FRESH}] > \epsilon/2$. In this case, $\mathscr{A}$ will make TAG, VRFY queries, where with probability $> \epsilon/2$, at some point she will make an accepting VRFY query $(\mathbf{m}, \phi)$ that defines a fresh $\mathbf{v}$. We now construct an adversary $\mathscr{D}_2^{\mathscr{O}}$ that uses $\mathscr{A}$ as a black-box, and can distinguish $\mathscr{O} = \Gamma^*_{\tau,2\kappa,d}(\mathbf{x}, \cdot)$ from $\mathscr{O} = U_{2\kappa+1}(\cdot)$ (as in Definition 4.2) with advantage

$$\frac{\epsilon}{2^{\mu+1}} - Q_{\text{tag}} \cdot \alpha''_{\kappa,d} - Q_{\text{vrfy}} \cdot \alpha'_{\tau',n}. \tag{4.11}$$

The construction of $\mathscr{D}_2^{\mathscr{O}}$ is very similar to the adversary $\mathscr{D}$ that we constructed in the proof of Theorem 4.7 (where we proved that the authentication protocol in Figure 4.5 is secure against active attacks). The queries to the prover in the first phase of an active attack directly correspond to TAG queries. However, we now have to additionally answer VRFY queries (we will always answer reject). Furthermore, we cannot choose the challenge $\mathbf{v}^*$ (as in the 2nd phase of an active

9 Note that for $j < i$ where $\phi_i = \phi_j$ we must have that $\mathbf{m}_i \neq \mathbf{m}_j$ since we assume that $\mathscr{A}$ does not repeat queries and does not ask VRFY queries $(\mathbf{m}, \phi)$ if ϕ was the output of a TAG query $\mathbf{m}$.

attack). Instead, we will simply hope that (in the case where $\mathscr{O} = \Gamma^*_{\tau,2\kappa,d}(\mathbf{x},\cdot)$) the $\mathbf{v}$ contained in the first valid VRFY query (i.e. forgery) that $\mathscr{A}$ makes is fresh (which by assumption happens with probability $\epsilon/2$). Moreover, we will hope that it is the unique $\mathbf{v}^*$ (out of 2^μ possible ones) for which $\mathscr{D}_2^{\mathscr{O}}$ can verify this. This gives us a distinguishing advantage of nearly $\epsilon/2^{\mu+1}$ as stated in Eq. (4.11). We do loose an additional additive term $Q_{\mathsf{tag}} \cdot \alpha''_{\kappa,d}$ as there is an exponentially small probability that the transformation of Subspace LPN samples to TAG queries will fail, and moreover an exponentially small term $Q_{\mathsf{vrfy}} \cdot \alpha'_{\tau',n}$ which accounts for the probability that $\mathscr{A}$ correctly guesses an accepting tag even in the case where $\mathscr{O} = U_{2\kappa+1}(\cdot)$.

Setup. The adversary $\mathscr{D}_2^{\mathscr{O}}$ samples π, h (but not $\mathbf{s}$) as defined by KGEN, and $\mathbf{y}^* \xleftarrow{\$} \mathbb{Z}_2^\mu$, $\mathbf{x}^* \xleftarrow{\$} \mathbb{Z}_2^{2\kappa}$. Let $\mathbf{v}^* = C(\mathbf{y}^*)$. Next, $\mathscr{D}_2^{\mathscr{O}}$ invokes $\mathscr{A}$ and answers its queries as follows (the intuition for the sampling below is given in the proof of Claim 5).

Tag queries. The answer ϕ to a TAG query $\mathbf{m} \in \mathcal{M}$ is computed by $\mathscr{D}_2^{\mathscr{O}}$ as follows:

1. Sample $\boldsymbol{\omega} \xleftarrow{\$} \mathbb{Z}_2^\nu$ and compute $\mathbf{v} = C(h(\mathbf{m}, \boldsymbol{\omega}))$. If $\mathbf{v} = \mathbf{v}^*$, output 0 and stop. Let $\mathbf{u} = \mathbf{v} \wedge \overline{\mathbf{v}}^*$ and $\mathbf{u}^* = \mathbf{v} \wedge \mathbf{v}^*$.
2. Query n times $\mathscr{O}$ on the input $\mathbf{u}$. Let $\widehat{\mathbf{R}}_1 \in \mathbb{Z}_2^{2\kappa \times n}$, $\mathbf{z}_1 \in \mathbb{Z}_2^n$ denote the n outputs of the oracle.
3. Sample $\widehat{\mathbf{R}}_0 \xleftarrow{\$} \mathbb{Z}_2^{2\kappa \times n}$ and set $z_0 = \widehat{\mathbf{R}}_0^\mathsf{T} \cdot (\mathbf{x}^* \wedge \mathbf{u}^*)$.
4. Return $\phi = \pi(\mathbf{R} = \widehat{\mathbf{R}}_{\downarrow \mathbf{v}} \in \mathbb{Z}_2^{\kappa \times n}, \mathbf{z} = \mathbf{z}_0 \oplus \mathbf{z}_1 \in \mathbb{Z}_2^n, \boldsymbol{\omega} \in \mathbb{Z}_2^\nu)$ to $\mathscr{A}$, where $\widehat{\mathbf{R}}$ is uniquely determined by requiring $\widehat{\mathbf{R}}_{\downarrow \mathbf{v}^*} = \widehat{\mathbf{R}}_0$ and $\widehat{\mathbf{R}}_{\downarrow \overline{\mathbf{v}}^*} = \widehat{\mathbf{R}}_1$.

Verification queries. If $\mathscr{A}$ makes a VRFY query $(\phi, \mathbf{m})$, then $\mathscr{D}_2^{\mathscr{O}}$ always answers reject, but also makes the following check:

1. Parse $\mathbf{y} = \pi^{-1}(\phi)$ as $(\mathbf{R} \in \mathbb{Z}_2^{\kappa \times n}, \mathbf{z} \in \mathbb{Z}_2^n, \boldsymbol{\omega} \in \mathbb{Z}_2^\nu)$ and compute $\mathbf{v} = C(h(\mathbf{m}, \boldsymbol{\omega}))$.
2. If $\mathbf{v} \neq \mathbf{v}^*$, processing this query is over, otherwise go to the next step.
3. If $\mathrm{rank}(\mathbf{R}) = n$ and $w_H(\mathbf{R}^\mathsf{T} \cdot \mathbf{x}^*_{\downarrow \mathbf{v}^*} \oplus \mathbf{z}) \leqslant n \cdot \tau'$ (i.e. we have a forgery) output 1 and stop.

If $\mathscr{A}$ has finished its queries, $\mathscr{D}_2^{\mathscr{O}}$ stops with output 0.

Claim 5. *If $\mathscr{O} = \Gamma^*_{\tau,2\kappa,d}(\mathbf{x},\cdot)$, then $\mathscr{D}_2^{\mathscr{O}}$ outputs 1 with probability $\geqslant \epsilon/2^{\mu+1}$.*

Proof. The proof of this claim is similar to the proof of Claim 1. The adversary $\mathscr{D}_2^{\Gamma^*_{\tau,2\kappa,d}(\mathbf{x},\cdot)}$ perfectly simulates access to $\mathsf{TAG}_K(\cdot), \mathsf{VRFY}_K(\cdot,\cdot)$ oracles with key $K = (\mathbf{s}, h, \pi)$ where $\mathbf{s} = (\mathbf{x}^* \wedge \mathbf{v}^*) \oplus (\mathbf{x} \wedge \overline{\mathbf{v}}^*)$ and h, π are sampled by $\mathscr{D}_2^{\mathscr{O}}$. By assumption, in this case, $\mathscr{A}$ outputs a valid fresh forgery with probability $\epsilon/2$. Conditioned on this, with probability $2^{-\mu}$, this fresh $\mathbf{v}$ will be $\mathbf{v}^*$ and therefore $\mathscr{D}_2^{\mathscr{O}}$ will output 1. □

Claim 6. *If $\mathscr{O} = U_{2\kappa+1}(\cdot)$, then $\mathscr{D}_2^{\mathscr{O}}$ outputs 1 with probability $\leqslant Q_{\mathsf{vrfy}} \cdot \alpha'_{\tau',n}$.*

Proof of Claim. The proof of this claim is almost identical to the proof of Claim 2, except that here we have an additional factor Q_{vrfy} as we have to take the union bound over all Q_{vrfy} queries, whereas in Claim 2 the adversary was (by definition of an active attack) only allowed one guess. □

Summing up, using $\mathscr{A}$ we can break the Subset LPN assumption with advantage which is given either by Eq. (4.10) or Eq. (4.11), i.e.

$$\epsilon_{\mathsf{SLPN}} = \min\left\{\frac{\epsilon}{2} - \frac{Q^2}{2^{\mu-2}}, \frac{\epsilon}{2^{\mu+1}} - Q_{\mathsf{tag}} \cdot \alpha''_{\kappa,d} - Q_{\mathsf{vrfy}} \cdot \alpha'_{\tau',n}\right\}.$$

□

We can derive the following corollary, whose proof is straightforward.

Corollary 4.9 (Concrete security of Π^1_M). *Choosing μ s.t. $2^\mu = Q^2 \cdot 2^4/\epsilon$ in Theorem 4.8, we get $\epsilon_{\mathsf{SLPN}} = \min\{\epsilon/4, \epsilon^2/(2^5 Q^2) - 2^{-\Theta(n)}\}$. The 2nd term is the minimum here, and solving for ϵ gives*

$$\epsilon = \sqrt{32} \cdot Q \cdot \sqrt{\epsilon_{\mathsf{SLPN}} + 2^{-\Theta(n)}}. \tag{4.12}$$

Note that to get security as claimed in the above corollary, we need to choose μ as a function of Q and ϵ_{SLPN} such that $2^\mu \approx Q^2 \cdot 2^4/\epsilon$ for ϵ as in Eq. (4.12). Of course we can just fix Q (as an upper bound to the number of queries made by the adversary) and ϵ_{SLPN} (as our guess on the actual hardness of $\Gamma^*_{\tau,2\kappa,d}$). But a too conservative guess on μ (i.e. choosing μ too small) will result in a construction whose security is worse than what is claimed in the above corollary. A too generous guess on the other hand will make the security reduction meaningless (we don't have any actual attacks on the MAC for large μ though).

SECOND CONSTRUCTION. As noted above, the main disadvantage of Π^1_M, is that one needs to fix the hardness of the LPN problem at the time of the construction. Our second construction has no such issues and achieves better security $\Theta(\epsilon \cdot Q)$. The efficiency of this construction is similar to that of the first construction, but a larger key is required.

The main difference to Π^1_M is the way we generate the values $\mathbf{s}(\mathbf{v})$. In the new construction, we define $\mathbf{s}(\mathbf{v}) = \mathbf{s}_0 \oplus \bigoplus_{i:\mathbf{v}[i]=1} \mathbf{s}_i$, where each $\mathbf{s}_i$ is a part of the secret key. The construction uses ideas from Waters' IBE scheme [Wat05], and parts of the security reduction use simulation tricks from [Boy10; ABB10] that we need to adapt to the binary case.

The message authentication code $\Pi^2_M = \{\mathsf{KGEN}, \mathsf{TAG}, \mathsf{VRFY}\}$ with associated message space $\mathcal{M}$ is defined as follows.

- `Public parameters.` Π^2_M has the following public parameters.

κ, τ, τ', n	as in the authentication protocol from Figure 4.5
$\mu \in \mathbb{N}$	output length of the hash function
$\nu \in \mathbb{N}$	length of the randomness

- `Key generation.` Algorithm $\mathsf{KGEN}(1^\kappa)$ samples $\mathbf{s}_i \xleftarrow{\$} \mathbb{Z}_2^\kappa$ (for $0 \leqslant i \leqslant \mu$) and chooses a pairwise independent hash function $h : \mathcal{M} \times \mathbb{Z}_2^\nu \to \mathbb{Z}_2^\mu$, as well as a pairwise independent permutation π over $\mathbb{Z}_2^{\kappa \times n + n + \nu}$. It returns $\mathsf{K} = (\mathbf{s}_0, \ldots, \mathbf{s}_\mu, h, \pi)$ as the secret key.

- `Tagging.` Given secret key $\mathsf{K} = (\mathbf{s}_0, \ldots, \mathbf{s}_\mu, h, \pi)$ and message $\mathbf{m} \in \mathcal{M}$, algorithm TAG proceeds as follows.
  1. $\mathbf{R} \xleftarrow{\$} \mathbb{Z}_2^{\kappa \times n}$, $\boldsymbol{\omega} \xleftarrow{\$} \mathbb{Z}_2^\nu$, $\mathbf{e} \xleftarrow{\$} \mathsf{Bern}_\tau^n$
  2. $\mathbf{v} = h(\mathbf{m}, \boldsymbol{\omega})$
  3. $\mathbf{s}(\mathbf{v}) = \mathbf{s}_0 \oplus \bigoplus_{i:\mathbf{v}[i]=1} \mathbf{s}_i$
  4. Return $\phi = \pi(\mathbf{R}, \mathbf{R}^\mathsf{T} \cdot \mathbf{s}(\mathbf{v}) \oplus \mathbf{e}, \boldsymbol{\omega})$

- `Verification.` On input a secret key $\mathsf{K} = (\mathbf{s}_0, \ldots, \mathbf{s}_\mu, h, \pi)$, message $\mathbf{m} \in \mathcal{M}$ and tag ϕ, algorithm VRFY proceeds as follows.
  1. Parse $\pi^{-1}(\phi)$ as $(\mathbf{R} \in \mathbb{Z}_2^{\kappa \times n}, \mathbf{z} \in \mathbb{Z}_2^n, \boldsymbol{\omega} \in \mathbb{Z}_2^\nu)$. If $\mathrm{rank}(\mathbf{R}) \neq n$, then return reject
  2. $\mathbf{v} = h(m, \boldsymbol{\omega})$
  3. $\mathbf{s}(\mathbf{v}) = \mathbf{s}_0 \oplus \bigoplus_{i:\mathbf{v}[i]=1} \mathbf{s}_i$
  4. If $w_H(\mathbf{z} \oplus \mathbf{R}^\mathsf{T} \cdot \mathbf{s}(\mathbf{v})) > n \cdot \tau'$ return reject, otherwise return accept

Again, it is straightforward to see that the completeness error of Π^2_M is $2^{-c_\tau \cdot n}$ where c_τ only depends on τ.

Theorem 4.10 (Security of Π^2_M). *If the* $\mathsf{SLPN}_{\tau,\kappa,\kappa}$ *problem is* $(t_{\mathsf{SLPN}}, nQ, \epsilon_{\mathsf{SLPN}})$*-hard, then* Π^2_M *is* (t, Q, ϵ)*-secure against uf-cma adversaries, where*

$$t \approx t_{\mathsf{SLPN}} \qquad \epsilon_{\mathsf{SLPN}} = \min\left\{\frac{\epsilon}{2} - \frac{Q^2}{2^{\mu-2}}, \frac{\epsilon}{4Q} - 2^{-\Theta(n)}\right\}.$$

We first give an intuition for the proof of Theorem 4.10. Similar to the proof of Theorem 4.8, we distinguish fresh and non-fresh forgeries. Here the new and interesting case is the fresh forgery. The idea is that in the reduction to the SLPN problem we define the function $\mathbf{s}(\mathbf{v}) = \mathbf{A}(\mathbf{v}) \cdot \mathbf{s} \oplus \mathbf{b}(\mathbf{v})$ (where $\mathbf{s} = \mathbf{x}$ is the SLPN secret) such that the following holds with non-negligible probability: (i) For each $\mathbf{v}_i$ from the TAG queries, $\mathbf{A}(\mathbf{v}_i)$ has full rank κ and hence the tags can be simulated using the provided $\Gamma_{\tau,\kappa,\kappa}(\mathbf{s}, \cdot, \cdot)$ oracle; (ii) For the first fresh forgery we have $\mathbf{A}(\mathbf{v}) = 0$ such that $\mathbf{s}(\mathbf{v})$ is independent of $\mathbf{s}$ and the reduction can check the forgery's correctness. The above two properties allow to maintain the simulation.

The setup of the function $\mathbf{s}(\cdot)$ is the crucial step and here we adapt a technique recently introduced by Boyen [Boy10] based on homomorphic encodings with full-rank differences that allows us to arbitrarily control the probability that the above simulation works.

Proof. Let $\mathscr{A}$ be an adversary that successfully forges in the uf-cma experiment with probability ϵ. We make the same conventions and the definition of freshness as in the proof of Theorem 4.8 and split the forging probability as $\mathbb{P}[\textsc{Fresh}] + \mathbb{P}[\neg\textsc{Fresh}] = \epsilon$.

THE CASE $\mathbb{P}[\textsc{Fresh}] \leqslant \epsilon/2$. We now give the description of $\mathscr{D}_1^{\mathscr{O}}$ attacking the standard LPN problem, i.e. $\mathscr{D}_1^{\mathscr{O}}$ can distinguish $\mathscr{O} = \Lambda_{\tau,\kappa}(\mathbf{s})$ from $\mathscr{O} = \mathsf{U}_{\kappa+1}$ with advantage

$$\frac{\epsilon}{2} - \frac{Q^2}{2^{\mu-2}}. \tag{4.13}$$

Setup. Adversary $\mathscr{D}_1^{\mathscr{O}}$ samples π, h (but not $\mathbf{s}$) as defined by KGEN and $\mathbf{b}_i \xleftarrow{\$} \mathbb{Z}_2^\kappa$, for $0 \leqslant i \leqslant \mu$. Next, it implicitly defines $\mathbf{s}_0 = \mathbf{s} \oplus \mathbf{b}_0$ (where $\mathbf{s}$ is unknown) and $\mathbf{s}_i = \mathbf{b}_i$. It is easy to see that with this setup of $K = (\mathbf{s}_0, \ldots, \mathbf{s}_\mu, h, \pi)$ we have that for each $\mathbf{v} \in \mathbb{Z}_2^\mu$,

$$\mathbf{s}(\mathbf{v}) = \mathbf{s} \oplus \mathbf{b}(\mathbf{v}), \text{ where } \mathbf{b}(\mathbf{v}) = \mathbf{b}_0 \oplus \bigoplus_{i:\mathbf{v}[i]=1} \mathbf{b}_i. \tag{4.14}$$

Note that adversary $\mathscr{D}_1^{\mathscr{O}}$ cannot evaluate $\mathbf{s}(\mathbf{v})$ but looking ahead, it will use its oracle $\mathscr{O}$ to answer $\mathscr{A}$'s queries as follows.

Tag queries. If $\mathscr{A}$ makes a $\mathsf{TAG}_K(\cdot)$ query for message $\mathbf{m} \in \mathcal{M}$, then $\mathscr{D}_1^{\mathscr{O}}$ does the following:

1. Samples $\boldsymbol{\omega} \xleftarrow{\$} \mathbb{Z}_2^{\nu}$ and compute $\mathbf{v} = h(\mathbf{m}, \boldsymbol{\omega})$. Compute $\mathbf{b}(\mathbf{v})$ as in Eq. (4.14).
2. Query the oracle $\mathscr{O}$ for n times. Let $\mathbf{R} \in \mathbb{Z}_2^{\kappa \times n}$, $\mathbf{z}' \in \mathbb{Z}_2^n$ denote the n answers from the oracle.
3. Return $\phi = \pi(\mathbf{R}, \mathbf{z}, \boldsymbol{\omega})$, where $\mathbf{z} = \mathbf{z}' \oplus \mathbf{R}^{\mathsf{T}} \cdot \mathbf{b}(\mathbf{v})$.

Verification queries. If $\mathscr{A}$ makes a $\mathsf{VRFY}_K(\cdot,\cdot)$ query $(\mathbf{m}, \phi)$, then $\mathscr{D}_1^{\mathscr{O}}$ simply answers with reject.

Finally, if any TAG or VRFY query contains a $\mathbf{v}$ which has appeared in a previous query, $\mathscr{D}_1^{\mathscr{O}}$ outputs 1 and stops. Otherwise, it outputs 0. Note that if $\mathscr{O} = \Lambda_{\tau,\kappa}(\mathbf{s})$, then $\mathscr{D}_1^{\mathscr{O}}$ perfectly simulates the $\mathsf{TAG}_K(\cdot)$ oracle.

The following two claims are the analogues of Claims 3 and 4, respectively. Their proofs are essentially the same and are therefore omitted.

Claim 7. *If $\mathscr{O} = \Lambda_{\tau,\kappa}(\mathbf{s})$, then $\mathscr{D}_1^{\mathscr{O}}$ outputs 1 with probability $\geqslant \epsilon/2$.*

Claim 8. *If $\mathscr{O} = \mathsf{U}_{\kappa+1}$, then $\mathscr{D}_1^{\mathscr{O}}$ outputs 1 with probability $< Q^2/2^{\mu-2}$.*

Before we start dealing with the case $\mathbb{P}[\textsc{Fresh}] > \epsilon/2$, we recall the concept of encodings with full-rank differences over general finite fields $\mathbb{F}$. (For our proof we will only be interested in the case $\mathbb{F} = \mathbb{Z}_2$.)

Definition 4.3 (Encoding with full-rank differences). Let $l \geqslant 1$ be an integer and $\mathbb{F}$ be a finite field. A mapping $\varphi : \mathbb{F}^l \to \mathbb{F}^{l \times l}$ is an encoding with full-rank differences (FRD) over $\mathbb{F}$, if

- φ is computable in polynomial time (in l);
- For all distinct vectors $\mathbf{a}, \mathbf{b} \in \mathbb{F}^l$, we have that $\varphi(\mathbf{a}) - \varphi(\mathbf{b})$ is an invertible matrix.

Further, φ is *homomorphic* if for all vectors $\mathbf{a}, \mathbf{b} \in \mathbb{F}^l$, we have that $\varphi(\mathbf{a}) + \varphi(\mathbf{b}) = \varphi(\mathbf{a} + \mathbf{b})$.

Cramer and Damgard [CD09], provided the following construction of an encoding with FRD. Let f be some polynomial of degree l in $\mathbb{F}[X]$ that is irreducible. Recall that if h is a polynomial over $\mathbb{F}[X]$, the polynomial $h \bmod f$ has degree less than l and therefore $\mathrm{coefs}(h \bmod f)$ can be viewed as a vector in $\mathbb{F}^l$. For an input $\mathbf{a} = (\mathbf{a}[0], \ldots, \mathbf{a}[l-1]) \in \mathbb{F}^l$ define the polynomial $g_{\mathbf{a}}(X) = \sum_{i=0}^{l-1} \mathbf{a}[i]X^i \in \mathbb{F}[X]$. Let,

$$\varphi(\mathbf{a}) = \begin{pmatrix} \mathrm{coefs}(g_{\mathbf{a}}) \\ \mathrm{coefs}(X \cdot g_{\mathbf{a}} \bmod f) \\ \mathrm{coefs}(X^2 \cdot g_{\mathbf{a}} \bmod f) \\ \vdots \\ \mathrm{coefs}(X^{l-1} \cdot g_{\mathbf{a}} \bmod f) \end{pmatrix} \in \mathbb{F}^{l \times l}$$

As proved in [CD09], this is an encoding with FRD. Furthermore, it is also homomorphic since $\mathrm{coefs}(X^i \cdot g_{\mathbf{a}+\mathbf{b}} \bmod f) = \mathrm{coefs}(X^i \cdot g_{\mathbf{a}} \bmod f) + \mathrm{coefs}(X^i \cdot g_{\mathbf{b}} \bmod f)$. Thus, in particular, we have that for any prime $q \geqslant 2$ and integer $l \geqslant 1$, there exists an homomorphic encoding with full-rank differences over $\mathbb{Z}_q$.

THE CASE $\mathbb{P}[\text{FRESH}] > \epsilon/2$. We will use games, denoting by G_i the output of the experiment in the i-th game.

Game 0: The uf-cma security experiment with the modification that it only returns 1 in case FRESH happens. By assumption, we have that $\mathbb{P}[G_0 = 1] \geqslant \epsilon/2$.

Game 1: Let $l = \lceil \log_2(2Q) \rceil$. The experiment is the same as in Game 0 with the following differences.

- Key Generation. Algorithm KGEN additionally picks $\mathbf{a}_i \xleftarrow{\$} \mathbb{Z}_2^l$ ($0 \leqslant i \leqslant \mu$) and defines

$$\mathbf{a}(\mathbf{v}) = \mathbf{a}_0 \oplus \bigoplus_{i:\mathbf{v}[i]=1} \mathbf{a}_i \in \mathbb{Z}_2^l. \tag{4.15}$$

- Tagging. For a query $\mathbf{m} \in \mathcal{M}$, algorithm TAG proceeds as follows. As in the last game, it computes tag ϕ and all the intermediate values. If $\mathbf{a}(\mathbf{v}) = 0^l \in \mathbb{Z}_2^l$ then the experiment stops and outputs 0. Otherwise, it outputs tag ϕ.
- Verification. For a query $(\mathbf{m}^*, \phi^*)$, algorithm VRFY proceeds as follows. As in the last game, it first computes the output of the real VRFY algorithm, together with all intermediate values. If the output

is out = accept and $\mathbf{a}(\mathbf{v}^*) \neq 0^l \in \mathbb{Z}_2^l$, then it stops and outputs 0. Otherwise, it returns out.

Claim 9. $\mathbb{P}[G_0 = 1] = 2Q \cdot \mathbb{P}[G_1 = 1]$.

Proof. Let FAIL be the event that the execution in Game 1 stops, but not in Game 0, i.e.,

$$\text{FAIL} = \mathbf{a}(\mathbf{v}^*) = 0^l \wedge \mathbf{a}(\mathbf{v}_1) \neq 0^l \wedge \ldots \wedge \mathbf{a}(\mathbf{v}_Q) \neq 0^l,$$

where $\mathbf{v}_i$ is the $\mathbf{v}$ value appearing in the i-th TAG query and $\mathbf{v}^*$ is the $\mathbf{v}$ value of the first fresh VRFY query. We use a variant of [Boy10, Lemma 27] to show that

$$\frac{1}{2^l}\left(1 - \frac{Q}{2^l}\right) \leqslant \mathbb{P}_{\mathbf{a}}[\text{FAIL}] \leqslant \frac{1}{2^l}, \tag{4.16}$$

where the probability is taken over $\mathbf{a} = (\mathbf{a}_0, \ldots, \mathbf{a}_\mu) \xleftarrow{\$} (\mathbb{Z}_2^l)^{\mu+1}$. Note that $\mathbf{a}(\cdot)$ from Eq. (4.15) is essentially pairwise independent over $\mathbb{Z}_2^l$, i.e. for each $\mathbf{v}_i \neq \mathbf{v}^* \in \mathbb{Z}_2^\mu$, we have $\Pr[\mathbf{a}(\mathbf{v}^*) = 0^l] = 1/2^l$ and $\Pr[\mathbf{a}(\mathbf{v}_i) = 0^l | \mathbf{a}(\mathbf{v}^*) = 0^l] = 1/2^l$. Then Eq. (4.16) follows by applying the union bound. The claim now follows by the definition of $l = \lceil \log_2(2Q) \rceil$. □

Game 2: Oracle $\text{TAG}_K(\cdot)$ now internally uses uniform $(\mathbf{R}, \mathbf{z}) \in \mathbb{Z}_2^{\kappa \times n} \times \mathbb{Z}_2^n$ to generate tag ϕ on message $\mathbf{m}$.

Claim 10. $\mathbb{P}[G_1 = 1] - \mathbb{P}[G_2 = 1] \leqslant \epsilon_{\text{SLPN}}$.

Proof. Assume there exists an adversary $\mathscr{A}$ that can distinguish between the two games. We now describe adversary $\mathscr{D}_2^{\mathscr{O}(\cdot,\cdot)}$ that $(t_{\text{SLPN}}, nQ, \epsilon_{\text{SLPN}})$-breaks the $\text{SLPN}_{\tau,\kappa,\kappa}$ problem (cf. Definition 4.1).

- **Setup.** Let $l = \lceil \log_2(2Q) \rceil$ and $\varphi : \mathbb{Z}_2^l \to \mathbb{Z}_2^{l \times l}$ be an homomorphic encoding with full-rank differences over $\mathbb{Z}_2$ from Definition 4.3. The adversary $\mathscr{D}_2^{\mathscr{O}}$ picks $\mathbf{a}_i \xleftarrow{\$} \mathbb{Z}_2^l$, $\mathbf{b}_i \xleftarrow{\$} \mathbb{Z}_2^\kappa$ (for $i = 0, \ldots, \mu$) and (implicitly) defines $\mathbf{s}_i = \mathbf{A}_i \cdot \mathbf{s} \oplus \mathbf{b}_i$, where $\mathbf{s} = \mathbf{x}$ is the (unknown) secret from the oracle $\Gamma_{\tau,\kappa,\kappa}(\mathbf{s}, \cdot, \cdot)$ and

$$\mathbf{A}_i = \begin{pmatrix} \underbrace{\boxed{\varphi(\mathbf{a}_i)}}_{\in \mathbb{Z}_2^{l \times l}} & 0 & \cdots & 0 \\ 0 & \boxed{\varphi(\mathbf{a}_i)} & & \vdots \\ \vdots & & \ddots & \vdots \\ 0 & \cdots & & \boxed{\varphi(\mathbf{a}_i)} \end{pmatrix} \in \mathbb{Z}_2^{\kappa \times \kappa}.$$

(Here we assume that $l \mid \kappa$. If that is not the case, the matrix $\mathbf{A}_i$ is truncated.) This way we have that $\mathbf{s}(\mathbf{v}) = \mathbf{A}(\mathbf{v}) \cdot \mathbf{s} \oplus \mathbf{b}(\mathbf{v})$, where $\mathbf{A}(\mathbf{v}) = \mathbf{A}_0 \oplus \bigoplus_{i:\mathbf{v}[i]=1} \mathbf{A}_i$ and $\mathbf{b}(\mathbf{v}) = \mathbf{b}_0 \oplus \bigoplus_{i:\mathbf{v}[i]=1} \mathbf{b}_i$. Define $\mathbf{a}(\mathbf{v}) = \mathbf{a}_0 \oplus \bigoplus_{i:\mathbf{v}[i]=1} \mathbf{a}_i$ as in Eq. (4.15). By the properties of the homomorphic encoding with FRD we have that

$$\mathbf{A}(\mathbf{v}) = \begin{cases} 0^{\kappa \times \kappa} & \text{if } \mathbf{a}(\mathbf{v}) = 0^l \\ \text{full-rank matrix} & \text{if } \mathbf{a}(\mathbf{v}) \neq 0^l \end{cases} \tag{4.17}$$

- **Tag queries.** Next, $\mathscr{D}_2^{\mathscr{O}}$ invokes $\mathscr{A}$. The answer $(\mathbf{R}, \mathbf{z})$ to a $\mathsf{TAG}_K(\cdot)$ query $\mathbf{m} \in \mathcal{M}$ by $\mathscr{A}$ is computed by $\mathscr{D}_2^{\mathscr{O}}$ as follows.
  1. Sample $\boldsymbol{\omega} \xleftarrow{\$} \mathbb{Z}_2^{\nu}$ and compute $\mathbf{v} = h(\mathbf{m}, \boldsymbol{\omega})$.
  2. Compute $\mathbf{a}(\mathbf{v})$, $\mathbf{b}(\mathbf{v})$ and $\mathbf{A}$. If $\mathbf{a}(\mathbf{v}) = 0^l$, then stop and output 0. If $\mathbf{a}(\mathbf{v}) \neq 0^l$ we have that $\mathbf{A}(\mathbf{v})$ is a full-rank matrix by Eq. (4.17).
  3. Query $\mathscr{O}(\cdot, \cdot)$ for n times on input $(\mathbf{A}(\mathbf{v}), \mathbf{b}(\mathbf{v}))$. Let $\mathbf{R} \in \mathbb{Z}_2^{\kappa \times n}$, $\mathbf{z} \in \mathbb{Z}_2^n$ denote the n answers from the oracle.
  4. Return $\phi = \pi(\mathbf{R}, \mathbf{z}, \boldsymbol{\omega})$ to $\mathscr{A}$.
- **Verification queries.** If $\mathscr{A}$ makes a verification query consisting of $\mathbf{m}^*$ and a tag $\phi^* = \pi(\mathbf{R}^*, \mathbf{z}^*, \boldsymbol{\omega}^*)$, then $\mathscr{D}_2^{\mathscr{O}}$ computes $\mathbf{v}^* = h(\mathbf{m}^*, \boldsymbol{\omega}^*)$. If $\mathbf{a}(\mathbf{v}^*) \neq 0^l$ then $\mathscr{D}_2^{\mathscr{O}}$ returns reject. Otherwise, we have that $\mathbf{s}(\mathbf{v}^*) = \mathbf{b}(\mathbf{v}^*)$ does not depend on the unknown secret $\mathbf{s}$ anymore and $\mathscr{D}_2^{\mathscr{O}}$ can perform the real check which is

$$\operatorname{rank}(\mathbf{R}^*) = n \quad \text{and} \quad w_H(\mathbf{z}^* \oplus \mathbf{R}^{*\mathsf{T}} \cdot \mathbf{b}(\mathbf{v}^*)) \leqslant n \cdot \tau'. \tag{4.18}$$

  The output is 1 if both checks succeed and 0 otherwise.

We now analyse $\mathscr{D}_2^{\mathscr{O}}$. If $\mathscr{O}(\cdot, \cdot) = \Gamma_{\tau,\kappa,\kappa}(\mathbf{s}, \cdot, \cdot)$ is the real Subspace LPN oracle, then the answers that $\mathscr{D}_2^{\mathscr{O}}$ gives to $\mathscr{A}$'s queries in the first phase of the attack have *exactly* the same distribution as what $\mathscr{A}$ would get in Game 1. (That is, the $\mathbf{z}$ from the oracle is of the form $\mathbf{z} = \mathbf{R}^{\mathsf{T}} \cdot (\mathbf{A}(\mathbf{v}) \cdot \mathbf{s} \oplus \mathbf{b}(\mathbf{v})) \oplus \mathbf{e} = \mathbf{R}^{\mathsf{T}} \cdot \mathbf{s}(\mathbf{v}) \oplus \mathbf{e}$, as in Game 1.) Hence $\mathbb{P}[\mathscr{D}_2^{\Gamma_{\tau,\kappa,\kappa}(\mathbf{s},\cdot,\cdot)} = 1] = \mathbb{P}[G_1 = 1]$.

If $\mathscr{O} = \mathsf{U}_{\kappa+1}(\cdot, \cdot)$ is the uniform oracle, then all the outputs made by the TAG oracle are uniformly random. Hence $\mathbb{P}[\mathscr{D}_2^{\mathsf{U}_{\kappa+1}(\cdot,\cdot)} = 1] = \mathbb{P}[G_2 = 1]$. □

We next bound the probability that the experiment in Game 2 outputs 1.

Claim 11. $\mathbb{P}[G_2 = 1] \leqslant \alpha'_{\tau',n} = 2^{-\Theta(n)}$.

Proof of Claim. If $\mathbf{R}^*$ does not have full rank then the experiment outputs 0 by definition. So, from now on, we only consider the case where $\text{rank}(\mathbf{R}^*) = n$. In Game 2, the answers $(\mathbf{R}, \mathbf{z})$ adversary $\mathscr{A}$ obtains from the TAG oracle are independent of the secrets $\mathbf{s}_0, \ldots, \mathbf{s}_\mu$. Since $\mathbf{s}(\mathbf{v}^*)$ is uniformly random and $\mathbf{R}^*$ has full rank, the vector $\mathbf{R}^{*\mathsf{T}} \cdot \mathbf{s}(\mathbf{v}^*) \oplus \mathbf{z}^*$ is uniformly random over $\mathbb{Z}_2^n$. Thus the probability that the second verification $w_H(\mathbf{z}^* \oplus \mathbf{R}^{*\mathsf{T}} \cdot \mathbf{s}(\mathbf{v}^*)) \leqslant n \cdot \tau'$ fails is $\mathbb{P}[w_H(\mathbf{x}) \leqslant n \cdot \tau'] = \alpha'_{\tau',n} = 2^{-\Theta(n)}$. □

To sum up, in the case $\mathbb{P}[\text{FRESH}] > \epsilon/2$ (putting all together the terms in Claim 9 - 11), we can use $\mathscr{A}$ to break the Subspace LPN assumption with advantage $\epsilon/(4Q) - 2^{-\Theta(n)}$. On the other hand in the case $\mathbb{P}[\text{FRESH}] \leqslant \epsilon/2$, we have an advantage as given in Eq. (4.13). Thus

$$\epsilon_{\text{SLPN}} = \min\left\{\frac{\epsilon}{2} - \frac{Q^2}{2^{\mu-2}}, \frac{\epsilon}{4Q} - 2^{-\Theta(n)}\right\},$$

as desired. □

4.5 EXTENSIONS AND OPEN PROBLEMS

We have provided new constructions of authentication protocols and even MACs from LPN. Unlike previous proposals, our constructions are not ad-hoc, and we gave a reduction to the LPN problem. We conclude this chapter with some extensions and open problems.

TRADING KEY-SIZE FOR COMMUNICATION COMPLEXITY. A disadvantage of the schemes proposed in this chapter is their large communication complexity. For example, in the authentication protocol from Section 4.3 the prover has to send the entire $\kappa \times n$ matrix $\mathbf{R}$ to the verifier. Similarly, in the MACs from Section 4.4, the tag is computed by permuting a string of the form $(\mathbf{R}, \mathbf{R}^{\mathsf{T}} \cdot f_{\mathbf{s}}(\mathbf{m}) \oplus \mathbf{e}, \boldsymbol{\omega})$, where again $\mathbf{R}$ is an $\kappa \times n$ matrix.

We now explain a simple efficiency trade-off that is originally due to Gilbert *et al.* [GRS08b]. Consider the authentication protocol from Figure 4.5. Let $1 \leqslant c \leqslant n$ be an integer parameter and let $n_s = c$ and $n_r = n/c$. The idea is to use a larger secret matrix $\mathbf{S} \in \mathbb{Z}_2^{2\kappa \times n_s}$ (instead of just one vector $\mathbf{s}$) and a smaller random matrix $\mathbf{R} \in \mathbb{Z}_2^{\kappa \times n_r}$ (instead of $\mathbf{R} \in \mathbb{Z}_2^{\kappa \times n}$). The resulting

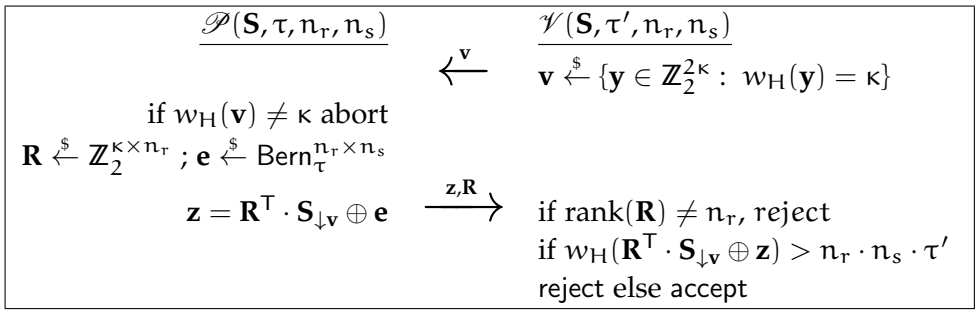

Figure 4.7: A generalisation of the protocol from Figure 4.5 where we trade a larger key (which now is a matrix $\mathbf{S} \in \mathbb{Z}_2^{2\kappa \times n_s}$) for lower communication and randomness complexity. The protocol is as secure as the protocol from Figure 4.5 (wich is the special case where $n_r = n$ and $n_s = 1$) with $n = n_r \cdot n_s$.

protocol is illustrated in Figure 4.7. Similar extensions can be easily derived for the MACs of Section 4.4, where the tradeoff is more important due to the pairwise independent permutation π which is the computational bottleneck of the protocol. See Figure 4.1 for a comparison of the resulting complexities. The proof of Theorem 4.7, Theorem 4.8 and Theorem 4.10, can be adapted to show the same security and completeness results.

GENERALIZATION TO LWE. All the protocols discussed in this chapter are based on the hardness of the LPN problem. A natural generalization of this problem is the learning with errors (LWE) problem [Reg05]. The most appealing characteristic of this problem is that it enjoys for certain parameters a worst-case hardness guarantee [Reg05; Pei09].

The Subspace/Subset version of the LWE problem can be defined exactly in the same fashion as for LPN. It was showed in [Pie10] that the Subspace/Subset LWE problems are equivalent to the LWE problem. All the protocols in this paper can be generalized to $\mathbb{Z}_q$ and proven secure under the hardness of the Subset LWE assumption (and hence the standard LWE assumption). This requires us to sample all the elements from $\mathbb{Z}_q$ (instead of $\mathbb{Z}_2$), replace Bern_τ with $\mathsf{Gau}_{q,\tau}$ and perform all the operations involved modulo q.

We need also to specify how to replace the verification steps involving the computation of Hamming weights $w_H(\cdot)$. Given a vector $\mathbf{e} \in \mathbb{Z}_q^n$ sampled from $\mathsf{Gau}_{q,\tau}^n$ (where $\mathbf{e}$ has the form $\mathbf{z} - \mathbf{R}^\intercal \cdot \mathbf{s}_{\downarrow \mathbf{v}} \bmod q$ for a honest execution of the protocol from Section 4.3 or $\mathbf{z} - \mathbf{R}^\intercal \cdot \mathbf{s}(\mathbf{v}) \bmod q$ for the schemes from Section 4.4),

Construction	Security	Complexity		Key-size	Reduction
		Communication	Computation		
HB [HB01]	passive (2 rnd)	$\kappa \cdot n/c$	$\Theta(\kappa \cdot n)$	$\kappa \cdot c$	ϵ (tight)
HB^+ [JW05]	active (3 rnd)	$\kappa \cdot n \cdot 2/c$	$\Theta(\kappa \cdot n)$	$\kappa \cdot 2 \cdot c$	$\sqrt{\epsilon}$
Π § 4.3	active (2 rnd)	$\kappa \cdot n \cdot 2.1/c$	$\Theta(\kappa \cdot n)$	$\kappa \cdot 4.2 \cdot c$	ϵ (tight)
Π^1_M § 4.4	MAC $\to$ MIM (2 rnd)	$\kappa \cdot n \cdot 2.1/c$	$\Theta(\kappa \cdot n) + \mathsf{PIP}$	$\kappa \cdot 12.6 \cdot c$	$\sqrt{\epsilon} \cdot Q$
Π^2_M § 4.4	MAC $\to$ MIM (2 rnd)	$\kappa \cdot n \cdot 1.1/c$	$\Theta(\kappa \cdot n) + \mathsf{PIP}$	$\kappa \cdot \mu \cdot c$	$\epsilon \cdot Q$
GGM [GGM86]	PRF $\to$ MIM (2 rnd)	μ	$\Theta(\kappa^2 \cdot \mu)$	$\Theta(\kappa)$	$\epsilon \cdot \mu$

Table 4.1: A comparison of our new authentication protocol and MACs with the HB, HB^+ protocols and the classical GGM construction. The trade-off parameter $c, 1 \leqslant c \leqslant n$ and the term PIP are explained below.

this can be done by checking that the (squared) Euclidean norm of $\mathbf{e}$, i.e., the quantity $\|\mathbf{e}\|^2 = \sum_{i=1}^{n} |\mathbf{e}[i]|^2$, does not exceed $n \left\lfloor \frac{q}{2} \right\rfloor \cdot \tau'$ (which will happen with overwhelming probability by the standard tail bound on Gaussians).

Thus the change of domain from $\mathbb{Z}_2$ to $\mathbb{Z}_q$ buys us security based on a different assumption, which is known to be equivalent (for a proper choice of parameters) to the hardness of well-studied (worst-case) lattice problems. This comes at the price of a higher computational complexity, which may be a problem in the context of resource bounded devices.

EFFICIENCY ISSUES. Our authentication protocol is roughly as efficient as the HB^+ protocol but has twice the key length. Our MACs perform roughly the same computation as the authentication protocol plus one evaluation of a pairwise independent permutation of an $\approx 2\kappa$ bit domain, where κ is the length of an LPN secret.

Figure 4.1 gives a rough comparison of our new protocol and MACs with the HB, HB^+ protocols and, as a reference, also the classical tree-based GGM construction [GGM86]. The second row in the table specifies the security notion that is (provably) achieved under the $\mathsf{LPN}_{\tau,\kappa}$ assumption. The value μ is also a security parameter and n denotes the number of "repetitions". Typical parameters can be $\kappa = 500, \mu = 80, n = 250$. Computation complexity counts the number of binary operations over $\mathbb{Z}_2$. Communication complexity counts the total length of all exchanged messages.[10] The last row in the table states the tightness of the security reduction, i.e. what exact security is achieved (ignoring constants and higher order terms) assuming the $\mathsf{LPN}_{\tau,\kappa}$ problem is ϵ-hard.

10 For MACs, we consider the communication one incurs by constructing a MIM secure 2-round protocol from the MAC by having the prover compute the tag on a random challenge message.

The prover and verifier in the HB, HB^+ and our new protocols have to perform $\Theta(\kappa \cdot n)$ basic binary operations, assuming the $\mathsf{LPN}_{\tau,\kappa}$ problem (i.e., LPN with secrets of length κ) is hard. This seems optimal, as $\Theta(\kappa)$ operations are necessary to compute the inner product which generates a single pseudorandom bit. We will thus consider an authentication protocol or MAC *efficient*, if it requires $O(\kappa \cdot n)$ binary operations. It is well known that one gets a length-doubling PRG under the $\mathsf{LPN}_{\tau,\kappa}$ assumption with $\Theta(\kappa^2)$ binary operations [FS96]. Via the classical GGM construction [GGM86], we obtain a PRF and hence a MAC. This PRF, however, requires $\Theta(\kappa^2 \cdot \mu)$ operations per invocation (where μ is the size of the domain of the PRF) which is not very practical. (Recall that $\kappa \approx 500$.)

As we have discussed above, for all constructions except GGM, there is a natural trade-off between communication and key-size, where for any constant c ($1 \leqslant c \leqslant n$), we can decrease communication by a factor of c and increase key-size by the factor c. For the first three protocols in the table, the choice of c does not affect the computational efficiency, but it does so for our MACs: to compute or verify a tag one has to evaluate a pairwise independent permutation (PIP) on the entire tag of length $\ell = \Theta(\kappa \cdot n/c)$.

The standard way to construct a PIP π over $\mathbb{Z}_{2^\ell}$ is to define $\pi(x) = a \cdot x + b \in \mathbb{Z}_{2^\ell}$ for random $a, b \in \mathbb{Z}_{2^\ell}$. Thus the computational cost of evaluating the PIP is one multiplication of two ℓ bit values: the PIP term in the table accounts for this complexity. Asymptotically, such a multiplication takes only $O(\ell \log \ell \log\log \ell)$ time [SS71; Für09], but for small ℓ (like in our scheme) this will not be faster than using schoolbook multiplication, which takes $\Theta(\ell^2)$ time. For parameters $\kappa = 500, n = 250$ and trade-off $c = n$ (which minimizes the tag-length ℓ) we get $\ell \approx 1200$ for Π^1_M (i.e., $1200 = 2\kappa$ plus some statistical security parameters) and $\ell \approx 600$ for Π^2_M. Hence, depending on the parameters, the evaluation of the PIP may be the computational bottleneck of our MACs. It is an interesting open problem how to get rid of the PIP π in our construction, thus getting a more efficient construction.

Very recently, building on our ideas, a new variant of the HB protocol—called HB^n—is introduced in [BHN11], exploiting a bilinear variant of the LPN assumption (also this variant can be proven equivalent to the standard LPN assumption). This construction directly achieves MiM-security, however it has similar efficiency as our MACs.

A PROOF OF THE CHERNOFF BOUND

In this appendix we recall the (standard) proof of the Chernoff bound.

Proof of Theorem 2.3. Define $X = \sum_{i=1}^{n} X_i$. If we denote with $p_i = \mathbb{P}[X_i = 1]$, we have $\mu = \sum_{i=1}^{n} p_i$. We prove only the first inequality, the other one can be proved similarly. We introduce a parameter t, which role will be clarified in a moment. We have

$$\mathbb{P}\left[X > (1+\delta)\mu\right] = \mathbb{P}\left[e^{tX} > e^{t(1+\delta)\mu}\right].$$

Markov's inequality (cf. Lemma 2.2) yields

$$\mathbb{P}\left[X > (1+\delta)\mu\right] \leqslant \frac{\mathbb{E}\left[e^{tX}\right]}{e^{t(1+\delta)\mu}}. \tag{A.1}$$

Note that since the random variables X_i are mutually independent, we can write

$$\mathbb{E}\left[e^{tX}\right] = \mathbb{E}\left[e^{t\sum_{i=1}^{n} X_i}\right] = \mathbb{E}\left[\prod_{i=1}^{n} e^{tX_i}\right] = \prod_{i=1}^{n} \mathbb{E}\left[e^{tX_i}\right]. \tag{A.2}$$

Moreover the term e^{tX_i} can be upper bounded exploiting the fact that $e^x > 1 + x$ and that $X_i \in \{0, 1\}$:

$$\mathbb{E}\left[e^{tX_i}\right] = p_i e^t + (1 - p_i) = 1 + (e^t - 1)p_i \leqslant e^{(e^t-1)p_i}.$$

Substituting in Eq. (A.2), we get

$$\mathbb{E}\left[e^{tX}\right] < \prod_{i=1}^{n} e^{(e^t-1)p_i} = e^{\sum_{i=1}^{n}(e^t-1)p_i} = e^{(e^t-1)\mu},$$

and putting this expression in Eq. (A.1) yields

$$\mathbb{P}\left[X > (1+\delta)\mu\right] \leqslant \frac{\mathbb{E}\left[e^{tX}\right]}{e^{t(1+\delta)\mu}} < \frac{e^{(e^t-1)\mu}}{e^{t(1+\delta)\mu}} = \left(e^{e^t-1-t(1+\delta)}\right)^{\mu}.$$

The last equation is valid for every t. To prove ($\star$), it suffices to find the optimal value of t which makes the above inequality tight. In other words, we

need to minimise $e^t - 1 - t(1+\delta)$ as a function of t. Forcing the first derivative to be 0, we get

$$\frac{d}{dt}(e^t - 1 - t(1+\delta)) = e^t - 1 - \delta \stackrel{!}{=} 0,$$

and we obtain the unique solution $t = \ln(1+\delta)$. Thus

$$\mathbb{P}[X > (1+\delta)\mu] < \left(e^{\delta-(1+\delta)\ln(1+\delta)}\right)^{\mu} = \left(\frac{e^{\delta}}{(1+\delta)^{1+\delta}}\right)^{\mu},$$

as desired.

To prove $(\star\star)$ we write the Mclaurin series[1] of $\ln(1+\delta)$, i.e.

$$(1+\delta)\ln(1+\delta) = (1+\delta)\sum_{i\geqslant 1}(-1)^{i+1}\frac{\delta^i}{i} = \delta + \sum_{i\geqslant 2}(-1)^i\delta^i\left(\frac{1}{i-1} - \frac{1}{i}\right).$$

Now, if $0 \leqslant \delta < 1$, we can ignore the higher order terms and write

$$(1+\delta)\ln(1+\delta) > \delta + \frac{\delta^2}{2} - \frac{\delta^3}{6} \geqslant \delta - \frac{\delta^2}{3}.$$

Hence,

$$\mathbb{P}[X > (1+\delta)\mu] < \left(e^{\delta-(1+\delta)\ln(1+\delta)}\right)^{\mu} \leqslant e^{-\frac{\delta^2\mu}{3}} \qquad (0 \leqslant \delta < 1),$$

as desired.

1 In mathematics, a result due to Taylor allows to approximate a function around a certain point in the space, trough an expression involving some special polynomials with coefficients depending only on the value of the derivatives of f in the given point. More precisely, let $f : [a,b] \to \mathbb{R}^n$ be a function which is differentiable n times in $[a,b]$ and let $x_0 \in [a,b]$. Then,

$$f(x) = \sum_{i=0}^{\infty}\frac{f^{(i)}(x_0)}{i!}(x-x_0)^i,$$

where we write $f^{(i)}(x_0)$ for the i-th derivative of f at point x_0. When $x_0 = 0$ we speak of the Mclaurin series. A simple calculation shows

$$\ln(1+x) = \sum_{i=1}^{\infty}(-1)^{i+1}\frac{x^i}{i} \qquad \text{when } -1 < x \leqslant 1.$$

It remains to show ($\star\star\star$). Let $c = (1+\delta)\mu$. When $c \geqslant 6\mu$ we have $\delta = c/\mu - 1 \geqslant 5$, thus using the result in ($\star$) we have

$$\begin{aligned}
\mathbb{P}[X \geqslant c] &\leqslant \left(\frac{e^{\delta}}{(1+\delta)^{1+\delta}}\right)^{\mu} \\
&\leqslant \left(\frac{e}{1+\delta}\right)^{(1+\delta)\mu} \\
&\leqslant \left(\frac{e}{6}\right)^{c} \leqslant 2^{-c}.
\end{aligned}$$

□

B KNOWN PARADIGMS FOR SECURE AUTHENTICATION

In this appendix we prove security of the protocols from Figure 4.1 and 4.2.

Proof of Theorem 4.1. The proof is by reduction: Given a PPT adversary $\mathscr{A}$ running in time t and breaking active security of the protocol with advantage ϵ, we build a PPT forger $\mathscr{F}$ against Π_M. The adversary $\mathscr{F}$ runs in the experiment $\mathbf{Exp}^{\text{ufcma}}_{\text{MAC},\Pi_M}(\mathscr{F},\kappa)$ and has to "fake" the environment for $\mathscr{A}$, who is expecting to execute an active attack against the protocol of Figure 4.1. Thus, $\mathscr{F}$ uses $\mathscr{A}$ as follows:

1. Run $\mathscr{A}(1^\kappa)$.
2. When $\mathscr{A}$ replaces $\mathscr{V}$ (in the first phase of an active attack), receive the challenge m sent by $\mathscr{A}$. Query the tag oracle on m, obtaining $\phi \leftarrow \mathsf{TAG}_K(m)$ (for some unknown key $K \in \mathcal{K}$) and send ϕ as a response to $\mathscr{A}$'s challenge.
3. When $\mathscr{A}$ tries to be authenticated in place of $\mathscr{P}$ (in the second phase of an active attack), choose a random challenge $m^* \xleftarrow{\$} \mathcal{M}$ and send it to $\mathscr{A}$. Let ϕ^* be $\mathscr{A}$'s response. Output the forgery (m^*, ϕ^*).

Of course $\mathscr{F}$ runs in polynomial time. Moreover, it is not difficult to see that the simulation provided by $\mathscr{F}$ is perfect, as long as the message m^* is distinct from *all* the messages seen by $\mathscr{A}$ in the first phase (otherwise the forgery output by $\mathscr{F}$ is not fresh). Let Fresh be this event and Auth be the event that $\mathscr{A}$ is authenticated. We can write:

$$
\begin{aligned}
\epsilon &= \mathbb{P}[\text{Auth}] = \mathbb{P}[\text{Auth} \wedge \neg\text{Fresh}] + \mathbb{P}[\text{Auth} \wedge \text{Fresh}] \\
&= \mathbb{P}[\text{Auth} \wedge \neg\text{Fresh}] + \mathbb{P}\left[\mathbf{Exp}^{\text{ufcma}}_{\text{MAC},\Pi_M}(\mathscr{F},\kappa) = 1\right] \\
&\leqslant \mathbb{P}[\neg\text{Fresh}] + \mathbb{P}\left[\mathbf{Exp}^{\text{ufcma}}_{\text{MAC},\Pi_M}(\mathscr{F},\kappa) = 1\right] \\
&\leqslant [\text{since } \mathscr{A} \text{ asks } Q \text{ queries}] \\
&\leqslant \frac{Q}{\#\mathcal{M}} + \mathbb{P}\left[\mathbf{Exp}^{\text{ufcma}}_{\text{MAC},\Pi_M}(\mathscr{F},\kappa) = 1\right] \\
&\leqslant [\text{since } \Pi_M \text{ is secure}]
\end{aligned}
$$

$$\leqslant \frac{Q}{\#\mathcal{M}} + \epsilon_{\mathsf{MAC}},$$

as desired. □

Proof of Theorem 4.2. We present only a sketch of the proof, since it is similar to the proof of Theorem 4.1. Given a PPT adversary $\mathscr{A}$ running in time t and breaking security of the protocol in a MiM attack with advantage ϵ, we build a PPT forger $\mathscr{F}$ against Π_M. The adversary $\mathscr{F}$ runs in the experiment $\mathbf{Exp}^{\mathrm{ufcma}}_{\mathsf{MAC},\Pi_M}(\mathscr{F},\kappa)$ and has to "fake" the environment for $\mathscr{A}$, who is expecting to execute a MiM attack against the protocol of Figure 4.1. Thus, $\mathscr{F}$ uses $\mathscr{A}$ as follows:

1. Run $\mathscr{A}(1^\kappa)$.
2. Whenever $\mathscr{A}$ replaces $\mathscr{V}$, receive the challenge m_V sent by $\mathscr{A}$. Draw a random $m_P \xleftarrow{\$} \mathcal{M}_P$ and query the tag oracle on $m_P\|m_V$, obtaining $\phi \leftarrow \mathsf{TAG}_K(m_P\|m_V)$ (for some unknown key $K \in \mathcal{K}$) and send ϕ as a response to $\mathscr{A}$'s challenge. Output $\mathrm{sid}_P = m_P\|m_V$.
3. Whenever $\mathscr{A}$ tries to be authenticated in place of $\mathscr{P}$, choose a random challenge $m_V^* \xleftarrow{\$} \mathcal{M}_V$ and send it to $\mathscr{A}$. Let (m_P^*, ϕ^*) be $\mathscr{A}$'s response. Output the forgery $(m_P^*\|m_V^*, \phi^*)$.

Of course $\mathscr{F}$ runs in polynomial time. Moreover, it is not difficult to see that the simulation provided by $\mathscr{F}$ is perfect, as long as the message $m_P^*\|m_V^*$ is fresh. Let Fresh be this event and Auth be the event that $\mathscr{A}$ is authenticated. Define also the event Confused as the event that $\mathscr{A}$ manages to make two different prover instances output the same session id (at any time). We can write:

$$\begin{aligned}
\epsilon &= \mathbb{P}[\textsc{Auth}] + \mathbb{P}[\textsc{Confused}] \\
&= \mathbb{P}[\textsc{Auth} \wedge \neg\textsc{Fresh}] + \mathbb{P}[\textsc{Auth} \wedge \textsc{Fresh}] + \mathbb{P}[\textsc{Confused}] \\
&\leqslant \frac{Q}{\#\mathcal{M}_V} + \epsilon_{\mathsf{MAC}} + \mathbb{P}[\textsc{Confused}]\,.
\end{aligned}$$

It remains to bound $\mathbb{P}[\textsc{Confused}]$. The probability of Confused is the probability that, in Q queries, two instance of the prover output the same value $\mathrm{sid}_P = m_P\|m_V$ in the simulation above. Given two sids, they are equal with probability $1/\#\mathcal{M}_P$; given a third value, this will be equal to one of the other two with probability $2/\#\mathcal{M}_P$ and so forth. Thus

$$\mathbb{P}[\textsc{Confused}] = \sum_{i=1}^{Q-1} \frac{i}{\#\mathcal{M}_P} = \frac{1}{\#\mathcal{M}_P} \cdot \frac{Q(Q-1)}{2},$$

finishing the proof. □

C | SECURITY OF THE HB FAMILY

In this appendix we recall the security proof of HB and HB$^+$ as given by Katz and Shin [KS06]. We will only deal with the case $\tau' < 1/4$, and refer the reader to [KSS10] for the proof of the general case.

Proof of Theorem 4.3. We show that, if there exists a PPT attacker $\mathscr{A}$ breaking passive security of the HB protocol with probability ϵ, then we can build another PPT attacker $\mathscr{D}$ (using $\mathscr{A}$) solving the decisional version of $\mathsf{LPN}_{\tau,\kappa}$ with probability better than ϵ_{LPN} as in the theorem statement. The distinguisher $\mathscr{D}$ has access to an oracle which returns strings in $\mathbb{Z}_2^{\kappa+1}$ and must decide if the oracle is $\Lambda_{\tau,\kappa}(\mathbf{s})$ (for some $\mathbf{s} \xleftarrow{\$} \mathbb{Z}_2^{\kappa}$) or the uniform oracle $U_{\kappa+1}$. To do so, $\mathscr{D}$ relies on $\mathscr{A}$ as follows:

1. Whenever $\mathscr{A}$ asks to see a transcript of a honest execution of the HB protocol, query the oracle n times, obtaining pairs $(\mathbf{R}[i], \mathbf{z}[i])$ with $i = 1, \ldots, n$. Let $\mathbf{R} = (\mathbf{R}[1], \ldots, \mathbf{R}[n])$ and $\mathbf{z} = (\mathbf{z}[1], \ldots, \mathbf{z}[n])$. Give $(\mathbf{R}, \mathbf{z})$ back to $\mathscr{A}$.

2. When $\mathscr{A}$ tries to impersonate $\mathscr{P}$, query the oracle again for n times, obtaining pairs $(\mathbf{R}'[i], \mathbf{z}'[i])$ with $i = 1, \ldots, n$. Use $\mathbf{R}' = (\mathbf{R}'[1], \ldots, \mathbf{R}'[n])$ as a challenge for $\mathscr{A}$ and receive the answer $\mathbf{z}''$.

3. Let $\mathbf{z}' = (\mathbf{z}'[1], \ldots, \mathbf{z}'[n])$. Output 1 if and only if $w_H(\mathbf{z}' \oplus \mathbf{z}'') \leqslant 2\tau' \cdot n$.

We distinguish two cases.

THE ORACLE OF $\mathscr{D}$ IS $U_{\kappa+1}$. In this case $\mathscr{A}$ sees just uniformely random strings. Hence, the string $\mathbf{z}' \oplus \mathbf{z}''$ has uniform distribution over $\mathbb{Z}_2^{\kappa}$. Applying the Chernoff bound, there exists a constant c''_τ (depending only on τ) such that the probability that $\mathscr{D}$ outputs 1 is $2^{-n} \cdot \sum_{i=0}^{2\lfloor \tau' \cdot n \rfloor} \binom{n}{i} = 2^{-c''_\tau \cdot n}$.

THE ORACLE OF $\mathscr{D}$ IS $\Lambda_{\tau,\kappa}(\mathbf{s})$. In this case, $\mathscr{D}$'s simulation in the first phase of $\mathscr{A}$'s attack is perfect. Denote with $\mathbf{z}^* = (\mathbf{R}')^{\mathsf{T}} \cdot \mathbf{s}$ the vector containing the values of the true inner products (i.e., without noise) $\mathbf{z}^*[i] = (\mathbf{R}'[i])^{\mathsf{T}} \cdot \mathbf{s}$. Since $\mathscr{A}$ impersonates $\mathscr{P}$ with probability ϵ, we have that with probability at least ϵ

the vectors $\mathbf{z}''$ and $\mathbf{z}^*$ differ in at most $\tau' \cdot n$ values. On the other hand, since the vector $\mathbf{z}'$ satisfies $\mathbf{z}' = (\mathbf{R}')^\mathsf{T} \cdot \mathbf{s} \oplus \mathbf{e}$ where $\mathbf{e} \xleftarrow{\$} \mathsf{Bern}_\tau^n$, the vector $\mathbf{z}'$ is distributed as in a honest answer of $\mathscr{P}$ and thus (since the protocol has completeness error $\alpha_{\tau,n}$) $\mathbf{z}'$ and $\mathbf{z}^*$ differ in at most $\tau' \cdot n$ values with probability at most $\alpha_{\tau,n}$. We conclude that the probability that $\mathbf{z}'$ and $\mathbf{z}''$ differ in at most $2\tau' \cdot n$ values is at least $\epsilon - \alpha_{\tau,n}$.

Putting all together we have shown,

$$\left| \mathbb{P}\left[\mathscr{D}^{\Lambda_{\tau,\kappa}(\mathbf{s})}(1^\kappa) = 1\right] - \mathbb{P}\left[\mathscr{D}^{\mathsf{U}_{\kappa+1}}(1^\kappa) = 1\right] \right| \geqslant \epsilon - \alpha_{\tau,n} - 2^{-c''_\tau \cdot n},$$

and thus $\mathscr{D}$ solves the decisional $\Lambda_{\tau,\kappa}$ problem with probability at least $\epsilon_{\mathsf{LPN}} = \epsilon - \alpha_{\tau,n} - 2^{-c''_\tau \cdot n} = \epsilon - 2^{-\Theta(n)}$, as desired. □

Note that the proof yields a meaningful result only when $\tau < \tau' < 1/4$, since when $\tau \geqslant 1/4$ we have $2\tau' \cdot n \geqslant 2\tau \cdot n \geqslant n/2$ and thus $2^{-n} \sum_{i=0}^{2\lfloor \tau' \cdot n \rfloor} \binom{n}{i} \geqslant 1/2$.

We now turn to the proof of security for the HB^+ protocol.

Proof of Theorem 4.4. We show that if there exists a PPT adversary $\mathscr{A}$ breaking active security of the HB^+ protocol with probability ϵ, then we can build a different attacker $\mathscr{D}$ (using $\mathscr{A}$) able to solve the decisional version of the $\Lambda_{\tau,\kappa}$ problem with probability ϵ_{LPN} as in the theorem statement. The adversary $\mathscr{D}$ has access to an oracle returning strings in $\mathbb{Z}_2^{\kappa+1}$ and must distinguish if this oracle is $\Lambda_{\tau,\kappa}(\mathbf{s}_1)$ (for some $\mathbf{s}_1 \xleftarrow{\$} \mathbb{Z}_2^\kappa$) or the uniform oracle $\mathsf{U}_{\kappa+1}$. To do so, $\mathscr{D}$ uses $\mathscr{A}$ as follows:

1. Sample $\mathbf{s}_2 \xleftarrow{\$} \mathbb{Z}_2^\kappa$.

2. When $\mathscr{A}$ replaces $\mathscr{V}$ (in the first phase of an active attack), query the oracle n times, obtaining $(\mathbf{R}_1[i], \mathbf{z}[i])$ with $i = 1, \ldots, n$. Thus, send the matrix $\mathbf{R}_1 = (\mathbf{R}_1[1], \ldots, \mathbf{R}_1[n])$ to $\mathscr{A}$. Given the answer $\mathbf{R}_2$ of $\mathscr{A}$, set $\overline{\mathbf{z}} = \mathbf{R}_2^\mathsf{T} \cdot \mathbf{s}_2 \oplus \mathbf{z}$, where $\mathbf{z} = (\mathbf{z}[1], \ldots, \mathbf{z}[n])$ and send this value to $\mathscr{A}$.

3. When $\mathscr{A}$ replaces $\mathscr{P}$ (in the second phase of an active attack), receive the challenge $\mathbf{R}_1 = (\mathbf{R}_1[1], \ldots, \mathbf{R}_1[n])$. Sample $\mathbf{R}'_2 \xleftarrow{\$} \mathbb{Z}_2^{\kappa \times n}$ and send $\mathbf{R}'_2 = (\mathbf{R}'_2[1], \ldots, \mathbf{R}'_2[n])$ to $\mathscr{A}$. Let $\mathbf{z}' = (\mathbf{z}'[1], \ldots, \mathbf{z}'[n])$ be the answer of $\mathscr{A}$.

4. Rewind $\mathscr{A}$ to the point it already chose $\mathbf{R}_1$ and sample a different challenge $\mathbf{R}''_2 \xleftarrow{\$} \mathbb{Z}_2^{\kappa \times n}$, where $\mathbf{R}''_2 = (\mathbf{R}''_2[1], \ldots, \mathbf{R}''_2[n])$. Let $\mathbf{z}'' = (\mathbf{z}''[1], \ldots, \mathbf{z}''[n])$ be the new answer of $\mathscr{A}$.

5. Set $\mathbf{z}^{\oplus} = \mathbf{z}' \oplus \mathbf{z}''$, and $\mathbf{R}^*[i] = \mathbf{R}'_2[i] \oplus \mathbf{R}''_2[i]$, $\mathbf{R}^* = (\mathbf{R}^*[1], \ldots, \mathbf{R}^*[n])$. Compute $\mathbf{z}^* = (\mathbf{R}^*)^T \cdot \mathbf{s}_2$. Return 1 if and only if $\mathbf{z}^{\oplus}$ and $\mathbf{z}^*$ differ in at most $2\tau' \cdot n$ values.

We need to distinguish two cases.

THE ORACLE OF $\mathscr{D}$ IS $U_{\kappa+1}$. In the first phase, the values $\mathbf{z}[i]$ are all random. Thus, the answer $\bar{\mathbf{z}}$ simulated by $\mathscr{D}$ is also random and in particular independent on $\mathbf{s}_2$.

Note that in this case also the vector $\mathbf{z}^{\oplus}$ is random. Since the vectors $\mathbf{R}^*[i]$ (for $i = 1, \ldots, n$) are independent and uniformly distributed, they are also linearly independent with probability $2^n/2^\kappa$.[1] When this happens, the vector $\mathbf{z}^*$ is also random and the probability that $\mathbf{z}^*$ and $\mathbf{z}^{\oplus}$ differ in at most $2\tau' \cdot n$ values is exactly $2^{-n} \cdot \sum_{i=0}^{2\lfloor \tau' \cdot n \rfloor} \binom{n}{i} = 2^{-c''_\tau \cdot n}$ (applying the Chernoff bound). Hence, $\mathscr{D}$ returns 1 with probability at most $2^n/2^\kappa + 2^{-c''_\tau \cdot n}$.

THE ORACLE OF $\mathscr{D}$ IS $\Lambda_{\tau,\kappa}(\mathbf{s}_1)$. In this case the simulation of the first phase is perfect. Let ω be the randomness used to simulate the first phase of $\mathscr{A}$'s attack (this includes the vectors $\mathbf{s}_1, \mathbf{s}_2$, the randomness of $\mathscr{A}$ and the randomness used to answer $\mathscr{A}$'s queries). For a fixed ω, let ϵ_ω be the probability (over the random choice of $\mathbf{R}_2[1], \ldots, \mathbf{R}_2[n]$ in $\mathbf{R}_2$) that $\mathscr{A}$ is successful. Note that the probability that $\mathscr{A}$ answers correctly to both the challenges $\mathbf{R}'_2$ and $\mathbf{R}''_2$ is ϵ_ω^2. Averaging and applying Jensen's inequality[2] we get

$$\mathbb{E}_\omega\left[\epsilon_\omega^2\right] \geqslant \left(\mathbb{E}_\omega\left[\epsilon_\omega\right]\right)^2 = \epsilon^2.$$

1 The argument is similar to the one in the proof of Lemma 4.6. Let $\{\mathbf{r}_i\}_{i=1}^n$ be random vectors in $\mathbb{Z}_2^\kappa$ and denote with $\textsc{Dep}_i$ the event that vector $\mathbf{r}_i$ is linearly *dependent* on one among $\mathbf{r}_1, \ldots, \mathbf{r}_{i-1}$ (for $i = 0$ this is the event that $\mathbf{r}_1$ is the zero vector). Since the subspace spanned by $i-1$ vectors has dimension at most 2^{i-1}, the probability of $\textsc{Dep}_i$ is at most $2^{i-1}/2^\kappa$. Applying the union bound, we get

$$\mathbb{P}\left[\bigvee_{i=1}^{n} \textsc{Dep}_i\right] \leqslant 2^{-\kappa} \sum_{i=0}^{n-1} 2^i < \frac{2^n}{2^\kappa},$$

as desired.

2 In probability, Jensen's inequality states that if X is a random variable and f is a convex function, then

$$f(\mathbb{E}[X]) \leqslant \mathbb{E}[f(X)].$$

Thus, assuming that $\mathscr{A}$ is successful in both cases, we conclude that the vector $\mathbf{z}'$ and the vector containing the correct answers

$$\begin{aligned}\xi' &= \mathbf{R}_1^\intercal \cdot \mathbf{s}_1 \oplus (\mathbf{R}_2')^\intercal \cdot \mathbf{s}_2 \\ &= \left(\mathbf{R}_1[1]^\intercal \cdot \mathbf{s}_1 \oplus \mathbf{R}_2'[1]^\intercal \cdot \mathbf{s}_2, \ldots, \mathbf{R}_1[n]^\intercal \cdot \mathbf{s}_1 \oplus \mathbf{R}_2'[n]^\intercal \cdot \mathbf{s}_2\right),\end{aligned}$$

differ in at most $\tau' \cdot n$ values. Similarly, the vector $\mathbf{z}''$ and the vector containing the correct answers

$$\begin{aligned}\xi'' &= \mathbf{R}_1^\intercal \cdot \mathbf{s}_1 \oplus (\mathbf{R}_2'')^\intercal \cdot \mathbf{s}_2 \\ &= \left(\mathbf{R}_1[1]^\intercal \cdot \mathbf{s}_1 \oplus \mathbf{R}_2''[1]^\intercal \cdot \mathbf{s}_2, \ldots, \mathbf{R}_1[n]^\intercal \cdot \mathbf{s}_1 \oplus \mathbf{R}_2''[n]^\intercal \cdot \mathbf{s}_2\right),\end{aligned}$$

differ in at most $\tau' \cdot n$ values. Hence, the vector $\mathbf{z}' \oplus \mathbf{z}'' = \mathbf{z}^{\oplus}$ and the vector

$$\begin{aligned}\xi' \oplus \xi'' &= \mathbf{R}_1^\intercal \cdot \mathbf{s}_1 \oplus (\mathbf{R}_2')^\intercal \cdot \mathbf{s}_2 \oplus \mathbf{R}_1^\intercal \cdot \mathbf{s}_1 \oplus (\mathbf{R}_2'')^\intercal \cdot \mathbf{s}_2 \\ &= \left(\mathbf{R}_2'[1]^\intercal \cdot \mathbf{s}_2 \oplus \mathbf{R}_2''[1]^\intercal \cdot \mathbf{s}_2, \ldots, \mathbf{R}_2'[n]^\intercal \cdot \mathbf{s}_2 \oplus \mathbf{R}_2''[n]^\intercal \cdot \mathbf{s}_2\right) \\ &= \left((\mathbf{R}_2'[1] \oplus \mathbf{R}_2''[1])^\intercal \cdot \mathbf{s}_2, \ldots, (\mathbf{R}_2'[n] \oplus \mathbf{R}_2''[n])^\intercal \cdot \mathbf{s}_2\right) \\ &= (\mathbf{R}^*)^\intercal \cdot \mathbf{s}_2 = \mathbf{z}^*,\end{aligned}$$

differ in at most $2\tau' \cdot n$. We conclude that in this case $\mathscr{D}$ outputs 1 with probability ϵ^2.

Putting all together, we have shown

$$\left|\mathbb{P}\left[\mathscr{D}^{\Lambda_{\tau,\kappa}(\mathbf{s}_1)}(1^\kappa) = 1\right] - \mathbb{P}\left[\mathscr{D}^{U_{\kappa+1}}(1^\kappa) = 1\right]\right| \geqslant \epsilon^2 - \frac{2^n}{2^\kappa} - 2^{-c''_\tau \cdot n},$$

and thus $\mathscr{D}$ solves the $\Lambda_{\tau,\kappa}$ problem with probability at least $\epsilon_{\mathsf{LPN}} = \epsilon^2 + 2^{-\Theta(n)}$, as desired. □

D PROOF OF PIETRZAK'S RESULT

In this appendix we give a proof of Pietrzak's theorem on tamper resilience of LPN.

Proof of Lemma 4.5. We start with an adversary $\mathscr{D}_{\mathsf{SLPN}}$ solving the $\mathsf{SLPN}_{\tau,\kappa,d+g}$ problem, and we build an adversary $\mathscr{D}$ against the $\mathsf{LPN}_{\tau,d}$ problem. Note that the adversary $\mathscr{D}$ is given access to an oracle $\Lambda_{\tau,d}(\mathbf{x})$ (for some unknown $\mathbf{x} \xleftarrow{\$} \mathbb{Z}_2^d$) and needs to simulate the answers from the $\Gamma_{\tau,\kappa,d+g}(\cdot,\cdot)$ oracle for another (unknown) vector $\hat{\mathbf{x}} \xleftarrow{\$} \mathbb{Z}_2^\kappa$.

Whenever $\mathscr{D}_{\mathsf{SLPN}}$ asks a query $(\mathbf{A}, \mathbf{b})$, adversary $\mathscr{D}$ checks that $\mathrm{rank}(\mathbf{A}) \geqslant d + g$; if this is not the case it returns $\perp$. Otherwise, it first queries the $\Lambda_{\tau,d}(\mathbf{x})$ oracle to get an LPN sample $(\mathbf{r}, \mathbf{r}^\mathsf{T} \cdot \mathbf{x} \oplus e)$. Then, it samples $\mathbf{W} \xleftarrow{\$} \mathbb{Z}_2^{\kappa \times d}$, $\mathbf{w} \xleftarrow{\$} \mathbb{Z}_2^\kappa$ and (implicitly) defines $\hat{\mathbf{x}} = \mathbf{W} \cdot \mathbf{x} \oplus \mathbf{w}$. Define the set $\mathcal{S} \subseteq \mathbb{Z}_2^\kappa$ of solutions to the system of linear equations:

$$\mathcal{S} = \left\{ \mathbf{y} : \ \mathbf{y} \cdot \mathbf{A} \cdot \mathbf{W} = \mathbf{r}^\mathsf{T} \right\}.$$

Note that whenever $\mathbf{A} \cdot \mathbf{W}$ has rank $\geqslant d$, the set $\mathcal{S}$ is not empty as the system of linear equations above is overdefined. At this point $\mathscr{D}$ samples $\hat{\mathbf{r}} \xleftarrow{\$} \mathcal{S}$ and outputs the sample

$$(\hat{\mathbf{r}}, \mathbf{r}^\mathsf{T} \cdot \mathbf{x} \oplus e \oplus z),$$

where z is computed from known values as $z = \hat{\mathbf{r}}^\mathsf{T} \cdot \mathbf{A} \cdot \mathbf{w} \oplus \hat{\mathbf{r}}^\mathsf{T} \cdot \mathbf{b}$.

For the analysis, note that $\mathscr{D}$ runs in time $t \approx t_{\mathsf{SLPN}}$. It remains to show that simulation performed by $\mathscr{D}$ is correct. This is shown in the following claims.

Claim 12. *If $\mathbf{V} = \mathbf{A} \cdot \mathbf{W}$ has rank $\geqslant d$, then $\hat{\mathbf{r}} \xleftarrow{\$} \mathcal{S}$ is uniformly random (given $\mathbf{A}, \mathbf{b}, \mathbf{W}, \mathbf{w}$).*

Proof. Recall that $\mathbf{V} \in \mathbb{Z}_2^{\kappa \times d}$. Fix some $\mathbf{v} \in \mathbb{Z}_2^\kappa$ such that $w_H(\mathbf{v}) = d$ and $\mathbf{V}_{\downarrow \mathbf{v}}$ has full rank. (Such $\mathbf{v}$ exists, since $\mathbf{V}$ has rank at least d.) We know that $\hat{\mathbf{r}} \xleftarrow{\$} \mathcal{S}$ is a random solution to the equation $\hat{\mathbf{r}} \cdot \mathbf{V} = \mathbf{r}^\mathsf{T}$. Using the fact that $\hat{\mathbf{r}} \cdot \mathbf{V} = \hat{\mathbf{r}}_{\downarrow \mathbf{v}} \cdot \mathbf{V}_{\downarrow \mathbf{v}} \oplus \hat{\mathbf{r}}_{\downarrow \bar{\mathbf{v}}} \cdot \mathbf{V}_{\downarrow \bar{\mathbf{v}}}$, we can write

$$\hat{\mathbf{r}}_{\downarrow \mathbf{v}} \cdot \mathbf{V}_{\downarrow v} = \mathbf{r}^\mathsf{T} \oplus \hat{\mathbf{r}}_{\downarrow \bar{\mathbf{v}}} \cdot \mathbf{V}_{\downarrow \bar{\mathbf{v}}}. \tag{D.1}$$

Hence, we can sample a random $\hat{\mathbf{r}}$ as follows. We first choose a random $\hat{\mathbf{r}}_{\downarrow\overline{\mathbf{v}}} \xleftarrow{\$} \mathbb{Z}_2^{\kappa-d}$. The remaining d positions are then uniquely determined by $\mathbf{r}$ and given by the solution of Eq. (D.1). Since $\mathbf{V}_{\downarrow\mathbf{v}}$ is a full rank square matrix, Eq. (D.1) defines a bijection between $\hat{\mathbf{r}}_{\downarrow\mathbf{v}}$ and $\mathbf{r}$. As $\mathbf{r}$ is random and $\hat{\mathbf{r}}_{\downarrow\overline{\mathbf{v}}}$ are both random, so is $\hat{\mathbf{r}}_{\downarrow\mathbf{v}}$ and thus $\hat{\mathbf{r}}$. □

Claim 13. *Adversary $\mathscr{D}$ perfectly simulates access to the oracle $\mathsf{SLPN}_{\tau,\kappa,d+g}(\cdot,\cdot)$, with respect to secret $\hat{\mathbf{x}} = \mathbf{W}\cdot\mathbf{x}\oplus\mathbf{w}$.*

Proof. This is because,

$$\begin{aligned}
\hat{\mathbf{r}},\hat{\mathbf{r}}^{\mathsf{T}}\cdot(\mathbf{A}\cdot\hat{\mathbf{x}}\oplus\mathbf{b})\oplus e &= \hat{\mathbf{r}},\hat{\mathbf{r}}^{\mathsf{T}}\cdot(\mathbf{A}\cdot(\mathbf{W}\cdot\mathbf{x}\oplus\mathbf{w})\oplus\mathbf{b})\oplus e\\
&= \hat{\mathbf{r}},\hat{\mathbf{r}}^{\mathsf{T}}\cdot\mathbf{A}\cdot\mathbf{W}\cdot\mathbf{x}\oplus\hat{\mathbf{r}}^{\mathsf{T}}\cdot\mathbf{A}\cdot\mathbf{w}\oplus\hat{\mathbf{r}}^{\mathsf{T}}\cdot\mathbf{b}\oplus e\\
&= [\text{since } \hat{\mathbf{r}}\in\mathcal{S}]\\
&= \hat{\mathbf{r}},\mathbf{r}^{\mathsf{T}}\cdot\mathbf{x}\oplus\underbrace{\hat{\mathbf{r}}^{\mathsf{T}}\cdot\mathbf{A}\cdot\mathbf{w}\oplus\hat{\mathbf{r}}^{\mathsf{T}}\cdot\mathbf{b}}_{z}\oplus e\\
&= \hat{\mathbf{r}},\mathbf{r}^{\mathsf{T}}\cdot\mathbf{x}\oplus e\oplus z.
\end{aligned}$$

□

Claim 14. *With probability at least 2^{-g} the set $\mathcal{S}$ is not empty.*

Proof. Recall that the set $\mathcal{S}$ is empty when $\mathbf{V}=\mathbf{A}\cdot\mathbf{W}\in\mathbb{Z}_2^{\kappa\times d}$ has rank less than d, where $\mathbf{A}\in\mathbb{Z}_2^{\kappa\times\kappa}$ has rank $\mathrm{rank}(\mathbf{A})\geqslant d+g$ and $\mathbf{W}\xleftarrow{\$}\mathbb{Z}_2^{\kappa\times d}$.

Denote with $\wp(d,g)$ the probability that a random matrix in $\mathbb{Z}_2^{(d+g)\times d}$ has rank less than d. Since the matrix $\mathbf{A}$ has rank at least $d+g$, we can assume, without loss of generality, that the first $d+g$ rows of $\mathbf{A}$ are linearly independent. Since the matrix $\mathbf{W}$ is random, the upper $(d+g)\times d$ matrix of $\mathbf{V}=\mathbf{A}\cdot\mathbf{W}$ is random in $\mathbb{Z}_2^{(d+g)\times d}$ and thus it has rank less than d with probability at most $\wp(d,g)$. We conclude that $\mathbf{V}$ has rank strictly less than d exactly with the same probability. Using Lemma 4.6, we see that this probability is bounded by 2^{-g}. □

Applying the union bound, we can upper bound the probability that for any of the Q queries the matrix $\mathbf{V}=\mathbf{A}\cdot\mathbf{W}$ has rank less than d by $Q\cdot 2^{-g}$. This error probability is thus an upper bound on the gap of the success probability ϵ_{SLPN} of $\mathscr{D}_{\mathsf{SLPN}}$ and the success probability ϵ we get in breaking LPN using the transformation.

Finally, we need to consider the fact that the queries $(\mathbf{A},\mathbf{b})$ chosen by $\mathscr{D}_{\mathsf{SLPN}}$ are chosen adaptively. To show that adaptivity does not help in picking an $\mathbf{A}$

where $\mathbf{A} \cdot \mathbf{W}$ has rank $< d$ we must show that the view of $\mathscr{D}$ is independent of $\mathbf{W}$ (except for the fact that so far no query was made where $\text{rank}(\mathbf{A} \cdot \mathbf{W}) < d$). To see this, first note that $\hat{\mathbf{x}} = \mathbf{W} \cdot \mathbf{x} \oplus \mathbf{w}$ is independent of $\mathbf{W}$ as it is blinded with a uniform $\mathbf{w}$. In fact, the only reason we use this blinding is to enforce this independence. The $\hat{\mathbf{r}}$ are independent as they are uniform given $\mathbf{W}$ as shown in the first claim in the proof. □

EXTENSIONS. The result of Pietrzak [Pie10] is actually more general than the one we have proved here. The reduction still hold even if the adversary can (adaptively) choose arbitrary affine transformations $\phi_r(\mathbf{r}) = \mathbf{A}_r \cdot \mathbf{r} \oplus \mathbf{b}_r$ and $\phi_x(\mathbf{x}) = \mathbf{A}_x \cdot \mathbf{x} \oplus \mathbf{b}_x$ and learn

$$(\mathbf{r}, \phi_r(\mathbf{r})^T \cdot \phi_x(\mathbf{x}) \oplus e),$$

as long as the matrices $\mathbf{A}_x, \mathbf{A}_r$ overlap in a subspace of dimension at least $d + g$.

Moreover the result is still valid if we replace $\mathbb{Z}_2$ with $\mathbb{Z}_q$ (where q is a prime or a prime power), and Bern_τ is replaced by any distribution χ over $\mathbb{Z}_q$ such that the LWE problem with distribution χ is hard (for instance the "discretised normal error" distribution $\text{Gau}_{q,\tau}$ in the case of LWE).

BIBLIOGRAPHY

[ABB10] Shweta Agrawal, Dan Boneh, and Xavier Boyen. "Efficient Lattice (H)IBE in the Standard Model". In: *EUROCRYPT*. 2010, pp. 553–572 (cit. on pp. 75, 76, 83).

[AD97] Miklós Ajtai and Cynthia Dwork. "A Public-Key Cryptosystem with Worst-Case/Average-Case Equivalence". In: *STOC*. 1997, pp. 284–293 (cit. on p. 25).

[AGV09] Adi Akavia, Shafi Goldwasser, and Vinod Vaikuntanathan. "Simultaneous Hardcore Bits and Cryptography against Memory Attacks". In: *TCC*. 2009, pp. 474–495 (cit. on p. 23).

[AHI11] Benny Applebaum, Danny Harnik, and Yuval Ishai. *Semantic Security Under Related-Key Attacks and Applications*. to appear in the 2nd Symposium on Innovations in Computer Science (ICS 2011). Full version available on `http://eprint.iacr.org/`. 2011 (cit. on pp. 54, 55).

[AK97] Ross J. Anderson and Markus G. Kuhn. "Low Cost Attacks on Tamper Resistant Devices". In: *Security Protocols Workshop*. 1997, pp. 125–136 (cit. on p. 31).

[AM09] Divesh Aggarwal and Ueli M. Maurer. "Breaking RSA Generically Is Equivalent to Factoring". In: *EUROCRYPT*. 2009, pp. 36–53 (cit. on p. 8).

[Bak10] Jacob R. Baker. *CMOS: Circuit Design, Layout, and Simulation (3rd edition)*. Wiley-IEEE Press, 2010 (cit. on p. 29).

[BB04] Dan Boneh and Xavier Boyen. "Efficient Selective-ID Secure Identity-Based Encryption Without Random Oracles". In: *EUROCRYPT*. 2004, pp. 223–238 (cit. on p. 75).

[BBK03] Elad Barkan, Eli Biham, and Nathan Keller. "Instant Ciphertext-Only Cryptanalysis of GSM Encrypted Communication". In: *CRYPTO*. 2003, pp. 600–616 (cit. on p. 2).

[BC08] Julien Bringer and Hervé Chabanne. "Trusted-HB: A Low-Cost Version of HB^+ Secure Against Man-in-the-Middle Attacks". In: *IEEE Transactions on Information Theory* 54.9 (2008), pp. 4339–4342 (cit. on p. 65).

[BC10] Mihir Bellare and David Cash. "Pseudorandom Functions and Permutations Provably Secure against Related-Key Attacks". In: *CRYPTO*. 2010, pp. 666–684 (cit. on pp. 54, 55).

[BCD06] Julien Bringer, Hervé Chabanne, and Emmanuelle Dottax. "HB^{++}: A Lightweight Authentication Protocol Secure against Some Attacks". In: *SecPerU*. 2006, pp. 28–33 (cit. on p. 65).

[BCDG10] Alexandre Berzati, Cécile Canovas-Dumas, and Louis Goubin. "Public Key Perturbation of Randomized RSA Implementations". In: *CHES*. 2010, pp. 306–319 (cit. on p. 11).

[BCG08] Alexandre Berzati, Cécile Canovas, and Louis Goubin. "Perturbating RSA Public Keys: An Improved Attack". In: *CHES*. 2008, pp. 380–395 (cit. on p. 11).

[BCK98] Mihir Bellare, Ran Canetti, and Hugo Krawczyk. "A Modular Approach to the Design and Analysis of Authentication and Key Exchange Protocols (Extended Abstract)". In: *STOC*. 1998, pp. 419–428 (cit. on p. 60).

[BCM11] Mihir Bellare, David Cash, and Rachel Miller. *Cryptography Secure Against Related-Key Attacks and Tampering*. Cryptology ePrint Archive, Report 2011/252. http://eprint.iacr.org/. 2011 (cit. on pp. 54, 55).

[BDL01] Dan Boneh, Richard A. DeMillo, and Richard J. Lipton. "On the Importance of Eliminating Errors in Cryptographic Computations". In: *J. Cryptology* 14.2 (2001), pp. 101–119 (cit. on pp. 10, 11).

[BDL97] Dan Boneh, Richard A. DeMillo, and Richard J. Lipton. "On the Importance of Checking Cryptographic Protocols for Faults (Extended Abstract)". In: *EUROCRYPT*. 1997, pp. 37–51 (cit. on p. 31).

[Bel+01] Mihir Bellare et al. "Identification Protocols Secure against Reset Attacks". In: *EUROCRYPT*. 2001, pp. 495–511 (cit. on p. 61).

[Ber+09] Alexandre Berzati et al. "Fault Attacks on RSA Public Keys: Left-To-Right Implementations Are Also Vulnerable". In: *CT-RSA*. 2009, pp. 414–428 (cit. on p. 11).

[BG10] Zvika Brakerski and Shafi Goldwasser. "Circular and Leakage Resilient Public-Key Encryption under Subgroup Indistinguishability - (or: Quadratic Residuosity Strikes Back)". In: *CRYPTO*. 2010, pp. 1–20 (cit. on p. 23).

[BHN11] Carl Bosley, Kristiyan Haralambiev, and Antonio Nicolosi. HB^N: *An HB-like protocol secure against man-in-the-middle attacks*. Cryptology ePrint Archive, Report 2011/350. http://eprint.iacr.org/. 2011 (cit. on p. 92).

[BK03] Mihir Bellare and Tadayoshi Kohno. "A Theoretical Treatment of Related-Key Attacks: RKA-PRPs, RKA-PRFs, and Applications". In: *EUROCRYPT*. 2003, pp. 491–506 (cit. on pp. 54, 55).

[BKW00] Avrim Blum, Adam Kalai, and Hal Wasserman. "Noise-tolerant learning, the parity problem, and the statistical query model". In: *STOC*. 2000, pp. 435–440 (cit. on p. 25).

[BKW03] Avrim Blum, Adam Kalai, and Hal Wasserman. "Noise-tolerant learning, the parity problem, and the statistical query model". In: *J. ACM* 50.4 (2003), pp. 506–519 (cit. on p. 25).

[Ble98] Daniel Bleichenbacher. "Chosen Ciphertext Attacks Against Protocols Based on the RSA Encryption Standard PKCS #1". In: *CRYPTO*. 1998, pp. 1–12 (cit. on p. 2).

[Bol+07] Alexandra Boldyreva et al. "A Closer Look at PKI: Security and Efficiency". In: *Public Key Cryptography*. 2007, pp. 458–475 (cit. on p. 22).

[Boy10] Xavier Boyen. "Lattice Mixing and Vanishing Trapdoors: A Framework for Fully Secure Short Signatures and More". In: *Public Key Cryptography*. 2010, pp. 499–517 (cit. on pp. 75, 76, 83, 84, 87).

[BPR00] Mihir Bellare, David Pointcheval, and Phillip Rogaway. "Authenticated Key Exchange Secure against Dictionary Attacks". In: *EUROCRYPT*. 2000, pp. 139–155 (cit. on p. 61).

[BR96] Mihir Bellare and Phillip Rogaway. "The Exact Security of Digital Signatures - How to Sign with RSA and Rabin". In: *EUROCRYPT*. 1996, pp. 399–416 (cit. on p. 8).

[Bra+10] Zvika Brakerski et al. "Overcoming the Hole in the Bucket: Public-Key Cryptography Resilient to Continual Memory Leakage". In: *FOCS*. 2010, pp. 501–510 (cit. on p. 23).

[Bri+06] Eric Brier et al. "Why One Should Also Secure RSA Public Key Elements". In: *CHES*. 2006, pp. 324–338 (cit. on p. 11).

[Bri+11] Eric Brier et al. *Modulus Fault Attacks Against RSA-CRT Signatures*. Cryptology ePrint Archive, Report 2011/388. http://eprint.iacr.org/. 2011 (cit. on p. 11).

[Can+00] Ran Canetti et al. "Exposure-Resilient Functions and All-or-Nothing Transforms". In: *EUROCRYPT*. 2000, pp. 453–469 (cit. on p. 23).

[Car65] Lewis Carroll. *Alice's Adventures in Wonderland*. Mac Millan & Co., 1865 (cit. on pp. 1, 3, 27).

[Car71] Lewis Carroll. *Through the Looking-Glass, and What Alice Found There*. Mac Millan & Co., 1871 (cit. on pp. 15, 57).

[CD09] Ronald Cramer and Ivan Damgård. "On the Amortized Complexity of Zero-Knowledge Protocols". In: *CRYPTO*. 2009, pp. 177–191 (cit. on pp. 76, 86).

[Cho+10] Sherman S. M. Chow et al. "Practical leakage-resilient identity-based encryption from simple assumptions". In: *ACM Conference on Computer and Communications Security*. 2010, pp. 152–161 (cit. on p. 23).

[CKS05] Christian Cachin, Klaus Kursawe, and Victor Shoup. "Random Oracles in Constantinople: Practical Asynchronous Byzantine Agreement Using Cryptography". In: *J. Cryptology* 18.3 (2005), pp. 219–246 (cit. on p. 22).

[Cor02] Jean-Sébastien Coron. "Optimal Security Proofs for PSS and Other Signature Schemes". In: *EUROCRYPT*. 2002, pp. 272–287 (cit. on p. 8).

[DDV10] Francesco Davì, Stefan Dziembowski, and Daniele Venturi. "Leakage-Resilient Storage". In: *SCN*. 2010, pp. 121–137 (cit. on p. 24).

[DF11] Stefan Dziembowski and Sebastian Faust. *Leakage-Resilient Cryptography From the Inner-Product Extractor*. Cryptology ePrint Archive, Report 2011/519. http://eprint.iacr.org/. 2011 (cit. on p. 24).

[DH76] Whitfield Diffie and Martin E. Hellman. "New directions in cryptography". In: *IEEE Transactions on Information Theory* 22.6 (1976), pp. 644–654 (cit. on p. 22).

[DKL09] Yevgeniy Dodis, Yael Tauman Kalai, and Shachar Lovett. "On cryptography with auxiliary input". In: *STOC*. 2009, pp. 621–630 (cit. on p. 24).

[DKW11a] Stefan Dziembowski, Tomasz Kazana, and Daniel Wichs. "Key-Evolution Schemes Resilient to Space-Bounded Leakage". In: *CRYPTO*. 2011, pp. 335–353 (cit. on p. 24).

[DKW11b] Stefan Dziembowski, Tomasz Kazana, and Daniel Wichs. "One-Time Computable Self-erasing Functions". In: *TCC*. 2011, pp. 125–143 (cit. on p. 24).

[Dod+10a] Yevgeniy Dodis et al. "Efficient Public-Key Cryptography in the Presence of Key Leakage". In: *ASIACRYPT*. 2010, pp. 613–631 (cit. on p. 23).

[Dod+10b] Yevgeniy Dodis et al. "Public-Key Encryption Schemes with Auxiliary Inputs". In: *TCC*. 2010, pp. 361–381 (cit. on p. 24).

[Dod+11] Yevgeniy Dodis et al. *Storing Secrets on Continually Leaky Devices*. Cryptology ePrint Archive, Report 2011/369. http://eprint.iacr.org/. 2011 (cit. on p. 24).

[DP08] Stefan Dziembowski and Krzysztof Pietrzak. "Leakage-Resilient Cryptography". In: *FOCS*. 2008, pp. 293–302 (cit. on p. 24).

[DPW10] Stefan Dziembowski, Krzysztof Pietrzak, and Daniel Wichs. "Non-Malleable Codes". In: *ICS*. 2010, pp. 434–452 (cit. on pp. 54, 55).

[DSS01] Yevgeniy Dodis, Amit Sahai, and Adam Smith. "On Perfect and Adaptive Security in Exposure-Resilient Cryptography". In: *EUROCRYPT*. 2001, pp. 301–324 (cit. on p. 23).

[Fau+10a] Sebastian Faust et al. "Leakage-Resilient Signatures". In: *TCC*. 2010, pp. 343–360 (cit. on p. 24).

[Fau+10b] Sebastian Faust et al. "Protecting Circuits from Leakage: the Computationally Bounded and Noisy Cases". In: *EUROCRYPT*. 2010, pp. 135–156 (cit. on pp. 24, 36).

[FPV11] Sebastian Faust, Krzysztof Pietrzak, and Daniele Venturi. "Tamper-Proof Circuits: How to Trade Leakage for Tamper-Resilience". In: *ICALP (1)*. 2011, pp. 391–402 (cit. on pp. ix, 12).

[FS09] Dmitry Frumkin and Adi Shamir. *Un-Trusted-HB: Security Vulnerabilities of Trusted-HB*. Cryptology ePrint Archive, Report 2009/044. http://eprint.iacr.org/. 2009.

[FS96] Jean-Bernard Fischer and Jacques Stern. "An Efficient Pseudo-Random Generator Provably as Secure as Syndrome Decoding". In: *EUROCRYPT*. 1996, pp. 245–255 (cit. on p. 92).

[Für09] Martin Fürer. "Faster Integer Multiplication". In: *SIAM J. Comput.* 39.3 (2009), pp. 979–1005 (cit. on p. 92).

[GA03] Sudhakar Govindavajhala and Andrew W. Appel. "Using Memory Errors to Attack a Virtual Machine". In: *IEEE Symposium on Security and Privacy*. 2003, pp. 154–165 (cit. on pp. 30, 31).

[Gen+04] Rosario Gennaro et al. "Algorithmic Tamper-Proof (ATP) Security: Theoretical Foundations for Security against Hardware Tampering". In: *TCC*. 2004, pp. 258–277 (cit. on pp. 36, 54, 55).

[GGM86] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. "How to construct random functions". In: *J. ACM* 33.4 (1986), pp. 792–807 (cit. on pp. 75, 91, 92).

[GL10] David Goldenberg and Moses Liskov. "On Related-Secret Pseudorandomness". In: *TCC*. 2010, pp. 255–272 (cit. on pp. 54, 55).

[GM82] Shafi Goldwasser and Silvio Micali. "Probabilistic Encryption and How to Play Mental Poker Keeping Secret All Partial Information". In: *STOC*. 1982, pp. 365–377 (cit. on p. 3).

[GMO01] Karine Gandolfi, Christophe Mourtel, and Francis Olivier. "Electromagnetic Analysis: Concrete Results". In: *CHES*. Generators. 2001, pp. 251–261 (cit. on p. 10).

[GMR85] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. "The Knowledge Complexity of Interactive Proof-Systems (Extended Abstract)". In: *STOC*. 1985, pp. 291–304 (cit. on p. 34).

[Gol+08] Zbigniew Golebiewski et al. *Practical Attacks on* HB *and* HB^+ *Protocols*. Cryptology ePrint Archive, Report 2008/241. http://eprint.iacr.org/. 2008 (cit. on p. 65).

[GR10] Shafi Goldwasser and Guy N. Rothblum. "Securing Computation against Continuous Leakage". In: *CRYPTO*. 2010, pp. 59–79 (cit. on pp. 24, 36).

[GRS05] Henri Gilbert, Matt Robshaw, and Herve Sibert. *An Active Attack Against* HB^+ *- A Provably Secure Lightweight Authentication Protocol*. Cryptology ePrint Archive, Report 2005/237. http://eprint.iacr.org/. 2005 (cit. on p. 65).

[GRS08a] Henri Gilbert, Matthew J. B. Robshaw, and Yannick Seurin. "Good Variants of HB^+ Are Hard to Find". In: *Financial Cryptography*. 2008, pp. 156–170 (cit. on p. 65).

[GRS08b] Henri Gilbert, Matthew J. B. Robshaw, and Yannick Seurin. "HB$^{\#}$: Increasing the Security and Efficiency of HB^{+}". In: *EUROCRYPT*. 2008, pp. 361–378 (cit. on p. 89).

[GRS08c] Henri Gilbert, Matthew J. B. Robshaw, and Yannick Seurin. "How to Encrypt with the LPN Problem". In: *ICALP (2)*. 2008, pp. 679–690 (cit. on p. 65).

[GRS08d] Henri Gilbert, Matthew J. B. Robshaw, and Yannick Seurin. "HB$^{\#}$: Increasing the Security and Efficiency of HB^{+}". In: *EUROCRYPT*. 2008, pp. 361–378 (cit. on p. 65).

[HB01] Nicholas J. Hopper and Manuel Blum. "Secure Human Identification Protocols". In: *ASIACRYPT*. 2001, pp. 52–66 (cit. on pp. 13, 58, 62, 91).

[Ish+06] Yuval Ishai et al. "Private Circuits II: Keeping Secrets in Tamperable Circuits". In: *EUROCRYPT*. 2006, pp. 308–327 (cit. on pp. 12, 14, 27, 34–37, 40, 42, 50, 52, 53, 55).

[ISW03] Yuval Ishai, Amit Sahai, and David Wagner. "Private Circuits: Securing Hardware against Probing Attacks". In: *CRYPTO*. 2003, pp. 463–481 (cit. on pp. 23, 50–52).

[JV10] Ali Juma and Yevgeniy Vahlis. "Protecting Cryptographic Keys against Continual Leakage". In: *CRYPTO*. 2010, pp. 41–58 (cit. on pp. 24, 36).

[JW05] Ari Juels and Stephen A. Weis. "Authenticating Pervasive Devices with Human Protocols". In: *CRYPTO*. 2005, pp. 293–308 (cit. on pp. 63, 64, 91).

[Kil+11] Eike Kiltz et al. "Efficient Authentication from Hard Learning Problems". In: *EUROCRYPT*. 2011, pp. 7–26 (cit. on pp. ix, 13, 58, 65).

[KJJ99] Paul Kocher, Joshua Jaffe, and Benjamin Jun. "Differential Power Analysis". In: *CRYPTO*. Ed. by M. Wiener. Lecture Notes in Computer Science No. 1666. Springer-Verlag, 1999, pp. 388–397 (cit. on p. 10).

[KK99] Oliver Kömmerling and Markus G. Kuhn. "Design principles for tamper-resistant smartcard processors". In: *Smartcard*. 1999, pp. 9–20 (cit. on p. 31).

[KKS11] Yael Tauman Kalai, Bhavana Kanukurthi, and Amit Sahai. "Cryptography with Tamperable and Leaky Memory". In: *CRYPTO*. 2011, pp. 373–390 (cit. on p. 55).

[KL07] Jonathan Katz and Yehunda Lindell. *Introduction to Modern Cryptography*. Chapman and Hall/CRC Press, 2007 (cit. on p. 42).

[Koc96] Paul C. Kocher. "Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems". In: *CRYPTO*. Ed. by Neal Koblitz. Lecture Notes in Computer Science No. 1109. Springer-Verlag, 1996, pp. 104–113 (cit. on p. 9).

[KP10] Eike Kiltz and Krzysztof Pietrzak. "Leakage Resilient ElGamal Encryption". In: *ASIACRYPT*. 2010, pp. 595–612 (cit. on p. 24).

[KS06] Jonathan Katz and Ji Sun Shin. "Parallel and Concurrent Security of the HB and HB$^+$ Protocols". In: *EUROCRYPT*. 2006, pp. 73–87 (cit. on pp. 25, 63, 101).

[KSS10] Jonathan Katz, Ji Sun Shin, and Adam Smith. "Parallel and Concurrent Security of the HB and HB$^+$ Protocols". In: *J. Cryptology* 23.3 (2010), pp. 402–421 (cit. on pp. 25, 63, 101).

[KV09] Jonathan Katz and Vinod Vaikuntanathan. "Signature Schemes with Bounded Leakage Resilience". In: *ASIACRYPT*. 2009, pp. 703–720 (cit. on p. 23).

[Len96] Arien K. Lenstra. "Memo on RSA Signature Generation in the Presence of Faults". Manuscript. 1996 (cit. on p. 11).

[LF06] Éric Levieil and Pierre-Alain Fouque. "An Improved LPN Algorithm". In: *SCN*. 2006, pp. 348–359 (cit. on p. 25).

[LLW11] Allison B. Lewko, Mark Lewko, and Brent Waters. "How to leak on key updates". In: *STOC*. 2011, pp. 725–734 (cit. on p. 24).

[LMM] Xuefei Leng, Keith Mayes, and Konstantinos Markantonakis. "HB – MP$^+$ Protocol: An Improvement on the HB – MP Protocol". In: () (cit. on p. 65).

[Luc04] Stefan Lucks. "Ciphers Secure against Related-Key Attacks". In: *FSE*. 2004, pp. 359–370 (cit. on pp. 54, 55).

[MP07] Jorge Munilla and Alberto Peinado. "HB – MP: A further step in the HB-family of lightweight authentication protocols". In: *Computer Networks* 51.9 (2007), pp. 2262–2267 (cit. on p. 65).

[MR04] Silvio Micali and Leonid Reyzin. "Physically Observable Cryptography (Extended Abstract)". In: *TCC*. 2004, pp. 278–296 (cit. on pp. 12, 23).

[Mui06] James A. Muir. "Seifert's RSA Fault Attack: Simplified Analysis and Generalizations". In: *ICICS*. 2006, pp. 420–434 (cit. on p. 11).

[NPR99] Moni Naor, Benny Pinkas, and Omer Reingold. "Distributed Pseudo-random Functions and KDCs". In: *EUROCRYPT*. 1999, pp. 327–346 (cit. on p. 22).

[NS09] Moni Naor and Gil Segev. "Public-Key Cryptosystems Resilient to Key Leakage". In: *CRYPTO*. 2009, pp. 18–35 (cit. on pp. 23, 24).

[OOV08] Khaled Ouafi, Raphael Overbeck, and Serge Vaudenay. "On the Security of $HB^{\#}$ against a Man-in-the-Middle Attack". In: *ASIACRYPT*. 2008, pp. 108–124 (cit. on p. 65).

[Ott04] Martin Otto. "Fault Attacks and Countermeasures". PhD thesis. Institut für Informatik, Universität Paderborn, 2004 (cit. on p. 32).

[Pei09] Chris Peikert. "Public-key cryptosystems from the worst-case shortest vector problem: extended abstract". In: *STOC*. 2009, pp. 333–342 (cit. on pp. 25, 90).

[Pet97] Ivars Peterson. "Chinks in digital armor — exploiting faults to break smart-card cryptosystems". In: *Science News* 151.5 (1997), pp. 78–79 (cit. on p. 31).

[Pie09] Krzysztof Pietrzak. "A Leakage-Resilient Mode of Operation". In: *EUROCRYPT*. 2009, pp. 462–482 (cit. on p. 24).

[Pie10] Krzysztof Pietrzak. *Subspace LWE*. 2010. URL: http://homepages.cwi.nl/~pietrzak/publications/SLWE.pdf (cit. on pp. 13, 58, 66, 90, 107).

[QS01] Jean-Jacques Quisquater and David Samyde. "ElectroMagnetic Analysis (EMA): Measures and Counter-Measures for Smart Cards". In: *E-smart*. 2001, pp. 200–210 (cit. on p. 10).

[QS02] Jean-Jacques Quisquater and David Samyde. "Eddy current for magnetic analysis with active sensor". In: *E-Smart*. 2002 (cit. on p. 31).

[Reg05] Oded Regev. "On lattices, learning with errors, random linear codes, and cryptography". In: *STOC*. 2005, pp. 84–93 (cit. on pp. 25, 90).

[RSA78] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems". In: *Commun. ACM* 21.2 (1978), pp. 120–126 (cit. on p. 7).

[SA02] Sergei P. Skorobogatov and Ross J. Anderson. "Optical Fault Induction Attacks". In: *CHES*. 2002, pp. 2–12 (cit. on p. 31).

[Sei05] Jean-Pierre Seifert. "On authenticated computing and RSA-based authentication". In: *ACM Conference on Computer and Communications Security*. 2005, pp. 122–127 (cit. on p. 11).

[Sha49] Claude Elwood Shannon. "Communication Theory of Secrecy Systems". In: *Bell Sys. Tech. J.* 28 (1949), pp. 657–715 (cit. on p. 6).

[SS71] Schönhage and V. Strassen. "Schnelle Multiplikation grosser Zahlen". In: *Computing* 7 (1971) (cit. on p. 92).

[ST] Adi Shamir and Eran Tromer. "Acoustic cryptanalysis. On nosy people and noisy machines". A webpage accessed on 09.08.2011. URL: http://people.csail.mit.edu/tromer/acoustic/ (cit. on p. 10).

[VDG98] Jeroen Van De Graaf. "Towards a formal definition of security for quantum protocols". AAINQ35648. PhD thesis. Canada, 1998. ISBN: 0-612-35648-5 (cit. on p. 65).

[Ven09] Daniele Venturi. *Introduction to Algorithmic Number Theory*. Tech. rep. ECCC TR09-62. SAPIENZA University of Rome, 2009. URL: http://eccc.hpi-web.de/report/2009/062/ (cit. on p. 6).

[Wat05] Brent Waters. "Efficient Identity-Based Encryption Without Random Oracles". In: *EUROCRYPT*. 2005, pp. 114–127 (cit. on pp. 75, 83).