THE UNIVERSITY
of EDINBURGH

Cryptographic Techniques for Hardware Security

Yiannis Tselekounis

Doctor of Philosophy

Laboratory for Foundations of Computer Science

School of Informatics

University of Edinburgh

2018

To my family.

Abstract

Traditionally, cryptographic algorithms are designed under the so-called *black-box model*, which considers adversaries that receive *black-box access* to the *hardware implementation*. Although a "black-box" treatment covers a wide range of attacks, it fails to capture reality adequately, as real-world adversaries can exploit *physical properties* of the implementation, mounting attacks that enable unexpected, *non-black-box access*, to the components of the cryptographic system. This type of attacks is widely known as *physical attacks*, and has proven to be a significant threat to the real-world security of cryptographic systems. The present dissertation is (partially) dealing with the problem of protecting *cryptographic memory* against *physical attacks*, via the use of *non-malleable codes*, which is a notion introduced in a preceding work, aiming to provide privacy of the encoded data, in the presence of adversarial faults.

In the present thesis we improve the current state-of-the-art on non-malleable codes and we provide practical solutions for protecting real-world cryptographic implementations against physical attacks. Our study is primarily focusing on the following adversarial models: (*i*) the extensively studied *split-state* model, which assumes that private memory splits into two parts, and the adversary tampers with each part, independently, and (*ii*) the model of *partial functions*, which is introduced by the current thesis, and models adversaries that access arbitrary subsets of codeword locations, with bounded cardinality. Our study is comprehensive, covering *one-time* and *continuous*, attacks, while for the case of partial functions, we manage to achieve a stronger notion of security, that we call *non-malleability* with *manipulation detection*, that in addition to privacy, it also guarantees *integrity* of the private data. It should be noted that, our techniques are also useful for the problem of establishing, *private*, *keyless communication*, over adversarial communication channels.

Besides physical attacks, another important concern related to cryptographic hard-

ware security, is that the *hardware fabrication process* is *assumed to be trusted*. In reality though, when aiming to minimize the production costs, or whenever access to leading-edge manufacturing facilities is required, the fabrication process requires the involvement of several, potentially malicious, facilities. Consequently, cryptographic hardware is susceptible to the so-called *hardware Trojans*, which are hardware components that are maliciously implanted to the original circuitry, having as a purpose to alter the device's functionality, while remaining undetected. Part of the present dissertation, deals with the problem of protecting cryptographic hardware against *Trojan injection attacks*, by (i) proposing a *formal model* for assessing the security of cryptographic hardware, whose production has been partially outsourced to a set of untrusted, and possibly malicious, manufacturers, and (ii) by proposing a *compiler* that transforms any cryptographic circuit, into another, that can be securely outsourced.

Acknowledgements

In this part, I would like to thank those who I feel played an important role in the process of writing the present thesis.

First of all, I would like to express my gratitude to my supervisor, Aggelos Kiayias, for his guidance and support, and for giving me the freedom to work on various research topics. Our discussions have been invaluable to me, as they cultivated my ability to identify new research directions and establish their importance, boosting my confidence as a researcher. Besides the scientific part, I am also thankful for the great non-scientific moments that we shared over the years; for the trips, the summer schools, and the hospitality.

I would like to thank Feng-Hao Liu, who, besides my supervisor, is the person that I have been closely collaborating with, during my PhD studies, and who introduced me to non-malleable cryptography while he was a post-doc at the University of Maryland. I am thankful for all the technical discussions and the uncountable hours of brainstorming. I also want to thank my coauthors, Elias Koutsoupias and Maria Kyropoulou, for the fruitful discussions on various game-theoretic concepts, and also, Guisepe Ateniese, Bernardo Magri and Daniele Venturi, for our discussions on Trojan resilience. Also, I want to thank Vassilis Dougalis, Elias Koutsoupias, Stavros Toumpis and Stathis Zachos, for inspiring me during my undergraduate and post-graduate studies, and for their support on my very first steps.

I would like to thank the members of the examination committee, Sebastian Faust and Markulf Kohlweiss, for their valuable comments and the insightful discussions, and the non-examining chair Myrto Arapinis, for always being helpful and willing to assist me with all practical concerns related to the process of submitting and examining my PhD thesis, and all the technical and non-technical discussions. I am also thankful to the people who participated to my annual reviews, namely my second supervisor, Kousha Etessami,

Myrto Arapinis, and of course my supervisor, Aggelos Kiayias.

Life would be much harder without having good friends to rely on. I want to thank my good friend, Thanos Tsouanas, for being in my life in a critical moment, inspiring me to choose a path that led to the development of the current thesis. I am grateful to my friend, neighbour and guarantor, Marilena Karanika, for all the good moments and for making my life in Edinburgh more convenient, and also, Charis Chanialidis for the friendship and all the chats the we had during my studies in Edinburgh.

I would like to thank my friends and colleagues Katerina Samari, Thomas Zacharias and Giorgos Panagiotakos, for all the beautiful moments; the trips, the discussions about life, as well as for all that we shared during hard times. Also, I want to thank Tatiana Kyrou for dealing with all sort of bureaucratic issues during my PhD studies, Chrystalla Pavlou, Veselin Blagoev and Alexandru Cojocaru, for the chats on the 5th floor of the Informatics Forum, Nikos Leonardos, and all the members of the Security and Privacy group, the Blockchain Lab and the CryptoSec group, for the interesting discussions.

I could never achieve anything without the support of my beloved family; my mother, Elli, with her endless love, her priceless advice and her discreetness, letting me to choose my own path in life, and my sister, Eleni, who is the most noble and generous person that I know, always being supportive in every step I make. I am thankful for their unconditional love and for always being by my side. I am grateful to my father Dimitris, who, even while not being present, has been a source of inspiration, that to a great extent shaped me into the person I am today.

Finally, I would like to express my gratitude to Sandy Ganoti, who stood by my side during my studies, making beautiful and giving special purpose to my life in Edinburgh. I am grateful for her love, her patience and support.

Thank you!

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified. The ideas presented in the current thesis are mainly based on the following papers:

- A. Kiayias, F.-H. Liu, and Y. Tselekounis. Non-Malleable Codes for Partial Functions with Manipulation Detection. In *Advances in Cryptology – CRYPTO 2018*, pages 577-607 [KLT18a].
- A. Kiayias, F.-H. Liu, and Y. Tselekounis. Practical Non-Malleable Codes from ℓ-more Extractable Hash Functions. In *ACM CCS 16: 23rd Conference on Computer and Communications Security*, pages 1317-1328 [KLT16].
- G. Ateniese, A. Kiayias, B. Magri, Y. Tselekounis and D. Venturi. Secure Outsourcing of Cryptographic Circuits Manufacturing. In *ProvSec 2018: 12th International Conference on Provable Security* [AKM+18].
- A. Kiayias, F.-H. Liu, and Y. Tselekounis. ℓ-more Extractable Hash Functions and Applications to Non-Malleable Cryptography [KLT18b].

Additional publications, dealing with problems *independent* of the ones studied by the present dissertation, are listed below, and they are *not included* in the current text:

- A. Kiayias and Y. Tselekounis. Tamper resilient circuits: The adversary at the gates. In *Advances in Cryptology – ASIACRYPT 2013*, pages 161-180 [KT13].
- A. Kiayias, E. Koutsoupias, M. Kyropoulou and Y. Tselekounis. Blockchain Mining Games. In *EC 2016: Proceedings of the 2016 ACM Conference on Economics and Computation*, pages 365-382 [KKKT16].

(Yiannis Tselekounis)

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The enormous technological advancements of the last decades, led to the emerge of an era in which, almost every aspect of our life has been digitized. While this brings convenience and facilitates innovation, it also poses serious concerns about the *privacy*, as well as the *integrity* and *authenticity* of the users' data. The role of cryptography is to provide rigorous and realistic mathematical models of security, followed by provably secure solutions that protect against malicious behaviours.

The wide use of *cryptographic hardware*, is one of the impacts of the digital era in our lives. By the term "cryptographic hardware" (or "cryptographic device"), we refer to any device that possess some sort of *private memory* (*private state*) and implements specific cryptographic algorithms, whose security depends solely on the privacy of the memory contents. In each invocation, the device evaluates the cryptographic algorithm over the private state and any potential public user input, and returns the output of the computation to the caller. An instance for such a cryptographic system is a *smart-card*. Smart-cards are light-weight hardware devices, widely used by payment systems, enabling the authentication of users, by requiring them to provide proof of knowledge of the secret code (referred to as pin), that is stored in the device's private state.

Although cryptographic devices have revolutionized and improved human lives, at the same time they pose serious security concerns, mainly due to the following reasons: (1) they can easily fall into the hands of malicious users, as they are susceptible to theft, and (2) their implementation requires the involvement of numerous entities, that are assumed to be trusted. In this context, it is possible for a malicious user to alter the functionality of the device, by exploiting *physical properties* of the implementation, in a way that enables recovery of the private state, thus allowing impersonation of legitimate users. Furthermore,

the adversarial advantage could increase, when the adversary is (partially) controlling the *fabrication process*, which is assumed to be trusted.

The present dissertation deals with problem of protecting cryptographic hardware against *physical* and *hardware Trojan injection*, attacks, using cryptographic methods. Along with that, we highlight the applicability of our techniques to the problem of establishing *secure communication* in the presence of *man-in-the-middle adversaries*.

Before presenting the contributions of the thesis, we introduce a formal model, together with basic notions.

1.1 Physical attacks and provable security

Traditionally, cryptographic algorithms are designed assuming adversaries that receive *black-box* access to the cryptographic device, i.e., they are allowed to execute the device on inputs of their choice, and by observing the output behaviour, they try to infer information related to the private state. In this setting, the value of the private memory, as well as the algorithm executed by the device, cannot be altered by the attacker. This mode of interaction is usually modeled as a *security game* that *formally* defines the capabilities of the adversary and the outcomes of the game that signify that security has been breached. After defining the model, the next step is to construct cryptographic schemes and formally prove their security.

Broadly speaking, there are two main types of cryptographic security, namely, *computational* and *information-theoretic*, or *unconditional*. The former type assumes *computationally bounded* adversaries and relies on *intractability assumptions* over problems that have been studied by researchers for decades, and there is strong evidence that such problems do not have *efficient*, i.e., *polynomial-time*, solvers. Under this assumption, the security of the cryptographic scheme is reduced to the underlying *hardness assumption*, by formally proving that, any *efficient* adversary that compromises the security of the cryptographic scheme, yields an efficient solver for the underlying problem, reaching a contradiction. On the other hand, security in the *information-theoretic* setting is with respect to *computationally unbounded* adversaries, and the proof of security is based on *probabilistic arguments*, showing that "bad" events occur with *negligible* probability. In this setting, the proof often relies on assumptions related to the *probabilistic behaviour* of the environment, like for instance, assumptions on the error probability, when studying the problem of data transmission over noisy channels.

Even though the approach described above covers many practical scenarios, it does not model reality adequately, as real-world adversaries can be much more powerful. In reality, the adversary, besides observing the device's input-output behaviour, it can also exploit the *physical properties* of the implementation, mounting attacks that enable unexpected, *non-black-box access*, to the components of the cryptographic system. Consequently, security can be compromised, as such attacks are not covered by the security model. An instance of this type of attacks was considered in [Koc96], in which Kocher presents an attack that recovers the secret key of an encryption scheme, by measuring the power consumed by the device during the encryption operation, while in [KJJ99], Kocher et al. manage the same thing, this time by measuring the time needed for the encryption to complete. In subsequent works [GST17, GST14], Genkin, Shamir and Tromer manage to extract 4096-bit RSA decryption keys within an hour, using the sound generated by the computer during the decryption of some chosen ciphertexts.

Besides physical attacks in the *passive* setting, in which the adversary performs different kinds of measurements, there is also a wide range of *active physical attacks*. The works of [BS97, BDL01, BDL97, GA03, AK96, SA03], consider adversaries that induce faults to the computation, and then show how to leak sensitive information, by inspecting the output of the tampered computation. Physical attacks have proven to be a significant threat to the real-world security of cryptographic implementations, as they can fully break the security of cryptographic systems.

Over the past decade, the cryptographic community has made a substantial progress towards protecting cryptographic implementations from physical attacks. From a theoretical perspective, this line of research focuses on providing rigorous adversarial models, that cover a wide range of real-world attacks, followed by provably secure solutions. The efforts made by the community, led to the development of two important directions in cryptography, establishing the branches that nowadays are known as *leakage-resilient* and *tamper-resilient*, *cryptography*. The former, deals with the problem of protecting cryptographic implementations against *passive attacks*, by considering adversaries that receive *bounded information* (*leakage*) related to the private state of the cryptographic primitive (see for instance [ISW03, MR04, DP08, FKPR10, FRR$^+$10, BKKV10, DDF14, DLZ15, DGL$^+$16]), while *tamper-resilient cryptography* is dealing with *active physical attacks*, considering adversaries that, either tamper with the memory of the cryptographic device, which is a model originally considered in the seminal work of Gennaro et al. [GLM$^+$04], or adversaries that induce faults to the computation; a model originally considered by [IPSW06]

and subsequently by [FPV11, DK12, DK14, KT13].

Part of this dissertation focuses on *securing cryptographic memory* against *tampering* and *leakage* attacks, via the use of *non-malleable codes* [DPW10].

1.1.1 Non-malleable codes and physical security

The notion of *non-malleable codes* (NMC) was introduced by Dziembowski, Pietrzak and Wichs [DPW10], as a relaxation of error correction and error detection codes, aiming to provide strong privacy, without ensuring correctness. Informally, non-malleable encoding schemes are *keyless primitives*, guaranteeing that any modified codeword, decodes, either to the original message, or to a completely unrelated one, with overwhelming probability. The definition of non-malleability is simulation-based, stating that for any tampering function f, there exists a simulator that simulates the tampering effect, by only accessing f, i.e., without making any assumptions on the distribution of the encoded message. Besides the original definition [DPW10], the work of Aggarwal et al. [ADKO15], considers a reduction based definition according to which, security against a function class $\mathcal{F}$, is proven by a reduction from $\mathcal{F}$ to a class of *trivial functions*, consisting of the identity function and all constant functions. As the authors prove, their definition is equivalent to the one given in [DPW10].

The main application of non-malleable codes, that motivated the seminal work of Dziembowski et al. [DPW10], is the protection of cryptographic implementations from *active physical attacks* against memory, known as *tampering attacks*. In this setting, the adversary tampers with the memory contents of the cryptographic device, receives the output of the computation on inputs of its choice, with respect to the tampered memory value, and tries to extract information related to the original memory value. Non-malleable codes provide a straightforward method to protect against such attacks [DPW10] and we briefly discuss it in the paragraph that follows.

Consider a *non-malleable encoding scheme* $(\mathsf{Enc}, \mathsf{Dec})$, consisting of the *encoder*, Enc, and the *decoder*, Dec, and any *reactive cryptographic functionality* G, with private state s, receiving public input m. For instance, consider G to be the computation of a digital signature over the input message m and signing key s. The evaluation of G, over m, s, will be denoted by $[G, s](m)$. Using $(\mathsf{Enc}, \mathsf{Dec})$ we can transform the functionality $[G, s]$ into a tamper-resilient functionality $[\hat{G}, c]$, which is secure against tampering with the private state c. The transformation is as follows: first we compute the non-malleable encoding of s, i.e., we compute $c \leftarrow \mathsf{Enc}(s)$, and then we define the functionality $[\hat{G}, c]$, such that on

input message m, $[\hat{G}, c](m)$, first recovers the signing key by computing $s \leftarrow \mathsf{Dec}(c)$, then executes $[G, s](m)$, i.e., it executes the original functionality $[G, s]$ on input message m, erases the private state c, and stores a fresh non-malleable encoding of s, by recomputing $c \leftarrow \mathsf{Enc}(s)$. In the i-th round of interaction between the adversary and the device, the adversary evaluates $[\hat{G}, c]$ on input m_i of its choice, and in addition, it is allowed to issue a tampering query, f_i, against c, receiving $[G, f_i(c)](m_i)$, i.e., the adversary receives the output of the functionality on input m_i, with respect to the tampered memory value $f_i(c)$. In case, $\mathsf{Dec}(f_i(c)) = \perp$, i.e., if the attacker creates an invalid codeword,[1] the private memory of the device is erased and the functionality outputs $\perp$, in all subsequent rounds (in that case we say that the devices self-destructs). As it is stressed by Gennaro et al. [GLM+04], the self-destruct mechanism is essential, otherwise the adversary can fully recover the private state.

The non-malleability property of $(\mathsf{Enc}, \mathsf{Dec})$, guarantees that in a single round, any admissible modification of the private state, c, decodes, either to the original value s, or to a completely unrelated one, and in addition, the tampering effect can be simulated without accessing s. This implies that, any information the attacker learns by evaluating $[G, f(c)](m_i)$, it could also be computed by either evaluating $[G, c](m_i)$, i.e., without tampering with c, or by evaluating $[G, c'](m_i)$, for a private state value c', that is completely unrelated to s, thus privacy of the private state is guaranteed. Since the non-malleable encoding of s is refreshed after each invocation of the functionality, the above construction is *continuously secure*, i.e., it provides security against multi-round adversaries, even if the underlying non-malleable code is *one-time secure*.

The transformation described above requires erasure of the private state after each invocation of the functionality. However, even if fully erasing the private state is feasible, such erasures can be problematic in the presence of tampering adversaries, as it was originally pointed out by Faust et al. [FMNV14]. Consider a scenario in which besides storing the encoding of a private key, the memory of the device also contains other, unencoded data. In this setting, erasing only the part that stores the encoding of the private key could compromise security, as the adversary might be able to copy the codeword on the unencoded part of the memory, rendering the self-destruct mechanism of the device useless. A solution to the problem would be to erase the entire memory of the device, still in most cases, there is a good chance that the users will use the uncoded part of the memory to store important data that should not be erased. The straightforward way to prevent the

[1] The output of the decoder on invalid input codewords, is defined to be the special symbol "$\perp$".

adversary from getting permanent access to the codeword by making copies of it, would be to encode the entire memory of the device. However, this approach has serious disadvantages, that where originally identified by [FMNV14]. First of all, protecting the entire memory is implausible, as in many cases a major part of it consists of non-sensitive data that don't require protection. Secondly, the proposed solution is highly inefficient, as each invocation of the device requires decoding and re-encoding the entire state, introducing substantial overhead.

Motivated by the above considerations, Faust et al. [FMNV14], introduce the notion of *continuous non-malleable codes* (CNMC) as an extension to the original notion, considering adversaries that tamper *continuously* with the *same* codeword, as opposed to the original notion, that requires *erasures* in order to be applied to the continuous setting.

In Chapter 2, we provide formal definitions for the notions of *non-malleable codes* and *continuous non-malleable codes*, while in Chapters 3,5,6, we construct these notions for various function classes, that we informally introduce below.

1.1.2 Models of memory tampering

Ideally, we would like to achieve non-malleability against arbitrary function classes, however, it not hard to see that this is impossible. For instance, consider $(\mathsf{Enc}, \mathsf{Dec})$, to be the *encoding* and *decoding*, respectively, procedures, of an encoding scheme, and assume that the tampering function class permits the application of $(\mathsf{Enc}, \mathsf{Dec})$. Then, we define the tampering function f as $f(c) := \mathsf{Enc}(\mathsf{Dec}(c) + 1)$, where c denotes the encoding of the private message, s. Clearly, for any codeword c, the decoding of $f(c)$ is equal to $s + 1$, which is a message highly related to the original one, thus no secure construction can exist against any function class that contains f. Therefore, when aiming for non-malleability, restricting the function class is unavoidable.

In the original paper [DPW10], the authors present the first non-malleable code for the class of *bit-wise independent tampering functions*, denoted as $\mathcal{F}_{\mathsf{bit}}$, that tamper with each bit of the codeword, *independently*. In particular, an element $f \in \mathcal{F}_{\mathsf{bit}}$, can be represented as a vector of functions $(f_1, \ldots, f_\nu)$, where ν denotes the codeword length, in bits, and each f_i tampers with the i-th codeword bit, independently of the value of the remaining bits. Besides an *explicit* construction against $\mathcal{F}_{\mathsf{bit}}$, the authors of [DPW10], using probabilistic arguments, they prove the *existence* of non-malleable codes for function classes of *bounded size*, namely for all function classes $\mathcal{F}$, such that $\log(\log(|\mathcal{F}|)) < \nu$, where $|\mathcal{F}|$ denotes the

size of the function class.[2] Although the probabilistic method of [DPW10], does not directly yield efficient explicit constructions, it gives efficient non-malleable codes in the *random oracle model*, for more general classes of tampering functions. The results of [DPW10] are in the *information-theoretic* setting, i.e., they provide security against computationally unbounded adversaries.

Due to their important application, constructing non-malleable codes has received a lot of attention over the last years. The main objective in this line of research, is to construct efficient encoding schemes against function classes, expressive enough to model real-world attacks. As in the case of error correction/detection codes, the efficiency of non-malleable encoding schemes, depends on how efficient the encoding and decoding procedures are, while another important measure of efficiency, is the *information rate* of the scheme, which is defined as the ratio of the message to codeword, length, as the message length goes to infinity. Based on the notation introduced so far, the *information rate* of an encoding scheme $(\mathsf{Enc}, \mathsf{Dec})$, is formally defined as,

$$\lim_{|s|\to\infty} \frac{|s|}{|c|},$$

where c is the codeword output by $\mathsf{Enc}(s)$, and $|s|$ (resp. $|c|$) denotes the message (resp. codeword) length.

The split-state model. Due to the impossibility of non-malleable codes for arbitrary functions classes, various models have been proposed and studied over the years. An important model, that the present thesis is partially focusing on, is the extensively studied *split-state model*. The split-state model is a generalization of the bit-wise independent tampering function class, and was originally introduced in the context of non-malleable codes, by Dziembowski et al. [DPW10] and Liu and Lysyanskaya [LL12], who considered the case of *two split-states*. Briefly speaking, in the split-state model with two states, the codeword (private memory) is split into two parts, c_0, c_1, and the attacker is allowed to apply on it any function $f = (f_0, f_1)$, that results in a tampered codeword equal to $(f_0(c_0), f_1(c_1))$. As in the case of bit-wise independent tampering, the critical point here is that each f_i, for $i \in \{0,1\}$, tampers with c_i, independently of the value c_{1-i}. This is a plausible model to assume, since there are many scenarios in which sensitive data may be split into two storage devices, that are physically separated, for security reasons. In this setting, an adversary that receives tampering access over the memory components,

[2] Note that for the function family $\mathcal{F}_{\mathsf{all}}$, consisting of all functions $f : \{0,1\}^\nu \to \{0,1\}^\nu$, we have $\log(\log(|\mathcal{F}_{\mathsf{all}}|)) = \nu + \log(\nu)$.

modifies the contents of each memory independently of the contents of the other. Note that, the model can generalize to multiple split-states, with the two-state variant being the hardest to achieve. Part of the current thesis (cf. Chapter 5,6), considers the problem of constructing efficient non-malleable codes against split-state attackers, with two states, and from now on, any reference to the split-state model will be with respect to the two-state variant.

The class of partial functions. The current state-of-the-art on non-malleable codes, which is discussed extensively in subsequent chapters, consider adversaries that receive *full access* over the codeword, while imposing *structural* or *computational* restrictions to the way the function computes over the input. In the present thesis (cf. Chapter 3), we initiate a study on the notion of non-malleable codes, for functions that receive *partial access* over the codeword. Informally speaking, the class of *partial functions* contains all functions that read/write on an arbitrary subset of codeword bits,[3] of specific cardinality. As we elaborate later in the thesis, this is a plausible and important, yet overlook class.

In Chapter 3, we construct efficient non-malleable codes for the class of partial functions, while in Chapter 5, we provide efficient constructions against split-state adversaries. In Chapter 6, we consider continuous attacks and we construct continuously secure non-malleable codes for split-state adversaries, while a weaker notion of continuous security against partial functions is also presented in Chapter 3.

1.2 Secure communication

Non-malleable codes can be useful, not only as a countermeasure against physical attacks, but in any setting in which a restriction is imposed to the way the adversary accesses the data. As an example, consider two parties that wish to communicate securely, while given access to two independent and possibly insecure channels. Assuming the adversary tampers independently with the data transmitted over the two channels,[4] split-state non-malleable codes provide a straightforward way for establishing a *private*, *keyless communication channel*, ensuring *non-malleability* of the transmitted data. This property has also been identified by [BDKM18, CGM+16], who present the notion of non-malleable codes in the so-called *streaming model*, in which the adversary is accessing codeword blocks in a streaming

[3] When considering alphabets larger than {0,1}, the function is accessing codeword symbols.

[4] This could happen because the adversary prefers not to slow down the transmission rate, in order to avoid detection.

fashion, i.e., tampering with the i-th block only depends on the view over the previous $i-1$ blocks. The main drawback of existing constructions, is that they do not provide integrity, which we believe is an important property when applying NMCs in the context of secure communication.

The results presented in the current dissertation find useful application, not only in the context of securing cryptographic hardware against memory tampering, but also for providing *private*, *keyless* and *non-malleable* communication, that also guarantees *integrity* of the transmitted data. As we discuss later in the thesis, this type of security can be achieved via the use of *non-malleable codes*, as well as via the use of *non-malleable commitments*, which are studied in Chapter 7 of the present thesis.

1.3 Protection against hardware Trojans

Up to now, and as far as hardware security is concerned, we have considered adversaries that receive *tampering access* over the cryptographic implementation, assuming that the *fabrication process* has been performed by a trusted party. In reality, though, when aiming to minimize the production costs, or whenever access to leading-edge manufacturing facilities is required, the fabrication process demands the involvement of several, potentially malicious, facilities. Consequently, integrated circuits (ICs) are susceptible to the so-called hardware Trojans, which are hardware components that are maliciously implanted to the original circuitry, having as a purpose to alter its functionality, while remaining undetected. Hardware Trojans are aiming at disabling or compromising the security defences of a system, or covertly leaking information related to the systems' private state [LKG+09, CNB09, BRPB14]. The injection of Hardware Trojans can occur during the design phase, by a malicious designer, or during the manufacturing phase, by a malicious fabrication facility. Once the Trojan is implanted, it may be active the entire time, or it may be triggered by some special event, e.g., when the user supplies the circuit with some special input, or after a specific number of circuit invocations.

In Chapter 8 of the thesis, we deal with the problem of *secure circuit outsourcing*, by providing a rigorous security model that covers a large class of hardware Trojans, together with a construction that is based on the notion of *secure multi-party computation*.

1.4 Contributions

In the current section, we briefly summarize the outcomes of the present dissertation.

1.4.1 Non-malleable codes for partial functions and applications

In the present thesis, we initiate a comprehensive study on *non-malleable codes* for the class of *partial functions*, that read/write on an arbitrary subset of codeword bits (symbols), with specific cardinality. Our constructions are efficient in terms of *information rate*, while allowing the attacker to access asymptotically *almost the entire codeword*. In addition, they satisfy a notion of security which is stronger than non-malleability, that we call *non-malleability* with *manipulation detection* (MD-NMC), guaranteeing that any modified codeword decodes, either to the original message, or to $\perp$, with overwhelming probability. Our results are informally summarized bellow.

1. (**MD-NMC in the CRS model**): First, we construct an information rate 1 MD-NMC, in the *common reference string* (CRS) model,[5] tolerating adversarial access over a $1-1/\Omega(\log k)$ faction of codeword bits, where k denotes the security parameter. The proposed construction combines Authenticated Encryption together with an inner code that protects the key of the encryption scheme.

2. (**MD-NMC in the standard model**): In our second result we show how to remove the CRS, and we present a computationally secure MD-NMC in the standard model, over alphabets of length $O(\log k)$, with information rate $1 - 1/\Omega(\log k)$, tolerating adversarial access over a $1 - 1/\Omega(\log k)$ fraction of codeword blocks (or symbols).

3. (**Continuous MD-NMC**): In our final result, we construct *continuous non-malleable codes*, for a weaker notion of security than the one presented in [FMNV14], since our codewords need to be updated after each tampering query. Nevertheless, our update operation uses only shuffling and refreshing operations, avoiding the full re-encoding process. Also, by incorporating the seed of a pseudo-random generator inside the codeword, the final construction is deterministic.

In Chapter 3, we propose various applications of the primitive in tamper-resilient cryptography, namely, for protecting the cryptographic memory of boolean and arithmetic circuits against tampering attacks, and also for protecting data transmission over adversarial channels.

[5]The common reference string (CRS) model, assumes that all parties receive access to a trusted, honestly and correctly, generated string, and generalizes the common random string model, in which the common string is sampled according to the uniform distribution.

NMC for partial functions and All-or-Nothing Transforms. Besides the importance of non-malleable codes for partial functions in the active setting, in which the function is allowed to partially *read/write* the codeword, the passive analogue of the primitive, i.e., when the function is only given *read access* over the codeword, matches the model considered by All-Or-Nothing Transforms (AONTs), which is a notion originally introduced by Rivest [Riv97], providing security guarantees similar to those of leakage resilience: reading an arbitrary subset (up to some bounded cardinality) of locations of the codeword does not reveal the underlying message. As non-malleable codes provide privacy, non-malleability for partial functions is the active analogue of (and in fact implies) AONTs, thus our constructions yield efficient AONTs under standard assumptions (only one-way functions), which, was an open question until now.

1.4.2 ℓ-more weakly extractable hash function families and non-malleability

In the present dissertation, we introduce the notion of *ℓ-more extractable, collision resistant, hash function families*. Briefly speaking, ℓ-more extractable hash function families capture the idea that, if an adversary, that is given $\ell \in \mathbb{N}$ precomputed hash values $v_1, \ldots, v_\ell$, manages to produce a new valid hash value $\tilde{v}$, then it must know a pre-image of $\tilde{v}$. This is a generalization of the notion of extractable hash functions by Bitansky et al. [BCCT12] and Goldwasser et al. [GLR11], which corresponds to the $\ell = 0$ case, in which the adversary gets no access to any valid hash values, prior to producing its own value. By appropriately relaxing the extractability requirement without hurting the applicability of the primitive, we manage to instantiate ℓ-more extractable hash functions under the *same* assumptions used by Bitansky et al. and Goldwasser et al., i.e., a variant of the *Knowledge of Exponent Assumption*. We call the resulting notion, *ℓ-more weakly extractable, collision resistant hash function family* (wECRH), and we present the following results.

1. (**Extractability [BCCT12]** $\not\Rightarrow$ **1-more extractability**): We prove that our generalization is strict, by showing that the extractable hash function family of [BCCT12, GLR11], is not 1-more extractable.

2. (**Constructing ℓ-more wECRH**): In our second result, we construct the notion of ℓ-more wECRH. In particular, we provide two instantiations of the primitive. The first one is based on the *hardness of the discrete logarithm problem* and the variant

of the *knowledge of exponent assumption* used by [BCCT12], while the latter is in the random oracle model.

3. (**Leakage-resilient ℓ-more wECRH**): Finally, as an extension of the notion of ℓ-more wECRH, we formalize and construct the notion of *leakage-resilient, ℓ-more, weakly extractable hash function families*, considering attackers that, in addition to receiving access to ℓ precomputed hash values, they also receive *bounded leakage* over the randomness used to compute those values.

Subsequently, we illustrate the power of the primitive by providing constructions that significantly improve the current state-of-the-art in non-malleable cryptography. In particular, we provide the following results based on ℓ-more wECRH.

1. (**NMC for split-state functions**): We significantly improve the efficiency of non-malleable codes in the split-state model, by constructing an encoding scheme with codewords of length $|s| + O(k)$, where $|s|$ is the length of the message, and k is the security parameter. This is a substantial improvement over previous constructions, both asymptotically and concretely.

2. (**Continuous NMC against split-state functions**): We leverage the power of ℓ-more wECRH in the continuous setting, and assuming leakage-resilient ℓ-more extractable hash functions, we construct efficient, continuously non-malleable, leakage-resilient codes against split-state attackers, improving the efficiency of [FMNV14].

3. (**Non-malleable commitments**): Finally, we prove that ℓ-more extractable hash functions imply *succinct, non-interactive, non-malleable commitments*, that satisfy a stronger definition than the ones by Crescenzo et al. [DIO98], and Pass and Rosen [PR05], in the sense that the simulator does not require access to the original message, while the attacker's auxiliary input is allowed to depend on it.

1.4.3 Circuit outsourcing and Trojan resilience

In Chapter 8 of the thesis, we propose a formal model for assessing the security of integrated circuits, whose fabrication has been outsourced to a set of untrusted and possibly malicious, manufacturers. The model that we propose assumes that the circuit specification and design, are trusted, but the fabrication facilities may be malicious. In this setting we provide the following, informally stated results.

1. (**A formal model for secure circuit outsourcing**): We provide a simulation-based definition for the problem of secure circuit outsourcing, ensuring that no information over circuit's private information will be released, even in the presence of Trojans. Our model provides security against powerful adversaries that are allowed to make arbitrary modifications over the outsourced circuit components.

2. (**A complier for secure circuit outsourcing**): We propose a compiler that transforms any cryptographic circuit, into another that can be securely outsourced. Our compiler, relies on *secure multi-party computation* (MPC) with certain suitable properties, that are attainable by existing schemes.

1.5 Outline

The outline of the thesis is as follows. In Chapter 2, we present basic definitions that will be used throughout the thesis. The notions introduced by the current thesis are presented in the corresponding chapters. Chapter 3 introduces the notion of *non-malleable codes* with *manipulation detection* and provides efficient constructions for the class of *partial functions*. In the same chapter, we also present and construct a weaker notion of *continuous non-malleable codes* [FMNV14]. In Chapter 4, we introduce the notion of *l-more weakly extractable, collision-resistant hash function families* (wECRH), and we provide two instantiations of the primitive. In subsequent chapters, i.e., Chapters 5,6,7, we demonstrate the applicability of the primitive by providing a variety of applications in non-malleable cryptography. In particular, in Chapter 5, we show how how to construct practically efficient non-malleable codes against split-state functions, while in Chapter 6 we show how to improve the efficiency of continuous non-malleable codes for split-state adversaries. In Chapter 7 we present another important application of wECRH, by showing how to construct *efficient, succinct, non-interactive non-malleable commitments*. Finally, in Chapter 8 we deal with the problem of *secure circuit outsourcing*, providing a formal model and a feasibility result based on *multi-party computation*.

Chapter 2

Preliminaries

In this Chapter we present the notation and basic definitions and that will be used throughout the thesis. Newly introduced notions will be presented in subsequent chapters.

2.1 Notation and basic notions

Definition 2.1.1 (Notation)**.** *Let t, i, j, be non-negative integers. Then, $[t]$ denotes the set $\{1,\ldots,t\}$. For vectors $\mathbf{x},\mathbf{y}$, $\langle \mathbf{x},\mathbf{y}\rangle$ denotes the inner product of $\mathbf{x}$, $\mathbf{y}$, and $[\mathbf{x}]_i$ is the i-th coordinate of $\mathbf{x}$. For bit-strings x, y, $x||y$, is the concatenation of x, y, $|x|$ denotes the length of x, for $i \in [|x|]$, $x[i]$ is the i-th bit of x, $||_{j=1}^{t} x_j := x_1||\ldots||x_t$, and for $i \leq j$, $x[i:j] := x[i]||\ldots||x[j]$. For a set I, $|I|$, $\mathcal{P}(I)$, are the cardinality and power set of I, respectively, and for $I \subseteq [|x|]$, $x_{|_I}$ is the projection of the bits of x with respect to I. For a string variable c and value v, $c \leftarrow v$ denotes the assignment of v to c, and $c[I] \leftarrow v$, denotes an assignment such that $c_{|_I}$ equals v.*

For a distribution D over a set $\mathcal{X}$, $x \leftarrow D$, denotes sampling an element $x \in \mathcal{X}$, according to D, $x \leftarrow \mathcal{X}$ denotes sampling a uniform element x from $\mathcal{X}$, $U_{\mathcal{X}}$ denotes the uniform distribution over $\mathcal{X}$ and $x_1,\ldots,x_t \xleftarrow{\mathsf{rs}} \mathcal{X}$ denotes sampling a uniform subset of $\mathcal{X}$ with t distinct elements, using rejection sampling. The statistical distance between two random variables X, Y, with range $\mathcal{D}$, is denoted by $\Delta(X,Y)$, i.e., $\Delta(X,Y) := \frac{1}{2}\sum_{u\in\mathcal{D}} |\Pr[X = u] - \Pr[Y = u]|$. In addition, "$\approx$" and "$\approx_c$", denote statistical and computational indistinguishability, respectively, and $\mathsf{negl}(k)$ denotes an unspecified, negligible function, in k. For random variables X, Y, $H_\infty(X)$ and $\tilde{H}_\infty(X|Y)$, denote the min-entropy and average min-entropy conditioned on Y, of X, respectively. For any element g and vector $\mathbf{r} = (r_1,\ldots,r_t)$, $g^{\mathbf{r}} := (g^{r_1},\ldots,g^{r_t})$.

*For a randomized algorithm $\mathcal{A}$, using $y \leftarrow A(x)$ we denote the execution of $\mathcal{A}$ on input x, receiving output y. In this setting, y is a random variable and $\mathcal{A}(x;r)$ denotes the execution of $\mathcal{A}$ on input x with randomness r. An algorithm $\mathcal{A}$ is probabilistic polynomial-time (*PPT*) if $\mathcal{A}$ is randomized and for any $x, r \in \{0,1\}^*$, the computation of $\mathcal{A}(x;r)$ terminates in at most* $\mathrm{poly}(|x| + |r|)$ *steps.*

Given two ensembles of random variables $X = \{X_k\}_{k\in\mathbb{N}}$ and $Y = \{Y_k\}_{k\in\mathbb{N}}$, we write $X \equiv Y$ to denote that the two ensembles are identically distributed, $X \approx Y$ to denote that the two ensembles are statistically close, i.e., $\Delta(X_k, Y_k) \leq \mathsf{negl}(k)$, and $X \approx_c Y$ to denote that the two ensembles are computationally indistinguishable, i.e., for all PPT *distinguishers D, $|\Pr[D(X_k) = 1] - \Pr[D(Y_k) = 1]| \leq \mathsf{negl}(k)$.*

Next we provide the definition of *one-time message authentication code* (MAC) following [KL14].

Definition 2.1.2 (One-time MAC [KL14])**.** *Let k be the security parameter. A message authentication code $\Pi = (\mathsf{Gen}, \mathsf{Mac}, \mathsf{Vrfy})$ is one-time ϵ-secure, if for all algorithms $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$,*

$$\Pr[\mathsf{Mac} - \mathsf{forge}_{\mathcal{A},\Pi}(k) = 1] \leq \epsilon,$$

where,

$\mathsf{Mac} - \mathsf{forge}_{\mathcal{A},\Pi}(k)$:
$sk \leftarrow \mathsf{Gen}(1^k)$
$(s, st) \leftarrow \mathcal{A}_1(1^k)$
$t \leftarrow \mathsf{Mac}_{sk}(s)$
$(\tilde{s}, \tilde{t}) \leftarrow \mathcal{A}_2(t, st)$
Output 1 if $\mathsf{Vrfy}_{sk}(\tilde{s}, \tilde{t}) = 1$ *and* $\tilde{s} \neq s$.

The definition of *leakage-resilient one-time* MAC follows.

Definition 2.1.3 (One-time MAC against leakage)**.** *Let k be the security parameter and $\mathcal{L}$ be a function class. A message authentication code $\Pi = (\mathsf{Gen}, \mathsf{Mac}, \mathsf{Vrfy})$ is one-time ϵ-secure against $\mathcal{L}$ if for all algorithms $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3)$,*

$$\Pr\left[\mathsf{LRMac} - \mathsf{forge}_{\mathcal{A},\Pi}(k) = 1\right] \leq \epsilon,$$

where,

$$\begin{aligned}
&\mathsf{LRMac}-\mathsf{forge}_{\mathcal{A},\Pi}(k):\\
&sk \leftarrow \mathsf{Gen}(1^k)\\
&g \leftarrow \mathcal{A}_1(1^k), g \in \mathcal{L}\\
&(s, st) \leftarrow \mathcal{A}_2(1^k, g(sk))\\
&t \leftarrow \mathsf{Mac}_{sk}(s)\\
&(\tilde{s}, \tilde{t}) \leftarrow \mathcal{A}_3(t, st)\\
&\textit{Output 1 if } \mathsf{Vrfy}_{sk}(\tilde{s}, \tilde{t}) = 1 \textit{ and } \tilde{s} \neq s.
\end{aligned}$$

The above definitions will be used for building authenticated encryption, which is defined in Definition 2.1.6.

Below, we provide formal definitions for the notions of *collision resistant hash function families* and the *hardness of the discrete logarithm problem* (DLOG).

Definition 2.1.4 (Collision resistant hash function family [KL14])**.** *A fixed length, collision resistant, hash function family, is a pair of probabilistic algorithms* $\mathcal{H}_k = (\mathsf{Gen}, h)$ *satisfying the following:*

- Gen *is a* PPT *algorithm which receives as input a security parameter* 1^k *and outputs a key* z*.*
- h *receives* z*, and* $x \in \{0,1\}^{p_1(k)}$ *and outputs* $h_z(x) \in \{0,1\}^{p_2(k)}$*, where* $p_1(k), p_2(k) = \mathrm{poly}(k)$*,* $p_2(k) < p_1(k)$*.*
- *For all* PPT*adversaries* $\mathcal{A}$*, the collision-finding experiment* $\mathsf{Hcoll}_{\mathcal{A},\mathcal{H}_k}$*, which is defined bellow, satisfies the following property:*

$$\Pr[\mathsf{Hcoll}_{\mathcal{A},\mathcal{H}_k}(1^k) = 1] \leq \mathsf{negl}(k),$$

for some negligible function $\mathsf{negl}(\cdot)$*, where*

$\mathsf{Hcoll}_{\mathcal{A},\mathcal{H}_k}(k)$*:*

  - *A key* z *is generated by executing* $\mathsf{Gen}(1^k)$*.*
  - $\mathcal{A}$ *is given* z *and outputs* x*,*x'*.*
  - *If* $x \neq x'$ *and* $h_z(x) = h_z(x')$ *output 1, otherwise output 0.*

For simplicity, the key of the hash function, z, will be omitted from the notation.

Definition 2.1.5 (Hardness of the discrete logarithm problem [KL14])**.** *For any $k \in \mathbb{N}$ and any group-generation algorithm $\mathcal{G}$, we say that the discrete logarithm problem is hard relative to $\mathcal{G}$, if for all* PPT *algorithms $\mathcal{A}$ there exists a negligible function* negl *such that*

$$\Pr\left[\mathsf{DLog}_{\mathcal{A},\mathcal{G}}(k) = 1\right] \leq \mathsf{negl}(k),$$

where,

$$\begin{aligned}&\mathsf{DLog}_{\mathcal{A},\mathcal{G}}(k):\\ &(\mathbb{G}, g, p) \leftarrow \mathcal{G}(1^k),\ |\mathbb{G}| = p\\ &s' \leftarrow \mathbb{Z}_p, w := g^{s'}\\ &s \leftarrow \mathcal{A}(\mathbb{G}, g, p, w)\end{aligned}$$

$$\textit{If } g^s = w,\ \textit{return } 1,\ \textit{otherwise, return } 0$$

and $\mathbb{G}$ is the description of a cyclic group for which the group operation if efficiently computed.

Below, we define the notion of *authenticated encryption* with *ciphertext indistinguishability under chosen-plaintext attacks* (IND-CPA), for a single oracle query and against *one-time leakage*.

Definition 2.1.6 (1-IND-CPA secure authenticated encryption against one-time leakage))**.** *Let k be the security parameter, let $(\mathsf{KGen}, \mathsf{E}, \mathsf{D})$ be a symmetric encryption scheme and let $\mathcal{L}$ be a set of functions. Then, $(\mathsf{KGen}, \mathsf{E}, \mathsf{D})$ is authenticated, 1-*IND-CPA *secure against one-time leakage with respect to $\mathcal{L}$, if it satisfies the following properties:*

1. **(Correctness)***: For every message s, $\Pr[\mathsf{D}_{sk}(\mathsf{E}_{sk}(s)) = s] = 1$, where $sk \leftarrow \mathsf{KGen}(1^k)$.*
2. **(1-IND-CPA security under leakage)***: for any $g \in \mathcal{L}$ and for any triple of messages s, s_0, s_1, $\Big(\mathsf{E}_{sk}(s), \mathsf{E}_{sk}(s_0), g(sk)\Big) \approx \Big(\mathsf{E}_{sk}(s), \mathsf{E}_{sk}(s_1), g(sk)\Big)$, where $sk \leftarrow \mathsf{KGen}(1^k)$.*
3. **(Unforgeability under leakage)***: For any $g \in \mathcal{L}$ and every algorithm $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$,*

$$\Pr\left[\tilde{e} \neq e \wedge \mathsf{D}_{sk}(\tilde{e}) \neq \perp \;\middle|\; \begin{array}{c} sk \leftarrow \mathsf{KGen}(1^k); (s, st) \leftarrow \mathcal{A}_1(1^k, g(sk));\\ e \leftarrow \mathsf{E}_{sk}(s); \tilde{e} \leftarrow \mathcal{A}_2(e, st)\end{array}\right] \leq \mathsf{negl}(k).$$

When the scheme is computationally secure, we consider computational indistinguishability instead of statistical, and $\mathcal{A}$ is PPT.

Here, it should be noted that the leakage function is being defined by the attacker before receiving the challenge ciphertext, otherwise security breaks. Also, when we don't consider oracle, or leakage, queries, we require that $\mathsf{E}_{sk}(s_0) \approx \mathsf{E}_{sk}(s_1)$, for any pair of messages s_0, s_1. In this setting, the encryption scheme is simply called *semantically secure*. Finally, when considering unforgeability without leakage, $\mathcal{A}_1$ is only receiving the security parameter.

Next we state the t-variant, due to [BCCT12], of the *Knowledge of Exponent Assumption* (KEA), [Dam92, HT98, BP04], with individual auxiliary inputs for the adversary and the extractor, which is known not to contradict the impossibility results of [BCPR14, BP15].

Assumption 2.1.7 (t-KEA [BCCT12])**.** *Let $t, k \in \mathbb{N}$. There exists a group generation algorithm $\mathcal{G}$, such that for any pair $(\mathbb{G}, g)$ sampled according to $\mathcal{G}(1^k)$, where $\mathbb{G}$ is a group of prime order $p \in (2^{k-1}, 2^k)$, the following holds: for any* PPT *algorithm $\mathcal{A}$ with auxiliary input $z_v \in \{0,1\}^{\mathrm{poly}(k)}$, there exist* PPT *extractor $\mathcal{E}_{\mathcal{A}}$ with auxiliary input $z_{\mathcal{E}} \in \{0,1\}^{\mathrm{poly}(k)}$, such that for all sufficiently large $k \in \mathbb{N}$,*

$$\Pr_{\substack{(\mathbb{G},g)\leftarrow\mathcal{G}(1^k)\\(a,\mathbf{r})\leftarrow\mathbb{Z}_p\times\mathbb{Z}_p^t}}\left[\begin{array}{c}(v,v')\leftarrow\mathcal{A}(g^{\mathbf{r}},g^{a\mathbf{r}},z_v), v'=v^a:\\ \mathbf{x}\leftarrow\mathcal{E}_{\mathcal{A}}(g^{\mathbf{r}},g^{a\mathbf{r}},z_{\mathcal{E}})\wedge g^{\langle\mathbf{r},\mathbf{x}\rangle}\neq v\end{array}\right]\leq \mathsf{negl}(k).$$

Next we recall the definition of extractable hash of [BCCT12]. The definition can be modified to support different auxiliary inputs for the adversary and the extractor, as the t-KEA above, but here we present the original version.

Definition 2.1.8 (Extractable hash [BCCT12])**.** *For $k \in \mathbb{N}$, an efficiently samplable hash function ensemble $\mathcal{H} = \{\mathcal{H}_k\}_{k\in\mathbb{N}}$ is extractable, if for any* PPT *algorithm $\mathcal{A}$, there exists a* PPT*extractor $\mathcal{E}_{\mathcal{A}}^{\mathcal{H}}$, such that for all large $k \in \mathbb{N}$ and any auxiliary input $z \in \{0,1\}^{\mathrm{poly}(k)}$:*

$$\Pr_{h\leftarrow\mathcal{H}_k}\left[\begin{array}{c}y\leftarrow\mathcal{A}(h,z), \exists x: h(x)=y:\\ x'\leftarrow\mathcal{E}_{\mathcal{A}}^{\mathcal{H}}(h,z)\wedge h(x')\neq y\end{array}\right]\leq \mathsf{negl}(k).$$

2.1.1 Randomness extractors and universal hash function families

Using extractors [NZ93] we can extract randomness from sources that produce weakly-random values, assuming those values have sufficient min-entropy. Here, we follow the definition given by [DRS04], that uses average conditional min-entropy $\tilde{H}_\infty(\cdot)$.

Definition 2.1.9 (Randomness Extractor [DRS04])**.** *A polynomially time computable function* $\mathsf{Ext} : \mathcal{M} \times \{0,1\}^n \to \{0,1\}^k$ *is an average case, strong,* (m, ϵ)*-extractor, if for all random variables* S*,* Z*, where* S *is a variable over* $\mathcal{M}$ *and* $\tilde{H}_\infty(S|Z) \geq m$*, it holds that*

$$\Delta(\mathsf{Ext}(S; R),\ U_k \mid (R, Z)) \leq \epsilon,$$

where R *denotes the random coins of* Ext*. The value* $L = m - k$ *is called the entropy loss of* Ext*, and* n *is the seed length of* Ext*.*

Universal hash functions are good randomness extractors, and they are defined as follows:

Definition 2.1.10 (ρ-Universal Hashing [CW77])**.** *A family* $\mathcal{H}$ *of deterministic functions* $h : \mathcal{M} \to \{0,1\}^k$ *is called a* ρ*-universal hash family, if for any* $s_1 \neq s_2 \in \mathcal{M}$*,* $\Pr_{h \leftarrow \mathcal{H}}[h(s_1) = h(s_2)] \leq \rho$*. If* $\rho = 1/2^k$*,* $\mathcal{H}$ *is called universal.*

Now we state the leftover-hash lemma [HILL99], following the definition given in [BDK+11].

Lemma 2.1.11 (Leftover-Hash Lemma [HILL99, BDK+11])**.** *Assume that the family* $\mathcal{H}$ *of functions* $h : \mathcal{M} \to \{0,1\}^k$ *is a* $\frac{1+\gamma}{2^k}$*-universal hash family. Then, the extractor* $\mathsf{Ext}(s; h) = h(s)$*, where* h *is sampled according to* $\mathcal{H}$*, is an average case, strong* (m, ϵ)*-extractor, where* $\epsilon = \frac{1}{2} \cdot \sqrt{\gamma + \frac{1}{2^L}}$ *and* $L = m - k$ *is the entropy loss.*

Below, we define the inner product hash function family and in Lemma 2.1.13 we prove that it is universal.

Definition 2.1.12 (The inner product hash function family)**.** *Let* $\mathbb{F}_p$ *be a finite field of prime order* p*, where* p *is a* k*-bit prime number. For any* $t \in \mathbb{N}$*, the inner-product function family* $\mathcal{H}_{\mathsf{ip}} = (\mathsf{Gen}, h)$*, for messages over* $\mathbb{F}_p^t$ *is defined as follows:*

- $\mathsf{Gen}(1^k)$*: sample* $(r_1, \ldots, r_t) \leftarrow \mathbb{F}_p^t$ *and set* $z = (r_1, \ldots, r_t)$*.*
- ***Hash computation****: on input message* $\mathbf{s} = (s_1, \ldots, s_t) \in \mathbb{F}_p^t$*, compute* $h_z(\mathbf{s}) = \sum_{i=1}^t s_i \cdot r_i$*, where the summation refers to the addition operation, and* $\cdot$ *is the multiplication operation, over* $\mathbb{F}_p$*.*

Lemma 2.1.13. *The function family $\mathcal{H}_{\mathsf{ip}}$ of Definition 2.1.12 is universal.*

Proof. For any k in $\mathbb{N}$, let $\mathbb{F}_p$ be any field of order p, where p is a k-bit integer, and let $\mathbf{s} = (s_1, \ldots, s_t)$, $\bar{\mathbf{s}} = (\bar{s}_1, \ldots, \bar{s}_t)$ be two distinct messages, i.e., $\mathbf{s}$ and $\bar{\mathbf{s}}$ differ in at least one coordinate. Without loss of generality, we assume that $s_1 \neq \bar{s}_1$. Then,

$$\Pr_{h_z \leftarrow \mathcal{H}_{\mathsf{ip}}}[h_z(\mathbf{s}) = h_z(\bar{\mathbf{s}})] = \Pr\left[\sum_{i=1}^{t} r_i \cdot (s_i - \bar{s}_i) = 0\right] = \Pr\left[r_1 = \frac{-\sum_{i=2}^{t} r_i \cdot (s_i - \bar{s}_i)}{(s_1 - \bar{s}_1)^{-1}}\right]$$

Hence, for any choice of $r_2, \ldots, r_t$, there is a unique r_1 for which $h_z(\mathbf{s}) = h_z(\bar{\mathbf{s}})$. Since r_1 is random over $\mathbb{F}_p$, we have that $\Pr[h_z(\mathbf{s}) = h_z(\bar{\mathbf{s}})] \leq 1/p \leq 1/2^k$. □

2.2 Tampering function classes

In this section we formally define the tampering function classes that will be considered in the present thesis. We begin by defining the class of *partial functions* that are the main subject of Chapter 3.

Definition 2.2.1 (The class of partial functions $\mathcal{F}_{\Gamma}^{\alpha\nu}$ (or $\mathcal{F}^{\alpha}$))**.** *Let $\Gamma \subseteq \{0,1\}^*$ be an alphabet, $\alpha \in [0,1)$ and $\nu \in \mathbb{N}$. Any $f \in \mathcal{F}_{\Gamma}^{\alpha\nu}$, $f : \Gamma^{\nu} \to \Gamma^{\nu}$, is indexed by a set $I \subseteq [\nu]$, $|I| \leq \alpha\nu$, and a function $f' : \Gamma^{\alpha\nu} \to \Gamma^{\alpha\nu}$, such that for any $x \in \Gamma^{\nu}$, $(f(x))_{|_I} = f'\left(x_{|_I}\right)$ and $(f(x))_{|_{I^{\mathsf{c}}}} = x_{|_{I^{\mathsf{c}}}}$, where $I^{\mathsf{c}} := [\nu] \backslash I$.*

For simplicity, in the rest of the text we will use the notation $f(x)$ and $f(x_{|_I})$ (instead of $f'\left(x_{|_I}\right)$). Also, the length of the codeword, ν, according to Γ, will be omitted from the notation and whenever Γ is omitted we assume that $\Gamma = \{0,1\}$. In Chapter 3.3, we consider $\Gamma = \{0,1\}$, while in Chapter 3.4, $\Gamma = \{0,1\}^{O(\log k)}$, i.e., the tampering function operates over blocks of size $O(\log k)$. When considering the CRS model, the functions are parameterized by the common reference string.

Below, we define the class of affine functions that will be used in Sections 4.3.1 and 4.3.2.

Definition 2.2.2 (The function family $\mathcal{F}_{\mathsf{aff}}$)**.** *For any field $\mathbb{F}$ and any positive integer t, we define the affine function class over $\mathbb{F}^t$, as follows*

$$\mathcal{F}_{\mathsf{aff}} = \left\{ f(x_1, \ldots, x_t) = (d \cdot x_1 + b_1, \ldots, d \cdot x_t + b_t) \mid (b_1, \ldots, b_t) \in \mathbb{F}^t, d \in \mathbb{F} \right\}.$$

Here, the pair $(+, \cdot)$ denotes the standard addition and multiplication operations over $\mathbb{F}$, and any element in $\mathcal{F}_{\mathsf{aff}}$ will be denoted by $(\mathbf{b}, d)$.

Below, we define the split-state functions class, $\mathcal{F}_{\mathsf{ss}}$, which is the main subject of Chapter 5.

Definition 2.2.3 (The split-state function family $\mathcal{F}_{\mathsf{ss}}$)**.** *For any, even, $\nu \in \mathbb{N}$ and any efficiently computable function $f : \{0,1\}^{\nu} \to \{0,1\}^{\nu}$, $f \in \mathcal{F}_{\mathsf{ss}}$, if there exist efficiently computable functions $f_0 : \{0,1\}^{\nu/2} \to \{0,1\}^{\nu/2}$, $f_1 : \{0,1\}^{\nu/2} \to \{0,1\}^{\nu/2}$, such that for every x_0, $x_1 \in \{0,1\}^{\nu/2} \times \{0,1\}^{\nu/2}$, $f(x_0||x_1) = f_0(x_0)||\ f_1(x_1)$.*

In the subsequent chapters, we will slightly abuse notation, allowing split-state functions to operate over pairs of bitstrings, i.e., for $f = (f_0, f_1) \in \mathcal{F}_{\mathsf{ss}}$ and x_0, $x_1 \in \{0,1\}^{\nu/2} \times \{0,1\}^{\nu/2}$, we will use the notation $f(x_0, x_1) = (f_0(x_0), f_1(x_1))$.

2.3 Non-malleable codes

Below, we define encoding schemes, based on the definitions by [DPW10, LL12].

Definition 2.3.1 (Encoding scheme [DPW10])**.** *A (κ, ν)-coding scheme, $\kappa, \nu \in \mathbb{N}$, is a pair of algorithms $(\mathsf{Enc}, \mathsf{Dec})$ such that: $\mathsf{Enc} : \{0,1\}^{\kappa} \to \{0,1\}^{\nu}$ is an encoding algorithm, $\mathsf{Dec} : \{0,1\}^{\nu} \to \{0,1\}^{\kappa} \cup \{\perp\}$ is a decoding algorithm, and for every $s \in \{0,1\}^{\kappa}$, $\Pr\left[\mathsf{Dec}(1^k, \mathsf{Enc}(1^k, s)) = s\right] = 1$, where the probability runs over the randomness used by $(\mathsf{Enc}, \mathsf{Dec})$.*

For encoding schemes in the CRS model the definition is as follows.

Definition 2.3.2 (Encoding scheme in the Common Reference String (CRS) Model [LL12]))**.** *A (κ, ν)-coding scheme in the* CRS *model, $\kappa, \nu \in \mathbb{N}$, is a triple of algorithms $(\mathsf{Init}, \mathsf{Enc}, \mathsf{Dec})$ such that: Init is a randomized algorithm which receives 1^k, where k denotes the security parameter, and produces a common reference string $\Sigma \in \{0,1\}^{\mathrm{poly}(k)}$, and $(\mathsf{Enc}(1^k, \Sigma, \cdot), \mathsf{Dec}(1^k, \Sigma, \cdot))$ is a (κ, ν)-coding scheme, $\kappa, \nu = \mathrm{poly}(k)$.*

For brevity, 1^k will be omitted from the inputs of Init, Enc and Dec. Also, we can easily generalize the above definitions with respect to larger alphabets, i.e., by considering $\mathsf{Enc} : \{0,1\}^{\kappa} \to \Gamma^{\nu}$ and $\mathsf{Dec} : \Gamma^{\nu} \to \{0,1\}^{\kappa} \cup \{\perp\}$, for some alphabet $\Gamma \subseteq \{0,1\}^*$.

Below we state the definition of strong non-malleability in the CRS model based on the definitions by [DPW10, LL12].

Definition 2.3.3 (Strong non-malleability in the CRS model [DPW10, LL12])**.**
Let $(\mathsf{Init}, \mathsf{Enc}, \mathsf{Dec})$ be a (κ, ν)-encoding scheme in the common reference string model, and

$\mathcal{F}$ be a family of functions $f : \{0,1\}^\nu \to \{0,1\}^\nu$. For any $f \in \mathcal{F}$, $s \in \{0,1\}^\kappa$, sample $\Sigma \leftarrow \mathsf{Init}(1^k)$ and define the tampering experiment

$$\mathsf{Tamper}_s^{\Sigma,f} := \left\{ \begin{array}{l} c \leftarrow \mathsf{Enc}(\Sigma, s), \tilde{c} \leftarrow f^\Sigma(c), \tilde{s} \leftarrow \mathsf{Dec}(\Sigma, \tilde{c}) \\ \textit{Output } \mathsf{same}^* \textit{ if } \tilde{c} = c, \textit{ and } \tilde{s} \textit{ otherwise.} \end{array} \right\}$$

which is a random variable over the randomness of Init, Enc *and* Dec. *The encoding scheme* $(\mathsf{Init}, \mathsf{Enc}, \mathsf{Dec})$ *is strongly non-malleable with respect to the function family $\mathcal{F}$, if for any $f \in \mathcal{F}$ and any s_0, $s_1 \in \{0,1\}^\kappa$,*

$$\left\{ \left(\Sigma, \mathsf{Tamper}_{s_0}^{\Sigma,f} \right) \right\}_{k \in \mathbb{N}} \approx \left\{ \left(\Sigma, \mathsf{Tamper}_{s_1}^{\Sigma,f} \right) \right\}_{k \in \mathbb{N}},$$

where $\Sigma \leftarrow \mathsf{Init}(1^k)$, and "$\approx$" may refer to statistical, or computational, indistinguishability, with parameter k.

In the above definition, f is parameterized by Σ to differentiate tamper-proof input, i.e., Σ, from tamperable input, i.e., c. Also, according to the standard definition of non-malleability, the decoding procedure is not randomized, however, as it is suggested by Ball et al. [BDKM16], Dec may be randomized.

Next, the *uniqueness* is defined due to [FMNV14]. The uniqueness property is essential for achieving non-malleability in the continuous setting against split-state attackers, which is the main subject of Chapter 6.

Definition 2.3.4 (Uniqueness [FMNV14])**.** *Let* $\mathsf{ES} = (\mathsf{Init}, \mathsf{Enc}, \mathsf{Dec})$ *be a split-state encoding scheme in the* CRS *model. Then,* ES *satisfies the uniqueness property if for any* PPT *algorithm $\mathcal{A}$ and all, sufficiently large $k \in \mathbb{N}$, we have:*

$$\Pr\left[\begin{array}{c} \Sigma \leftarrow \mathsf{Init}(1^k); (c_0, c_1, c_1') \leftarrow \mathcal{A}(1^k, \Sigma) : \\ \mathsf{Dec}(\Sigma, (c_0, c_1)) \neq \perp \wedge \mathsf{Dec}(\Sigma, (c_0, c_1')) \neq \perp \wedge c_1 \neq c_1' \end{array} \right] \leq \mathsf{negl}(k),$$

and symmetrically for the case in which we fix the right part of the codeword.

The definition of the split-state continuous tampering oracle due to [FMNV14] follows.

Definition 2.3.5 (The split-state tampering oracle $\mathcal{O}_{\mathsf{cnm}}$ [FMNV14])**.**
Let $(\mathsf{Init}, \mathsf{Enc}, \mathsf{Dec})$ *be a split-state, (κ, ν)-encoding scheme, in the* CRS *model. For any $(c_0, c_1) \in \{0,1\}^{\nu/2} \times \{0,1\}^{\nu/2}$, and any split-state function $f = (f_0, f_1)$, f_0, $f_1 : \{0,1\}^{\nu/2} \to$*

$\{0,1\}^{\nu/2}$, *define the stateful oracle* $\mathcal{O}_{\mathsf{cnm}}(\cdot,\cdot)$ *with initial state* $st := 0$, *as follows,*

$\mathcal{O}_{\mathsf{cnm}}((c_0,c_1),(f_0,f_1))$:
If $st = 1$, *return* $\perp$
$(\tilde{c}_0,\tilde{c}_1) \leftarrow (f_0(c_0), f_1(c_1))$
If $(c_0,c_1) = (\tilde{c}_0,\tilde{c}_1)$ *return* same^*
If $\mathsf{Dec}(\Sigma,(\tilde{c}_0,\tilde{c}_1)) = \perp$, *return* $\perp$ *and set* $st \leftarrow 1$
Else return $(\tilde{c}_0,\tilde{c}_1)$

where $\Sigma \leftarrow \mathsf{Init}(1^k)$.

The λ-bit leakage oracle, returning a total of at most λ bits.

Definition 2.3.6 (The λ-bit leakage oracle $\mathcal{O}^\lambda(\cdot,\cdot)$)**.** *A leakage oracle* $\mathcal{O}^\lambda(\cdot,\cdot)$, *is a stateful oracle, with initial state* $st := 0$. *For any* $\lambda \in \mathbb{N}$, *string* s, *and function* $g : \{0,1\}^{|s|} \to \{0,1\}^{\lambda'}$, *if* $\lambda' + st \leq \lambda$, $\mathcal{O}^\lambda(s,g)$ *outputs* $g(s)$, *and updates it state to* $st \leftarrow st + \lambda'$, *otherwise it outputs* $\perp$.

Below we provide the definition of *continuously non-malleable, leakage-resilient codes*, due to [FMNV14].

Definition 2.3.7 (Continuously non-malleable, leakage-resilient codes [FMNV14])**.** *Let* $\mathsf{ES} = (\mathsf{Init},\mathsf{Enc},\mathsf{Dec})$ *be a split-state encoding scheme in the* CRS *model, and let* $\lambda, q \in \mathbb{N}$. *Then,* ES *is a* q*-continuously* λ*-leakage resilient (* (q,λ)-CNMLR *) code, if for every, sufficiently large* $k \in \mathbb{N}$, *any pair of messages* s_0, $s_1 \in \{0,1\}^{\mathrm{poly}(k)}$, *and any* PPT *algorithm* $\mathcal{A}$,

$$\left\{\mathsf{Tamper}^{\mathsf{cnmlr}}_{\mathcal{A},s_0}(k)\right\}_{k\in\mathbb{N}} \approx_c \left\{\mathsf{Tamper}^{\mathsf{cnmlr}}_{\mathcal{A},s_1}(k)\right\}_{k\in\mathbb{N}},$$

where,

$$\mathsf{Tamper}^{\mathsf{cnmlr}}_{\mathcal{A},s}(k) := \left\{ \begin{array}{c} \Sigma \leftarrow \mathsf{Init}(1^k); (c_0,c_1) \leftarrow \mathsf{Enc}(\Sigma,s); \\ out \leftarrow \mathcal{A}^{\mathcal{O}^\lambda(c_0,\cdot),\mathcal{O}^\lambda(c_1,\cdot),\mathcal{O}_{\mathsf{cnm}}((c_0,c_1),\cdot)}(\Sigma); \mathit{Output} : out \end{array} \right\}$$

and $\mathcal{A}$ *makes at most* q *tampering queries against* $\mathcal{O}_{\mathsf{cnm}}$.

Chapter 3

Non-malleable codes for partial functions with manipulation detection

3.1 Introduction

The notion of *non-malleable codes* (NMC) was introduced by Dziembowski, Pietrzak and Wichs [DPW10], as a relaxation of error correction and error detection codes, aiming to provide strong privacy, without ensuring correctness. Informally, non-malleable encoding schemes are *keyless primitives*, guaranteeing that any modified codeword decodes, either to the original message, or to a completely unrelated one, with overwhelming probability. As non-malleability against general function classes is impossible (cf. Chapter 1), various subclasses of tampering functions have been considered, such as split-state functions [ADL14, DKO13, ADKO15, LL12, AAG+16, DPW10, KLT16], bitwise tampering and permutations [DPW10, AGM+15a, AGM+15b], bounded-size function classes [FMVW14], bounded depth/fan-in circuits [BDKM16], space-bounded tampering [FHMV17], and others (cf. Section 3.1.3). One characteristic shared by those function classes is that they allow *full access* to the codeword, while imposing structural or computational restrictions to the way the function computes over the input. In the present chapter, we initiate a study on non-malleability for functions that receive *partial access* over the codeword, which is an important yet overlooked class, as we elaborate below.

The class of partial functions. The class of *partial functions* contains all functions that read/write on an arbitrary subset of codeword bits,[1] with specific cardinality. Concretely,

[1] When considering alphabets larger than {0,1}, the function is accessing codeword symbols.

let c be a codeword with length ν. For $\alpha \in [0, 1)$, the function class $\mathcal{F}^{\alpha\nu}$ (or $\mathcal{F}^{\alpha}$ for brevity) consists of all functions that operate over any subset of bits of c with cardinality at most $\alpha\nu$, while leaving the remaining bits intact. The work of Cheraghchi and Guruswami [CG14] explicitly defines this class and uses a subclass (the one containing functions that always touch the first $\alpha\nu$ bits of the codeword) in a negative way, namely as the tool for deriving capacity lower bounds for *information-theoretic* non-malleable codes against split-state functions. Partial functions were also studied implicitly by Faust et al. [FMVW14], while aiming for non-malleability against bounded-size circuits.[2]

Even though capacity lower bounds for partial functions have been derived (cf. [CG14]), our understanding about *explicit* constructions is still limited. Existential results can be derived by the probabilistic method, as shown in prior works [DPW10, CG14],[3] but they do not yield explicit constructions. On the other hand, the capacity bounds do not apply to the computational setting, which could potentially allow more practical solutions. This is a direction that needs to be explored, as besides the theoretical interest, partial functions is a natural model that complies with existing attacks that require partial access to the registers of the cryptographic implementation [BDL97, BDL01, BS97, BDH$^+$98, TMA11].[4]

Besides the importance of partial functions in the active setting, i.e., when the function is allowed to partially *read*/*write* the codeword, the passive analogue of the class, i.e., when the function is only given *read access* over the codeword, matches the model considered by All-Or-Nothing Transforms (AONTs), which is a notion originally introduced by Rivest [Riv97], providing security guarantees similar to those of leakage resilience: reading an arbitrary subset (up to some bounded cardinality) of locations of the codeword does not reveal the underlying message. As non-malleable codes provide privacy, non-malleability for partial functions is the active analogue of (and in fact implies) AONTs, that find numerous applications [Riv97, Boy99, CDH$^+$00, Sti01, RP11].

Plausibility. At a first glance one might think that partial functions better comply with the framework of error-correction/detection codes (ECC/EDC), as they do not touch the

[2] Specifically, in [FMVW14], the authors consider a model where a common reference string (CRS) is available, with length roughly logarithmic in the size of the tampering function class; as a consequence, the tampering function is allowed to read/write the whole codeword while having only partial information over the CRS.

[3] Informally, prior works [DPW10, CG14] showed existence of non-malleable codes for classes of certain bounded cardinalities. The results cover the class of partial functions.

[4] The attacks by [BDL97, BDL01, BDH$^+$98] require the modification of a single (random) memory bit, while in [BS97] a single error per each round of the computation suffices. In [TMA11], the attack requires a single faulty byte.

whole codeword. However, if we allow the adversary to access asymptotically almost the entire codeword, it is conceivable it can use this generous *access rate*, i.e., the fraction of the codeword that can be accessed (see below), to create correlated encodings, thus solving non-malleability in this setting is essential. Additionally, non-malleability provides simulation based security, which is not considered by ECC/EDC.

We illustrate the separation between the notions using the following example. Consider the set of partial functions that operate either on the right or on the left half of the codeword (the function chooses if it is going to be left or right), and the trivial encoding scheme that on input message s, outputs (s, s). The decoder, on input (s, s'), checks if $s = s'$, in which case it outputs s, otherwise it outputs $\perp$. This scheme is clearly an EDC against the aforementioned function class,[5] as the output of the decoder is in $\{s, \perp\}$, with probability 1; however, it is malleable since the tampering function can create encodings whose validity depends on the message. On the other hand, an ECC would provide a trivial solution in this setting, however it requires restriction of the adversarial access fraction to 1/2 (of the codeword); by accessing more than this fraction, the attacker can possibly create invalid encodings depending on the message, as general ECCs do not provide privacy. Thus, the ECC/EDC setting is inapt when aiming for simulation based security in the presence of attackers that access almost the entire codeword. Later in this section, we provide an extensive discussion on challenges of non-malleability for partial functions.

Besides the plausibility and the lack of a comprehensive study, partial functions can potentially allow stronger primitives, as constant functions are excluded from the class. This is similar to the path followed by Jafargholi and Wichs [JW15], aiming to achieve *tamper detection* (cf. Section 3.1.3) against a class of functions that implicitly excludes constant functions and the identity function. In what follows we prove that this intuition holds, by showing that partial functions allow a stronger primitive that we define as *non-malleability* with *manipulation detection* (MD-NMC), which in addition to simulation based security, also guarantees that any tampered codeword will either decode to the original message, or to $\perp$. Again, and as in the case of ECC/EDC, we stress that manipulation/tamper-detection codes do not imply MD-NMC, as they do not provide simulation based security.[6]

Having the above discussion in mind, it is clear that partial functions is an interesting and well-motivated model, and the goal of the present chapter is to answer the following

[5] It is not an ECC as the decoder does not know which side has been modified by the tampering function.

[6] Clearly, MD-NMC imply manipulation/error-detection codes.

(informally stated) question:

> *Is it possible to construct efficient (high information rate) non-malleable codes for partial functions, while allowing the attacker to access almost the entire codeword?*

In what follows we answer the above question in the affirmative. Before presenting the results of the present chapter (cf. Section 3.1.1), we identify several challenges that are involved in tackling the problem.

Challenges. We first define some useful notions that will be used later.

- *Information rate*: the ratio of message to codeword length, as the message length goes to infinity.
- *Access rate*: the fraction of the number of bits (resp. symbols)[7] that the attacker is allowed to access, over the total codeword bits (resp. symbols), as the message length goes to infinity.

The access rate measures the effectiveness of a non-malleable code in the partial function setting and reflects the level of adversarial access to the codeword. In this chapter, we aim at constructing non-malleable codes for partial functions with high *information rate* and high *access rate*, i.e., both rates should approach 1, simultaneously. Before discussing the challenges posed by this requirement, we first review some known impossibility results. First, non-malleability for partial functions with concrete access rate 1 is impossible, as the function can fully decode the codeword and then re-encode a related message [DPW10]. Second, information-theoretic non-malleable codes with constant information rate (e.g., 0.5) are not possible against partial functions with constant access rate [CG14],[8] and consequently, solutions in the information-theoretic settings such as ECC and Robust Secret Sharing (RSS) do not solve our problem. Based on these facts, in order to achieve our goal, the only path is to explore the computational setting, aiming for access rate at most $1 - \epsilon$, for some $\epsilon > 0$.

At a first glance one might think that non-malleability for partial functions is easier to achieve, compared to other function classes, as partial functions cannot touch the whole

[7] This is of the case of bigger alphabets.

[8] Informally, in [CG14] (Theorem 5.3) the authors showed that any information-theoretic non-malleable code with a constant access rate and a constant information rate must have a constant distinguishing probability.

codeword. Having that in mind, it would be tempting to conclude that existing designs/techniques with minor modifications are sufficient to achieve our goal. However, we will show that this intuition is misleading, by pointing out why prior approaches fail to provide security against partial functions with high access rate.

The current state of the art in the computational setting considers tools such as (Authenticated) Encryption [DLSZ15, AAG$^+$16, KLT16, FN17, LL12, DKS17a], *non-interactive zero-knowledge* (NIZK) proofs [LL12, FN17, FMNV14, DKS17a], and ℓ-more weakly extractable collision resistant hashes [KLT16], where others use KEM/DEM techniques [DLSZ15, AAG$^+$16]. Those constructions share a common structure, incorporating a short secret key sk (or a short encoding of it), as well as a long ciphertext, e, and a proof π (or a hash value). Now, consider the partial function f that gets full access to the secret key sk and a constant number of bits of the ciphertext e, partially decrypts e and modifies the codeword depending on those bits. Then, it is not hard to see that non-malleability falls apart, as the security of the encryption no longer holds. The attack requires access rate only $O((|sk|)/(|sk| + |e| + |\pi|))$, for [LL12, FN17, DKS17a] and $O(\text{poly}(k)/|s|)$ for [DLSZ15, AAG$^+$16, KLT16]. A similar attack applies to [FMNV14], which is in the continuous setting.

One possible route to tackle the above challenges, is to use an encoding scheme over the ciphertext, such that partial access over it does not reveal the underlying message.[9] The guarantees that we need from such a primitive resemble the properties of AONTs, however this primitive does not provide security against active, i.e., tampering, attacks. Another approach would be to use Reconstructable Probabilistic Encodings [BDKM16], which provide error-correcting guarantees, yet still it is unknown whether we can achieve information rate 1 for such a primitive. In addition, the techniques and tools for protecting the secret key can be used to achieve optimal information rate as they are independent of the underlying message, yet at the same time, they become the weakest point against partial functions with high access rate. Thus, the question is how to overcome the above challenges, allowing access to almost the entire codeword.

In the present thesis we solve the challenges presented above based on the following observation: in existing solutions the structure of the codeword is fixed and known to the attacker, and independently of the primitives that we use, the only way to resolve the above issues is by hiding the structure via randomization. This requires a structure recovering mechanism that can either be implemented by an "external" source, or otherwise

[9] In the presence of NIZKs we can have attacks with low access rate that read sk, e, and constant number of bits from the proof.

the structure needs to be reflected in the codeword in some way that the attacker cannot exploit. In the present thesis we implement this mechanism in both ways, by first proposing a construction in the *common reference string* (CRS) model (cf. Section 3.3), and then we show how to remove the CRS using slightly bigger alphabets (cf. Section 3.4).

3.1.1 Results

In the present chapter, we introduce the notion of *non-malleable codes* with *manipulation-detection* (MD-NMC), and we present the first construction for this type of codes. We focus on achieving simultaneously high *information rate* and high *access rate*, in the partial functions setting, which by the results of [CG14], it can be achieved only in the computational setting.

The contribution is threefold. First, we construct an information rate 1 non-malleable code in the CRS model, with access rate $1 - 1/\Omega(\log k)$, where k denotes the security parameter. The proposed construction combines Authenticated Encryption together with an inner code that protects the key of the encryption scheme (cf. Section 3.3). The result is informally summarized in the following theorem.

Theorem 3.1.1 (Informal)**.** *Assuming one-way functions, there exists an explicit computationally secure* MD-NMC *in the* CRS *model, with information rate* 1 *and access rate* $1 - 1/\Omega(\log k)$*, where k is the security parameter.*

The proposed scheme, in order to achieve security with error $2^{-\Omega(k)}$, produces codewords of length $|s| + O(k^2 \log k)$, where $|s|$ denotes the length of the message, and uses a CRS of length $O(k^2 \log k \log(|s| + k))$. We note that the construction does not require the CRS to be fully tamper-proof and we refer the reader to the end of Section 3.3 for a general discussion on the topic.

As a second result, we show how to remove the CRS by slightly increasing the size of the alphabet. This yields a computationally secure MD-NMC in the standard model, achieving information and access rate $1 - 1/\Omega(\log k)$. The proposed construction is proven secure by a reduction to the security of the scheme presented in Theorem 4.1.4. Below, we informally state the result.

Theorem 3.1.2 (Informal)**.** *Assuming one-way functions, there exists an explicit, computationally secure* MD-NMC *in the standard model, with alphabet length $O(\log k)$, information rate $1 - 1/\Omega(\log k)$ and access rate $1 - 1/\Omega(\log k)$, where k is the security parameter.*

The scheme produces codewords of length $|s|(1 + 1/O(\log k)) + O(k^2 \log^2 k)$.

In Section 3.5, we consider security against continuous attacks. We show how to achieve a weaker notion of continuous security, while avoiding the use of a self-destruct mechanism, which was originally achieved by [FN17]. Our notion is weaker than full continuous security [FMNV14], since the codewords need to be updated. Nevertheless, our update operation is deterministic and avoids the full re-encoding process [DPW10, LL12]; it uses only shuffling and refreshing operations, i.e., we avoid cryptographic computations such as group operations and NIZKs. We call such an update mechanism a "light update." Informally, we prove the following result.

Theorem 3.1.3 (Informal)**.** *One-way functions imply continuous non-malleable codes with deterministic light updates and without self-destruct, in the standard model, with alphabet length $O(\log k)$, information rate $1 - 1/\Omega(\log k)$ and access rate $1 - 1/\Omega(\log k)$, where k is the security parameter.*

As we stated earlier in this chapter, non-malleable codes against partial functions imply AONTs [Riv97]. The first AONT was presented by Boyko [Boy99] in the random oracle model, and then Canetti et al. [CDH$^+$00] consider AONTs with public/private parts as well as a secret-only part, which is the full notion. Canetti et al. [CDH$^+$00] provide efficient constructions for both settings, yet the fully secure AONT (called "secret-only" in that paper) is based on non-standard assumptions.[10]

Assuming one-way functions, our results yield efficient, fully secure AONTs, in the standard model. This resolves, the open question left in [CDH$^+$00], where the problem of constructing AONTs under standard assumptions was posed. Our result is presented in the following theorem.

Theorem 3.1.4 (Informal)**.** *Assuming one-way functions, there exists an explicit secret-only* AONT *in the standard model, with information rate* 1 *and access rate $1 - 1/\Omega(\log k)$, where k is the security parameter.*

The above theorem is derived by the Informal Theorem 4.1.4, yielding an AONT whose output consists of both the CRS and the codeword produced by the NMC scheme in the CRS model. A similar theorem can be derived with respect to the Informal Theorem 3.1.2. Finally, and in connection to AONTs that provide leakage resilience, our results imply leakage-resilient codes [LL12] for partial functions.

[10] In [Sti01] the authors present a deterministic AONT construction that provides weaker security.

Later in this chapter, we provide concrete instantiations of the proposed constructions, using textbook instantiations ([KL14]) for the underlying authenticated encryption scheme. For completeness, we also provide information theoretic variants of our constructions that maintain high access rate and thus necessarily sacrifice information rate.

3.1.2 Applications of MD-NMC for partial functions

In the present section we present applications of MD-NMC for the class of partial functions.

Security against passive attackers - AONTs. Regarding the passive setting, our model and constructions find useful application in all settings where AONTs are useful (cf. [Riv97, Boy99, CDH$^+$00, RP11]), e.g., for increasing the security of encryption without increasing the key-size, for improving the efficiency of block ciphers and constructing remotely keyed encryption [Riv97, Boy99], and also for constructing computationally secure secret sharing [RP11]. Other uses of AONTs are related to optimal asymmetric encryption padding [Boy99].

Security against memory tampering - (binary alphabets, logarithmic length CRS). As with every NMC, the most notable application of the proposed model and constructions is when aiming for protecting cryptographic devices against memory tampering. Using our CRS based construction we can protect a large tamperable memory with a small (logarithmic in the message length) tamperproof memory, that holds the CRS.

The construction is as follows. Consider any device performing cryptographic operations, e.g., a smart card, whose memory is initialized when the card is being issued. Each card is initialized with an independent CRS, which is stored in a tamper-proof memory, while the codeword is stored in a tamperable memory. Due to the independency of the CRS values, it is plausible to assume that the adversary is not given access to the CRS prior to tampering with the card; the full CRS is given to the tampering function while it tampers with the codeword during computation. This idea is along the lines of the *only computation leaks information* model [MR04], where data can only be leaked during computation, i.e., the attacker learns the CRS when the devices performs computations that depend on it. We note that in this work we allow the tampering function to read the full CRS, in contrast to [FMVW14], in which the tampering function receives partial information over it (our CRS can also be tampered, cf. the end of the current section). In subsequent rounds, the CRS and the codeword are being updated by the device, which is the

standard way to achieve security in multiple rounds while using a one-time NMC [DPW10].

Security against memory tampering - (logarithmic length alphabets, no CRS). In modern architectures data is stored and transmitted in chunks, thus our block-wise encoding scheme can provide tamper-resilience in all these settings. For instance, consider the case of arithmetic circuits, having memory consisting of consecutive blocks storing integers. Considering adversaries that access the memory of such circuits in a block-wise manner, is a plausible scenario. In terms of modeling, this is similar to tamper-resilience for arithmetic circuits [GIP+14], in which the attacker, instead of accessing individual circuit wires carrying bits, it accesses wires carrying integers. The case is similar for RAM computation where the CPU operates over 32 or 64 bit registers (securing RAM programs using NMC was also considered by [FMNV15, DLSZ15, DKS17b, DKS18]). We note that, the memory segments in which the codeword blocks are stored do not have to be physically separated, as partial functions output values that depend on the whole input in which they receive access to. This is in contrast to the split-state setting in which the tampering function tampers with each state independently, and thus the states need to be physically separated.

Security against adversarial channels. In Wiretap Channels [BTV12, Wyn75, OW] the goal is to communicate data privately against eavesdroppers, under the assumption that the channel between the sender and the adversary is "noisier" than the channel between the sender and the receiver. The model that we propose and our block-wise construction can be applied in this setting to provide privacy against a wiretap adversary under the assumption that due to the gap of noise there is a small (of rate $o(1)$) fraction of symbols that are delivered intact to the receiver and dropped from the transmission to the adversary. This enables private, key-less communication between the parties, guaranteeing that the receiver will either receive the original message, or $\perp$. In this way, the communication will be non-malleable in the sense that the receiver cannot be lead to output $\perp$ depending on any property of the plaintext. Our model allows the noise in the receiver side to depend on the transmission to the wiretap adversary, that tampers with a large (of rate $1 - o(1)$) fraction of symbols, leading to an "active" variant of the wiretap model.

3.1.3 Related work

Manipulation detection has been considered independently of the notion of non-malleability, in the seminal paper by Cramer et. al. [CDF+08], who introduced the notion of *algebraic*

manipulation detection (AMD) codes, providing security against additive attacks over the codeword. A similar notion was considered by Jafargholi and Wichs [JW15], called *tamper detection*, aiming to detect malicious modifications over the codeword, independently of how those affect the output of the decoder. Tamper detection ensures that the application of any (admissible) function to the codeword leads to an invalid decoding.

Non-malleable codes for other function classes have been extensively studied, such as constant split-state functions [CZ14, DNO17]. The first explicit non-malleable code in the split-state model, for the information-theoretic setting was proposed by [DKO13], yet their scheme can only encode single-bit messages. Subsequent constructions for multi-bit messages are discussed in subsequent chapters. Non-malleable codes for other function classes have been extensively studied, e.g., bit-wise independent tampering [DPW10], bounded-size function classes [FMVW14], the k-split setting [CZ14], block-wise tampering [CKM11, CGM+15], and bounded depth and fan-in circuits [BDKM16]. The work of [ADKO15] develops beautiful connections among different function classes. In [BDKM18] the authors consider AC0 circuits, bounded-depth decision trees and streaming, space-bounded adversaries.

Other aspects of non-malleable codes have also been studied, such as rate-function class tradeoff, in the information-theoretic setting [CG14]. Other variants of non-malleable codes have been proposed, such as continuous non-malleable codes [FMNV14, JW15], augmented non-malleable codes [AAG+16], locally decodable/updatable non-malleable codes [DLSZ15, DKS17b, FMNV15, CKR16, DKS18], which were used to secure the implementation of RAM computation, and non-malleable codes with split-state refresh [FN17]. Leakage resilience was also considered as an additional feature, e.g., [LL12, DLSZ15, CKR16, FN17].

A related line of work in tamper resilience aims to protect circuit computation against tampering attacks on circuit wires [IPSW06, FPV11, DK12, DK14] or gates [KT13]. In this setting, using non-malleable codes for protecting the circuit's private memory is an option, still in order to achieve security the encoding and decoding procedures should be protected against fault injection attacks using the techniques from [IPSW06, FPV11, DK12, DK14, KT13].

3.2 Preliminaries on MD-NMC

In the present chapter, we exploit the fact that the class of partial functions does not include constant functions and we achieve a notion that is stronger than non-malleability, which

we call *non-malleability* with *manipulation detection*. Below, we formalize this notion as a strengthening of non-malleability and we show that the proposed constructions achieve this stronger notion. Informally, manipulation detection ensures that any tampered codeword will either decode to the original message, or to $\perp$, with overwhelming probability.

The definition provided below is with respect to alphabets, as in Section 3.4 we consider alphabets of size $O(\log k)$.

Definition 3.2.1 (Non-Malleability with Manipulation Detection (MD-NMC))**.** *Let Γ be an alphabet, let* $(\mathsf{Init}, \mathsf{Enc}, \mathsf{Dec})$ *be a (κ, ν)-encoding scheme in the common reference string model, and $\mathcal{F}$ be a family of functions $f : \Gamma^\nu \to \Gamma^\nu$. For any $f \in \mathcal{F}$ and $s \in \{0,1\}^\kappa$, define the tampering experiment*

$$\mathsf{Tamper}_s^f := \left\{ \begin{array}{c} \Sigma \leftarrow \mathsf{Init}(1^k), c \leftarrow \mathsf{Enc}(\Sigma, s), \tilde{c} \leftarrow f_\Sigma(c), \tilde{s} \leftarrow \mathsf{Dec}(\Sigma, \tilde{c}) \\ \textit{Output} : \tilde{s}. \end{array} \right\}$$

which is a random variable over the randomness of Enc, Dec *and* Init. *The encoding scheme* $(\mathsf{Init}, \mathsf{Enc}, \mathsf{Dec})$ *is non-malleable with manipulation detection with respect to the function family $\mathcal{F}$, if for all, sufficiently large k and for all $f \in \mathcal{F}$, there exists a distribution $D_{(\Sigma,f)}$ over $\{0,1\}^\kappa \cup \{\perp, \mathsf{same}^*\}$, such that for all $s \in \{0,1\}^\kappa$, we have:*

$$\left\{\mathsf{Tamper}_s^f\right\}_{k\in\mathbb{N}} \approx \left\{ \begin{array}{c} \tilde{s} \leftarrow D_{(\Sigma,f)} \\ \textit{Output } s \textit{ if } \tilde{s} = \mathsf{same}^*, \textit{ and } \perp \textit{ otherwise} \end{array} \right\}_{k\in\mathbb{N}}$$

where $\Sigma \leftarrow \mathsf{Init}(1^k)$ and $D_{(\Sigma,f)}$ is efficiently samplable given access to f, Σ. Here, "$\approx$" may refer to statistical, or computational, indistinguishability.

In the above definition, f is parameterized by Σ to differentiate tamper-proof input, i.e., Σ, from tamperable input, i.e., c. The following lemma is useful for proving MD-NMC security throughout the chapter.

Lemma 3.2.2. *Let* $(\mathsf{Enc}, \mathsf{Dec})$ *be a (κ, ν)-coding scheme and $\mathcal{F}$ be a family of functions. For every $f \in \mathcal{F}$ and $s \in \{0,1\}^\kappa$, define the tampering experiment*

$$\mathsf{Tamper}_s^f := \left\{ \begin{array}{c} c \leftarrow \mathsf{Enc}(s), \tilde{c} \leftarrow f(c), \tilde{s} \leftarrow \mathsf{Dec}(\tilde{c}) \\ \textit{Output } \mathsf{same}^* \textit{ if } \tilde{s} = s, \textit{ and } \tilde{s} \textit{ otherwise.} \end{array} \right\}$$

which is a random variable over the randomness of Enc *and* Dec. $(\mathsf{Enc}, \mathsf{Dec})$ *is an* MD-NMC *with respect to $\mathcal{F}$, if for any $f \in \mathcal{F}$ and all sufficiently large k: (i) for any pair of messages s_0, $s_1 \in \{0,1\}^\kappa$,*

$$\left\{\mathsf{Tamper}_{s_0}^f\right\}_{k\in\mathbb{N}} \approx \left\{\mathsf{Tamper}_{s_1}^f\right\}_{k\in\mathbb{N}},$$

and (ii) for any s,

$$\Pr\left[\mathsf{Tamper}_s^f \notin \{\perp, s\}\right] \leq \mathsf{negl}(k).$$

Here, "$\approx$" may refer to statistical, or computational, indistinguishability.

Proof. By Definition 3.2.1 we have that $(\mathsf{Enc}, \mathsf{Dec})$ is an MD-NMC against $\mathcal{F}$, if for any $f \in \mathcal{F}$, there exists an efficiently samplable distribution D_f over $\{0,1\}^k \cup \{\perp, \mathsf{same}^*\}$, such that for any message s

$$\left\{ \begin{array}{c} c \leftarrow \mathsf{Enc}(s), \tilde{c} \leftarrow f(c), \tilde{s} \leftarrow \mathsf{Dec}(\tilde{c}) \\ \text{Output} : \tilde{s} \end{array} \right\} \approx \left\{ \begin{array}{c} \tilde{s} \leftarrow D_f \\ \text{Output } s \text{ if } \tilde{s} = \mathsf{same}^*, \text{ and } \perp \text{ otherwise} \end{array} \right\} \tag{3.1}$$

Let $\mathsf{0}$ be the zero message in $\{0,1\}^\kappa$. For any $f \in \mathcal{F}$, we define D_f as follows:

- Sample $c \leftarrow \mathsf{Enc}(\mathsf{0})$ and compute $\tilde{c} \leftarrow f(c)$, $\tilde{s} \leftarrow \mathsf{Dec}(\tilde{c})$.
- **Output**: if $\tilde{s} = \mathsf{0}$, set $\tilde{s} \leftarrow \mathsf{same}^*$, else, $\tilde{s} \leftarrow \perp$. Output $\tilde{s}$.

From the above we have that for any s,

$$\begin{aligned} \left\{ \begin{array}{c} \tilde{s} \leftarrow D_f \\ \\ \text{Output } s \text{ if } \tilde{s} = \mathsf{same}^*, \text{ and } \perp \text{ otherwise} \end{array} \right\} &\equiv \left\{ \left\{ \begin{array}{c} c \leftarrow \mathsf{Enc}(\mathsf{0}), \tilde{c} \leftarrow f(c), \tilde{s} \leftarrow \mathsf{Dec}(\tilde{c}) \\ \text{if } \tilde{s} = \mathsf{0},\ \tilde{s} \leftarrow \mathsf{same}^*, \text{ else, } \tilde{s} \leftarrow \perp. \text{ Output } \tilde{s} \end{array} \right\} \right\} \\ &\approx \left\{ \left\{ \begin{array}{c} c \leftarrow \mathsf{Enc}(s), \tilde{c} \leftarrow f(c), \tilde{s} \leftarrow \mathsf{Dec}(\tilde{c}) \\ \text{if } \tilde{s} = s,\ \tilde{s} \leftarrow \mathsf{same}^*, \text{ else, } \tilde{s} \leftarrow \perp. \text{ Output } \tilde{s} \end{array} \right\} \right\} \\ &\approx \left\{ \begin{array}{c} c \leftarrow \mathsf{Enc}(s), \tilde{c} \leftarrow f(c), \tilde{s} \leftarrow \mathsf{Dec}(\tilde{c}) \\ \text{Output} : \tilde{s} \end{array} \right\}, \end{aligned}$$

where the first relation follows by the definition of D_f, the second one follows from the main assumption which states that for any pair of messages s_0, s_1, $\mathsf{Tamper}_{s_0}^f \approx \mathsf{Tamper}_{s_1}^f$, and the third one follows from the assumption that $\Pr\left[\mathsf{Tamper}_s^f \notin \{\perp, s\}\right] \leq \mathsf{negl}(k)$. This concludes our proof since for any $f \in \mathcal{F}$ and any message s, Relation 3.1 is satisfied. □

For encoding schemes in the CRS model the above lemma is similar, and Tamper_s^f internally samples $\Sigma \leftarrow \mathsf{Init}(1^k)$.

3.3 An MD-NMC for partial functions, in the CRS model

In this section we consider $\Gamma = \{0,1\}$ and we construct a rate 1 MD-NMC for $\mathcal{F}^\alpha$, with access rate $\alpha = 1 - 1/\Omega(\log k)$.

Before presenting the encoding scheme for $\mathcal{F}^{\alpha}$, we provide the intuition behind the construction. As a staring point, we consider a naive scheme (which does not work), and then show how we resolve all the challenges. Let $(\mathsf{KGen}, \mathsf{E}, \mathsf{D})$ be a (symmetric) authenticated encryption scheme and consider the following encoding scheme: to encode a message s, the encoder computes $(sk||e)$, where $e \leftarrow \mathsf{E}_{sk}(s)$ is the ciphertext and $sk \leftarrow \mathsf{KGen}(1^k)$, is the secret key. We observe that the scheme is secure if the tampering function can only read-/write on the ciphertext, e, assuming the authenticity property of the encryption scheme, however, restricting access to sk, which is short, is unnatural and makes the problem trivial. On the other hand, even partial access to sk, compromises the authenticity property of the scheme, and even if there is no explicit attack against the non-malleability property of the code, there is no hope for proving security based on the properties of $(\mathsf{KGen}, \mathsf{E}, \mathsf{D})$, in a black-box way.

A solution to the above problems would be to protect the secret key using an inner encoding, yet the amount of tampering is now restricted by the capabilities of the inner scheme, as the attacker knows the exact locations of the "*sensitive*" codeword bits, i.e., the non-ciphertext bits. In the proposed construction, we manage to protect the secret key while avoiding the bottleneck on the access rate, by designing an inner encoding scheme that provides limited security guarantees when used standalone, still when it is used in conjunction with a *shuffling technique* that permutes the inner encoding and ciphertext bit locations, it guarantees that any attack against the secret key will create an invalid encoding with overwhelming probability, even when allowing access to *almost the entire* codeword.

The proposed scheme is depicted in Figure 3.1 and works as follows: on input message s, the encoder (i) encrypts the message by computing $sk \leftarrow \mathsf{KGen}(1^k)$ and $e \leftarrow \mathsf{E}_{sk}(s)$, (ii) computes an m-out-of-m secret sharing, z, of $(sk||sk^3)$ (interpreting both sk and sk^3 as elements in some finite field),[11] and outputs a random shuffling of $(z||e)$, denoted as $P_{\Sigma}(z||e)$, according to the common reference string, Σ. Decoding proceeds as follows: on input c, the decoder (i) inverts the shuffling operation by computing $(z||e) \leftarrow P_{\Sigma}^{-1}(c)$, (ii) reconstructs $(sk||sk')$, and (iii) if $sk^3 = sk'$, it outputs $\mathsf{D}_{sk}(e)$, otherwise, it outputs $\perp$. The proposed instantiation yields a rate 1 computationally secure MD-NMC in the CRS model, with access rate $1 - 1/\Omega(\log k)$ and codewords of length $|s| + O(k^2 \log k)$, under mild assumptions, e.g., one way functions.

[11] In general, any polynomial of small degree, e.g., sk^c, would suffice, depending on the choice of the underlying finite field. Using sk^3 suffices when working over fields of characteristic 2. We could also use sk^2 over fields of characteristic 3.

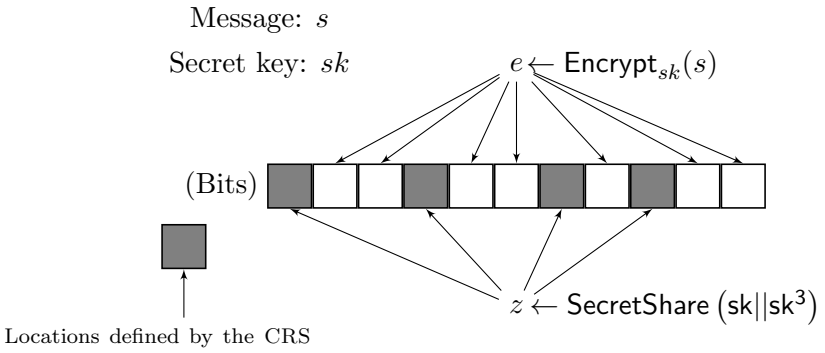

Figure 3.1: Description of the MD-NMC scheme in the CRS model.

Below, we formally define our construction.

Construction 3.3.1. *Let k, $m \in \mathbb{N}$, let $(\mathsf{KGen}, \mathsf{E}, \mathsf{D})$ be a symmetric encryption scheme, $(\mathsf{SS}_m, \mathsf{Rec}_m)$ be an m-out-of-m secret sharing scheme, and let $l \leftarrow 2m|sk|$, where sk follows $\mathsf{KGen}(1^k)$. We define an encoding scheme $(\mathsf{Init}, \mathsf{Enc}, \mathsf{Dec})$, that outputs $\nu = l + |e|$ bits, $e \leftarrow \mathsf{E}_{sk}(s)$, as follows:*

- $\mathsf{Init}(1^k)$*: Sample $r_1, \ldots, r_l \xleftarrow{rs} \{0,1\}^{\log(\nu)}$, and output $\Sigma := (r_1, \ldots, r_l)$.*
- $\mathsf{Enc}(\Sigma, \cdot)$*: for input message s, sample $sk \leftarrow \mathsf{KGen}(1^k)$, $e \leftarrow \mathsf{E}_{sk}(s)$.*
  - **(Secret share)** *Sample $z \leftarrow \mathsf{SS}_m(sk||sk^3)$, where $z = ||_{i=1}^{2|sk|} z_i$, $z \in \{0,1\}^{2m|sk|}$, and for $i \in [|sk|]$, z_i (resp. $z_{|sk|+i}$) is an m-out-of-m secret sharing of $sk[i]$ (resp. $sk^3[i]$).*
  - **(Shuffle)** *Compute $c \leftarrow P_\Sigma(z||e)$ as follows:*
    1. **(Sensitive bits)***: Set $c \leftarrow 0^\nu$. For $i \in [l]$, $c[r_i] \leftarrow z[i]$.*
    2. **(Ciphertext bits)***: Set $i \leftarrow 1$. For $j \in [l + |e|]$, if $j \notin \{r_p \mid p \in [l]\}$: $c[j] \leftarrow e[i]$, i++.*

  Output c.
- $\mathsf{Dec}(\Sigma, \cdot)$*: on input c, compute $(z||e) \leftarrow P_\Sigma^{-1}(c)$, $(sk||sk') \leftarrow \mathsf{Rec}_m(z)$, and if $sk^3 = sk'$, output $\mathsf{D}_{sk}(e)$, otherwise output $\perp$.*

The set of indices of z_i in the codeword will be denoted by Z_i.

In the above, we consider sk, sk^3, as elements over $\mathbf{GF}(2^{\mathrm{poly}(k)})$.

In a high level, the construction presented above, combines authenticated encryption with an inner encoding that works as follows. It interprets sk as an element in the finite field $\mathbf{GF}(2^{|sk|})$ and computes sk^3 as a field element. Then, for each bit of $(sk||sk^3)$, it computes an m-out-of-m secret sharing of the bit, for some parameter m (we note that elements in $\mathbf{GF}(2^{|sk|})$ can be interpreted as bit strings). Then, by combining the inner encoding with the shuffling technique, we get an encoding scheme whose security follows from the observations that we briefly present below:

- For any tampering function which does not have access to all m shares of a single bit of $(sk||sk^3)$, the tampering effect on the secret key can be expressed essentially as a linear shift, i.e., as $((sk+\delta)||(sk^3+\eta))$ for some $(\delta,\eta) \in \mathbf{GF}(2^{|sk|}) \times \mathbf{GF}(2^{|sk|})$, independent of sk.

- By permuting the locations of the inner encoding and the ciphertext bits, we have that with overwhelming probability any tampering function who reads/writes on a $(1-o(1))$ fraction of codeword bits, will not learn any single bit of $(sk||sk^3)$.

- With overwhelming probability over the randomness of sk and the CRS, for non-zero η and δ, $(sk+\delta)^3 \neq sk^3+\eta$, and this property enables us to design a consistency check mechanism whose output is simulatable, without accessing sk.

- The security of the final encoding scheme follows by composing the security of the inner encoding scheme with the authenticity property of the encryption scheme.

Below we present the formal security proof of the above ideas.

Theorem 3.3.2. *Let k, $m \in \mathbb{N}$ and $\alpha \in [0,1)$. Assuming $(\mathsf{SS}_m, \mathsf{Rec}_m)$ is an m-out-of-m secret sharing scheme and $(\mathsf{KGen}, \mathsf{E}, \mathsf{D})$ is* 1-IND-CPA *secure (cf. Definition 2.1.6),*[12] *authenticated encryption scheme, the code of Construction 3.3.1 is a* MD-NMC *against $\mathcal{F}^{\alpha}$ (cf. Definition 2.2.1), for any α, m, such that $(1-\alpha)m = \omega(\log(k))$.*

Proof. Let I be the set of indices chosen by the attacker and $I^{\mathsf{c}} = [\nu] \backslash I$, where $\nu = 2m|sk| + |e|$. The tampered components of the codeword will be denoted using the symbol "~" on top of the original symbol, i.e., we have $\tilde{c} \leftarrow f(c)$, the tampered secret key sk (resp. sk^3) that we get after executing $\mathsf{Rec}_m(\tilde{z})$ will be denoted by $\tilde{sk}$ (resp. $\tilde{sk}'$). Also the

[12] This is an abbreviations for indistinguishability under chosen plaintext attack, for a single pre-challenge query to the encryption oracle.

tampered ciphertext will be $\tilde{e}$. We prove the needed using a series of hybrid experiments that are depicted in Figure 5.1. Below, we describe the hybrids.

$\mathsf{Exp}_0^{f,s}$:
$\Sigma \leftarrow \mathsf{Init}(1^k)$
$c \leftarrow \mathsf{Enc}(\Sigma, s), \tilde{c} \leftarrow 0^\nu$
$\tilde{c}[I] \leftarrow f_\Sigma(c_{|_I}), \tilde{c}[I^\mathsf{c}] \leftarrow c_{|_{I^\mathsf{c}}}$
$\tilde{s} \leftarrow \mathsf{Dec}(\tilde{c})$

Output same^* if $\tilde{s} = s$ and $\tilde{s}$ otherwise.

$\mathsf{Exp}_1^{f,s}$:
$\Sigma \leftarrow \mathsf{Init}(1^k)$
$c \leftarrow \mathsf{Enc}(\Sigma, s), \tilde{c} \leftarrow 0^\nu$
$\tilde{c}[I] \leftarrow f_\Sigma(c_{|_I}), \tilde{c}[I^\mathsf{c}] \leftarrow c_{|_{I^\mathsf{c}}}$
If $\exists i : |(I \cap Z_i)| = m$:
  $\tilde{s} \leftarrow \perp$
Else:
  $\tilde{s} \leftarrow \mathsf{Dec}(\tilde{c})$

Output same^* if $\tilde{s} = s$ and $\tilde{s}$ otherwise.

$\mathsf{Exp}_2^{f,s}$:
$\Sigma \leftarrow \mathsf{Init}(1^k)$
$sk \leftarrow \mathsf{KGen}(1^k), e \leftarrow \mathsf{E}_{sk}(s)$
$z^* \leftarrow \bar{\mathsf{SS}}_m^f(\Sigma, sk), c \leftarrow P_\Sigma(z^*||e)$
$\tilde{c} \leftarrow 0^\nu, \tilde{c}[I] \leftarrow f_\Sigma(c_{|_I}), \tilde{c}[I^\mathsf{c}] \leftarrow c_{|_{I^\mathsf{c}}}$

If $\exists i : |(I \cap Z_i)| = m$:
  $\tilde{s} \leftarrow \perp$
Else:
  If $\exists i : \bigoplus_{j \in (I \cap Z_i)} c[j] \neq \bigoplus_{j \in (I \cap Z_i)} \tilde{c}[j]$:
    $\tilde{s} \leftarrow \perp$
  Else:
    $\tilde{s} \leftarrow \mathsf{D}_{sk}(\tilde{e})$

Output same^* if $\tilde{s} = s$ and $\tilde{s}$ otherwise.

$\mathsf{Exp}_3^{f,s}$:
$\Sigma \leftarrow \mathsf{Init}(1^k)$
$sk \leftarrow \mathsf{KGen}(1^k), e \leftarrow \mathsf{E}_{sk}(s)$
$z^* \leftarrow \bar{\mathsf{SS}}_m^f(\Sigma, sk), c \leftarrow P_\Sigma(z^*||e)$

$\tilde{c} \leftarrow 0^\nu, \tilde{c}[I] \leftarrow f_\Sigma(c_{|_I})$

If $\exists i : |(I \cap Z_i)| = m$:
  $\tilde{s} \leftarrow \perp$
Else:
  If $\exists i : \bigoplus_{j \in (I \cap Z_i)} c[j] \neq \bigoplus_{i \in (I \cap Z_i)} \tilde{c}[j]$:
    $\tilde{s} \leftarrow \perp$
  Else: $\tilde{s} \leftarrow \perp$
    If $\tilde{e} = e$:
      $\tilde{s} \leftarrow \mathsf{same}^*$

Output $\tilde{s}$.

Figure 3.2: The hybrid experiments for the proof of Theorem 3.3.2. The gray part signifies the portion of the code of an experiment that differs from the previous one.

- $\mathsf{Exp}_0^{f,s}$: We prove security of our code using Lemma 3.2.2, i.e., by showing that (i) for any s_0, s_1, $\mathsf{Tamper}_{s_0}^f \approx \mathsf{Tamper}_{s_1}^f$, and (ii) for any s, $\Pr\left[\mathsf{Tamper}_s^f \notin \{\perp, s\}\right] \leq \mathsf{negl}(k)$, where Tamper_s^f is defined in Lemma 3.2.2. For any f, s, the first experiment, $\mathsf{Exp}_0^{f,s}$, matches the experiment Tamper_s^f in the CRS model, where Σ is sampled by Tamper_s^f.

- $\mathsf{Exp}_1^{f,s}$: In the second experiment we define Z_i, $i \in [2|sk|]$, to be the set of codeword indices in which the secret sharing z_i is stored, $|Z_i| = m$. The main difference from the previous experiment is that the current one outputs $\perp$, if there exists a bit of sk or sk^3 for which the tampering function reads all the shares of it, while accessing at most $\alpha\nu$ bits of the codeword. Intuitively, and as we prove in Claim 3.3.3, by permuting the location indices of $z||e$, this event happens with probability negligible in k, and the attacker does not learn any bit of sk and sk^3, even if it is given access to $(1 - o(1))\nu$ bits of the codeword.

- $\mathsf{Exp}_2^{f,s}$: By the previous hybrid we have that for all $i \in [2|sk|]$, the tampering function will not access all bits of z_i, with overwhelming probability. In the third experiment we unfold the encoding procedure, and in addition, we substitute the secret sharing procedure SS_m with $\bar{\mathsf{SS}}_m^f$ that computes shares z_i^* that reveal no information about $sk||sk^3$; for each i, $\bar{\mathsf{SS}}_m^f$ simply "drops" the bit of z_i with the largest index that is not being accessed by f. We formally define $\bar{\mathsf{SS}}_m^f$ below.

  $\bar{\mathsf{SS}}_m^f(\Sigma, sk)$:

  1. Sample $\left(z_1, \ldots, z_{2|sk|}\right) \leftarrow \mathsf{SS}_m\left(sk||sk^3\right)$ and set $z_i^* \leftarrow z_i$, $i \in [2|sk|]$.
  2. For $i \in [2|sk|]$, let $l_i := \max_d \{d \in [m] \wedge \mathsf{Ind}\left(z_i[d]\right) \notin I)\}$, where Ind returns the index of $z_i[d]$ in c, i.e., l_i is the largest index in $[m]$ such that $z_i[l_i]$ is not accessed by f.
  3. (**Output**): For all i set $z_i^*[l_i] = *$, and output $z^* :=\|_{i=1}^{2|sk|} z_i^*$.

  In $\mathsf{Exp}_1^{f,s}$, $z = \|_{i=1}^{2|sk|} z_i$, and each z_i is an m-out-of-m secret sharing for a bit of sk or sk^3. From Claim 3.3.3, we have that for all i, $|I \cap Z_i| < m$ with overwhelming probability, and we can observe that the current experiment is identical to the previous one up to the point of computing $f(c_{|_I})$, as $c_{|_I}$ and $f(c_{|_I})$ depend only on z^*, that carries no information about sk and sk^3.

  Another difference between the two experiments is in the external "Else" branch: $\mathsf{Exp}_1^{f,s}$ makes a call to the decoder while $\mathsf{Exp}_2^{f,s}$, before calling $\mathsf{D}_{sk}(\tilde{e})$, checks if the tampering function has modified the shares in a way such that the reconstruction procedure $((\tilde{sk}, \tilde{sk}') \leftarrow \mathsf{Rec}_m(\tilde{z}))$ will give $\tilde{sk} \neq sk$ or $\tilde{sk}' \neq sk'$. This check is done by the statement "If $\exists i : \bigoplus_{j \in (I \cap Z_i)} c[j] \neq \bigoplus_{j \in (I \cap Z_i)} \tilde{c}[j]$", without touching sk or

sk^3.[13] In case modification is detected the current experiments outputs $\perp$. The intuition is that an attacker that partially modifies the shares of sk and sk^3, creates shares of $\tilde{sk}$ and $\tilde{sk}'$, such that $\tilde{sk}^3 = \tilde{sk}'$, with negligible probability in k. We prove this by a reduction to the 1-IND-CPA security of the encryption scheme: any valid modification over the inner encoding of the secret key gives us method to compute the original original secret key sk, with non-negligible probability. The ideas are presented formally in Claim 3.3.4.

- $\mathsf{Exp}_3^{f,s}$: The difference between the current experiment and the previous one is that instead of executing the decryption, $\mathsf{D}_{sk}(\tilde{e})$, we first check if the attacker has modified the ciphertext, in which case the current experiment outputs $\perp$, otherwise it outputs same^*. By the previous hybrid, we reach this newly introduced "Else" branch of $\mathsf{Exp}_3^{f,s}$, only if the tampering function didn't modify the secret key. Thus, the indistinguishability between the two experiments follows from the authenticity property of the encryption scheme in the presence of z^*: given that $\tilde{sk} = sk$ and $\tilde{sk}' = sk'$, we have that if the attacker modifies the ciphertext, then with overwhelming probability $\mathsf{D}_{sk}(\tilde{e}) = \perp$, otherwise, $\mathsf{D}_{sk}(\tilde{e}) = s$, and the current experiment correctly outputs $\perp$ or same^* (cf. Claim 3.3.5).

- Finally, we prove that for any $f \in \mathcal{F}^\alpha$, and message s, $\mathsf{Exp}_3^{f,s}$ is indistinguishable from $\mathsf{Exp}_3^{f,0}$, where 0 denotes the zero-message. This follows by the semantic security of the encryption scheme, and gives us the indistinguishability property required by Lemma 3.2.2. The manipulation detection property is derived by the indistinguishability between the hybrids and the fact that the output of $\mathsf{Exp}_3^{f,s}$ is in the set $\{\mathsf{same}^*, \perp\}$.

In what follows, we prove indistinguishability between the hybrids using a series of claims.

Claim 3.3.3. *For k, $m \in \mathbb{N}$, assume $(1-\alpha)m = \omega(\log(k))$. Then, for any $f \in \mathcal{F}^\alpha$ and any message s, we have $\mathsf{Exp}_0^{f,s} \approx \mathsf{Exp}_1^{f,s}$, where the probability runs over the randomness used by* Init, Enc.

Proof. The difference between the two experiments is that $\mathsf{Exp}_1^{f,s}$ outputs $\perp$ when the attacker learns all shares of some bit of sk or sk^3, otherwise it produces output as $\mathsf{Exp}_0^{f,s}$ does. Let E be the event "$\exists i : |(I \cap Z_i)| = m$". Clearly, $\mathsf{Exp}_0^{f,s} = \mathsf{Exp}_1^{f,s}$ conditioned on $\neg E$, thus the statistical distance between the two experiments is bounded by $\Pr[E]$. In the following we show that $\Pr[E] \leq \mathsf{negl}(k)$. We define by E_i the event in which f learns

[13] Recall that our operations are over $\mathbf{GF}(2^{\mathrm{poly}(k)})$.

the entire z_i. Assuming the attacker reads n bits of the codeword, we have that for all $i \in [2|sk|]$,

$$\Pr_{\Sigma}[E_i] = \Pr_{\Sigma}[\ |I \cap Z_i| = m\ ] = \prod_{j=0}^{m-1} \frac{n-j}{\nu-j} \leq \left(\frac{n}{\nu}\right)^m.$$

We have $n = \alpha\nu$ and assuming $\alpha = 1 - \epsilon$ for $\epsilon \in (0, 1]$, we have

$$\Pr[E_i] \leq (1-\epsilon)^m \leq 1/e^{m\epsilon},$$

and

$$\Pr[E] = \Pr_{\Sigma}\left[\bigcup_{i=1}^{2|sk|} E_i\right] \leq \frac{2|sk|}{e^{m\epsilon}},$$

which is negligible when $(1-\alpha)m = \omega(\log(k))$, and the proof of the claim is complete. □

Claim 3.3.4. *Assuming* $(\mathsf{KGen}, \mathsf{E}, \mathsf{D})$ *is* 1-IND-CPA *secure, for any* $f \in \mathcal{F}^{\alpha}$ *and any message* s, $\mathsf{Exp}_1^{f,s} \approx \mathsf{Exp}_2^{f,s}$, *where the probability runs over the randomness used by* Init, Enc.

Proof. In $\mathsf{Exp}_2^{f,s}$ we unfold the encoding procedure, however instead of calling SS_m, we make a call to $\bar{\mathsf{SS}}_m^f$. As we have already stated above, this modification does not induce any difference between the output of $\mathsf{Exp}_2^{f,s}$ and $\mathsf{Exp}_1^{f,s}$, with overwhelming probability, as z^* is indistinguishable from z in the eyes of f. Another difference between the two experiments is in the external "Else" branch: $\mathsf{Exp}_1^{f,s}$ makes a call on the decoder while $\mathsf{Exp}_2^{f,s}$, before calling $\mathsf{D}_{sk}(\tilde{e})$, checks if the tampering function has modified the shares in a way such that the reconstruction procedure will give $\tilde{sk} \neq sk$ or $\tilde{sk}' \neq sk'$. This check is done by the statement "If $\exists i : \bigoplus_{j\in(I\cap Z_i)} c[j] \neq \bigoplus_{j\in(I\cap Z_i)} \tilde{c}[j]$", without touching sk or sk[3] (cf. Claim 3.3.3).[14] We define the events E, E' as follows

$$E\colon\ \mathsf{Dec}(\tilde{c}) \neq \perp,\ E'\colon\ \exists i : \bigoplus_{j\in(I\cap Z_i)} c[j] \neq \bigoplus_{j\in(I\cap Z_i)} \tilde{c}[j].$$

Clearly, conditioned on $\neg E'$ the two experiments are identical, since we have $\tilde{sk} = sk$ and $\tilde{sk}' = sk'$, and the decoding process will output $\mathsf{D}_{sk}(\tilde{e})$ in both experiments. Thus, the statistical distance is bounded by $\Pr[E']$. Now, conditioned on $E' \wedge \neg E$, both experiments output $\perp$. Thus, we need to bound $\Pr[E \wedge E']$. Assuming $\Pr[E \wedge E'] > p$, for $p = 1/\mathrm{poly}(k)$, we define an attacker $\mathcal{A}$ that simulates $\mathsf{Exp}_2^{f,s}$, and uses f, s to break the 1-IND-CPA security of $(\mathsf{KGen}, \mathsf{E}, \mathsf{D})$ in the presence of z^*, with probability at least $1/2 + p''/2$, for $p'' = 1/\mathrm{poly}(k)$.

[14] Recall that our operations are over $\mathbf{GF}(2^{\mathrm{poly}(k)})$.

First we prove that any 1-IND-CPA secure encryption scheme, remains secure even if the attacker receives $z^* \leftarrow \bar{\mathsf{SS}}^f_m(\Sigma, sk)$, as z^* consists of $m-1$ shares of each bit of sk and sk[3], i.e., for the entropy of sk we have $\mathbf{H}(sk|z^*) = \mathbf{H}(sk)$. Towards contradiction, assume there exists $\mathcal{A}$ that breaks the 1-IND-CPA security of $(\mathsf{KGen}, \mathsf{E}, \mathsf{D})$ in the presence of z^*, i.e., there exist s, s_0, s_1 such that $\mathcal{A}$ distinguishes between $(z^*, \mathsf{E}_{sk}(s), \mathsf{E}_{sk}(s_0))$ and $(z^*, \mathsf{E}_{sk}(s), \mathsf{E}_{sk}(s_1))$, with non-negligible probability p. We define an attacker $\mathcal{A}'$ that breaks the 1-IND-CPA security of $(\mathsf{KGen}, \mathsf{E}, \mathsf{D})$ as follows: $\mathcal{A}'$, given $(\mathsf{E}_{sk}(s), \mathsf{E}_{sk}(s_b))$, for some $b \in \{0,1\}$, samples $\hat{sk} \leftarrow \mathsf{KGen}(1^k)$, $\hat{z}^* \leftarrow \bar{\mathsf{SS}}^f_m(\Sigma, \hat{sk})$ and outputs $b' \leftarrow \mathcal{A}(z^*, \mathsf{E}_{sk}(s), \mathsf{E}_{sk}(s_b))$. Since $(z^*, \mathsf{E}_{sk}(s), \mathsf{E}_{sk}(s_b)) \approx (\hat{z}^*, \mathsf{E}_{sk}(s), \mathsf{E}_{sk}(s_b))$ the advantage of $\mathcal{A}'$ in breaking the 1-IND-CPA security of the scheme, is equal to the advantage of $\mathcal{A}$ in breaking the 1-IND-CPA security of the scheme in the presence of z^*, which by assumption is non-negligible, and this completes the current proof. We note that, the proof idea presented in the current paragraph also applies for proving that other properties that will be used in the rest of the proof, such as semantic security and authenticity, of the encryption scheme, are retained in the presence of z^*.

Now we prove our claim. Assuming $\Pr[E \wedge E'] > p$, for $p = 1/\text{poly}(k)$, we define an attacker $\mathcal{A}$ that breaks the 1-IND-CPA security of $(\mathsf{KGen}, \mathsf{E}, \mathsf{D})$ in the presence of z^*, with non-negligible probability. $\mathcal{A}$ receives the encryption of s, which corresponds to the oracle query right before receiving the challenge ciphertext, the challenge ciphertext $e \leftarrow \mathsf{E}_{sk}(s_b)$, for uniform $b \in \{0,1\}$ and uniform messages s_0, s_1, as well as z^*. $\mathcal{A}$ is defined below.

$\mathcal{A}\left(z^* \leftarrow \bar{\mathsf{SS}}^f_m(\Sigma, sk), e' \leftarrow \mathsf{E}_{sk}(s), e \leftarrow \mathsf{E}_{sk}(s_b)\right)$:

1. **(Define the shares that will be accessed by f)**: For $i \in [2|sk|]$, define $w_i := (z^*_i)_{|_{[m]\setminus\{l_i\}}}$ and for $i \in [m-1]$ define $C_i = ||_{j=1}^{|sk|} w_j[i]$, $D_i = ||_{j=|sk|+1}^{2|sk|} w_j[i]$.

2. **(Apply f)** Set $c \leftarrow P_\Sigma(z^*||e)$, compute $\tilde{c}[I] \leftarrow f_\Sigma(c_{|_I})$ and let $\tilde{C}_i$, $\tilde{D}_i$, $i \in [m]$, be the tampered shares resulting after the application of f to $c_{|_I}$.

3. **(Guessing the secret key)** Let $U = \sum_{i=1}^{m-1} C_i$, $V = \sum_{i=1}^{m-1} D_i$, i.e., U, V denote the sum of the shares that are being accessed by the attacker (maybe partially), and $\tilde{U} = \sum_{i=1}^{m-1} \tilde{C}_i$, $\tilde{V} = \sum_{i=1}^{m-1} \tilde{D}_i$, are the corresponding tampered values after applying f on U, V. Define

$$p(X) := (U - \tilde{U})X^2 + (U^2 - \tilde{U}^2)X + (U^3 - \tilde{U}^3 - V + \tilde{V}),$$

and compute the set of roots of $p(X)$, denoted as $\mathcal{X}$, which are at most two. Then set

$$\hat{\mathcal{SK}} := \{x + U | x \in \mathcal{X}\}\,. \tag{3.2}$$

4. **(Output)** Execute the following steps,

   a) For $\hat{sk} \in \hat{\mathcal{SK}}$, compute $s' \leftarrow \mathsf{D}_{\hat{sk}}(e')$, and if $s' = s$, compute $s'' \leftarrow \mathsf{D}_{\hat{sk}}(e)$. Return b' such that $s_{b'} = s''$.

   b) Otherwise, return $b' \leftarrow \{0,1\}$.

In the first step $\mathcal{A}$ removes the dummy symbol "$*$" and computes the shares that will be partially accessed by f, denoted as C_i for sk and as D_i for sk^3. In the second step, it simulates the codeword partially, applies the tampering function on it, and defines the tampered shares, $\tilde{C}_i$, $\tilde{D}_i$. Conditioned on E', it is not hard to see that $\mathcal{A}$ simulates perfectly $\mathsf{Exp}_2^{f,s}$. In particular, it simulates perfectly the input to f as it receives $e \leftarrow \mathsf{E}_{sk}(s)$ and all but $2|sk|$ of the actual bit-shares of sk, sk^3. Part of those shares will be accessed by f. Since for all i, $|I \cap Z_i| < m$, the attacker is not accessing any single bit of sk, sk^3. Let C_m, D_m, be the shares (not provided by the encryption oracle) that completely define sk and sk^3, respectively. By the definition of the encoding scheme and the fact that sk, $sk^3 \in \mathbf{GF}(2^{\mathrm{poly}(k)})$, we have $\sum_{i=1}^{m} C_i = sk$, $\sum_{i=1}^{m} D_i = sk^3$, and

$$(U + C_m)^3 = V + D_m. \tag{3.3}$$

In order for the decoder to output a non-bottom value, the shares created by the attacker must decode to $\tilde{sk}$, $\tilde{sk}'$, such that $\tilde{sk}^3 = \tilde{sk}'$, or in other words, if

$$\left(\tilde{U} + C_m\right)^3 = \tilde{V} + D_m. \tag{3.4}$$

From 3.3 and 3.4 we receive

$$(U - \tilde{U})C_m^2 + (U^2 - \tilde{U}^2)C_m + (U^3 - \tilde{U}^3) = V - \tilde{V}. \tag{3.5}$$

Clearly, $\Pr[E \wedge E' \wedge (U = \tilde{U})] = 0$. Thus, assuming $\Pr[E \wedge E'] > p$, for $p > 1/\mathrm{poly}(k)$, we receive

$$\begin{aligned} p < \Pr\left[E \wedge E' \wedge (U \neq \tilde{U})\right] &\leq \Pr\left[\mathsf{Dec}(\tilde{c}) \neq \perp \wedge E' \wedge U \neq \tilde{U}\right] \\ &\leq \Pr\left[\tilde{sk}^3 = \tilde{sk}' \wedge E' \wedge (U \neq \tilde{U})\right] \\ &\overset{(3.5,3.2)}{=} \Pr\left[C_m \in \mathcal{X}\right] \overset{(3.2)}{\leq} \Pr\left[sk \in \hat{\mathcal{SK}}\right], \end{aligned} \tag{3.6}$$

and $\mathcal{A}$ manages to recover C_m, and thus sk, with non-negligible probability $p' \geq p$. Let W be the event of breaking 1-IND-CPA security. Then,

$$\begin{aligned} \Pr[W] &= \Pr[W|sk \in \hat{\mathcal{SK}}] \cdot \Pr[sk \in \hat{\mathcal{SK}}] \\ &+ \Pr[W|sk \notin \hat{\mathcal{SK}}] \cdot \Pr[sk \notin \hat{\mathcal{SK}}] \\ &\overset{(3.6)}{=} p' + \frac{1}{2}(1-p') = \frac{1}{2} + \frac{p'}{2}, \end{aligned} \tag{3.7}$$

and the attacker breaks the IND-CPA security of $(\mathsf{KGen}, \mathsf{E}, \mathsf{D})$. Thus, we have $\Pr[E \wedge E'] \leq \mathsf{negl}(k)$, and both experiments output $\perp$ with overwhelming probability. □

Claim 3.3.5. *Assuming the authenticity property of* $(\mathsf{KGen}, \mathsf{E}, \mathsf{D})$*, for any* $f \in \mathcal{F}^{\alpha}$ *and any message* s*,* $\mathsf{Exp}_2^{f,s} \approx \mathsf{Exp}_3^{f,s}$*, where the probability runs over the randomness used by* Init*,* KGen *and* E*.*

Proof. Before proving the claim, recall that the authenticity property of the encryption scheme is preserved under the presence of z^* (cf. Claim 3.3.4). Let E be the event $\tilde{sk} = sk \wedge \tilde{sk}' = sk$[3] and E' be the event $\tilde{e} \neq e$. Conditioned on $\neg E$, the two experiments are identical, as they both output $\perp$. Also, conditioned on $E \wedge \neg E'$, both experiments output same^*. Thus, the statistical distance between the two experiments is bounded by $\Pr[E \wedge E']$. Let B be the event $\mathsf{D}_{sk}(\tilde{e}) \neq \perp$. Conditioned on $E \wedge E' \wedge \neg B$ both experiments output $\perp$. Thus, we need to bound $\Pr[E \wedge E' \wedge B]$.

Assuming there exist s, f, for which $\Pr[E \wedge E' \wedge B] > p$, where $p = 1/\mathrm{poly}(k)$, we define an attacker $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ that simulates $\mathsf{Exp}_3^{f,s}$ and breaks the authenticity property of the encryption scheme in the presence of z^*, with non-negligible probability. $\mathcal{A}$ is defined as follows: sample $(s, st) \leftarrow \mathcal{A}_1(1^k)$, and then, on input (z^*, e, st), where $e \leftarrow \mathsf{E}_{sk}(s)$, $\mathcal{A}_2$, samples $\Sigma \leftarrow \mathsf{Init}(1^k)$, sets $\tilde{c} \leftarrow 0^{\nu}$, $c \leftarrow P_{\Sigma}(z^*||e)$, computes $\tilde{c}[I] \leftarrow f(c_{|_I})$, $\tilde{c}[I^{\mathsf{c}}] \leftarrow c_{|_{I^{\mathsf{c}}}}$, $(\tilde{z}^*||\tilde{e}) \leftarrow P_{\Sigma}^{-1}(\tilde{c})$, and outputs $\tilde{e}$. Assuming $\Pr[E \wedge E' \wedge B] > p$, we have that $\mathsf{D}_{sk}(\tilde{e}) \neq \perp$ and $\tilde{e} \neq e$, with non-negligible probability and the authenticity property of $(\mathsf{KGen}, \mathsf{E}, \mathsf{D})$ breaks. □

Claim 3.3.6. *Assuming* $(\mathsf{KGen}, \mathsf{E}, \mathsf{D})$ *is semantically secure, for any* $f \in \mathcal{F}^{\alpha}$ *and any message* s*,* $\mathsf{Exp}_3^{f,s} \approx \mathsf{Exp}_3^{f,0}$ *, where the probability runs over the randomness used by* Init*,* KGen*,* E*. "*$\approx$*" may refer to statistical or computational indistinguishability, and* 0 *denotes the zero-message.*

Proof. Recall that $(\mathsf{KGen}, \mathsf{E}, \mathsf{D})$ is semantically secure even in the presence of $z^* \leftarrow \bar{\mathsf{SS}}_m^f(\Sigma, sk)$ (cf. 3.3.4), and towards contradiction, assume there exist $f \in \mathcal{F}^{\alpha}$, message s, and PPT

distinguisher D such that

$$\left|\Pr\left[\mathsf{D}\left(\Sigma, \mathsf{Exp}_3^{f,s}\right)=1\right]-\Pr\left[\mathsf{D}\left(\Sigma, \mathsf{Exp}_3^{f,0}\right)\right]=1\right|>p,$$

for $p=1/\text{poly}(k)$. We are going to define an attacker $\mathcal{A}$ that breaks the semantic security of $(\mathsf{KGen}, \mathsf{E}, \mathsf{D})$ in the presence of z^*, using $s_0 := s$, $s_1 := 0$. $\mathcal{A}$, given z^*, e, executes $\mathsf{Program}$.

$\mathsf{Program}(z^*, e)$:
$c \leftarrow P_\Sigma(z^*||e), \tilde{c} \leftarrow 0^\nu, \tilde{c}[I] \leftarrow f(c_{|_I})$
If $\exists i : |(I \cap Z_i)| = m$: $\tilde{s} \leftarrow \perp$
Else:
  If $\exists i : \bigoplus_{j\in(I\cap Z_i)} c[j] \neq \bigoplus_{j\in(I\cap Z_i)} \tilde{c}[j]$:
    $\tilde{s} \leftarrow \perp$
  Else: $\tilde{s} \leftarrow \perp$
    If $\tilde{e} = e$:
      $\tilde{s} \leftarrow \mathsf{same}^*$
Output $\tilde{s}$.

It is not hard to see that $\mathcal{A}$ simulates Exp_3^{f,s_b}, thus the advantage of $\mathcal{A}$ against the semantic security of $(\mathsf{KGen}, \mathsf{E}, \mathsf{D})$ is the same with the advantage of D in distinguishing between Exp_3^{f,s_0}, Exp_3^{f,s_1}, which by assumption is non-negligible. We have reached a contradiction and the proof of the claim is complete. □

From the above claims we have that for any $f \in \mathcal{F}^\alpha$ and any s, $\mathsf{Exp}_0^{f,s} \approx \mathsf{Exp}_3^{f,0}$, thus for any $f \in \mathcal{F}^\alpha$ and any s_0, s_1, $\mathsf{Exp}_0^{f,s_0} \approx \mathsf{Exp}_0^{f,s_1}$. Also, by the indistinguishability between $\mathsf{Exp}_0^{f,s}$ and $\mathsf{Exp}_3^{f,0}$, the second property of Lemma 3.2.2 has been proven as the output of $\mathsf{Exp}_3^{f,0}$ is in $\{s, \perp\}$, with overwhelming probability, and non-malleability with manipulation detection of our code follows by Lemma 3.2.2, since $\mathsf{Exp}_0^{f,s}$ is identical to Tamper_s^f of Lemma 3.2.2. □

On the CRS. In the above, the tampering function, and consequently the codeword locations that the function is given access to, are fixed before sampling the CRS and this is critical for achieving security. However, by the proof of Theorem 3.3.2, we observe that proving security in this setting is highly non-trivial. In addition, the tampering function receives full access to the CRS when tampering with the codeword, which is in contrast to the work by Faust et. al. [FMVW14] in the information-theoretic setting, where the (internal) tampering function receives partial information over the CRS.

In addition, the proposed scheme tolerates adaptive selection of the codeword locations, with respect to the CRS, in the following way: each time the attacker requests access to a location, he also learns if it corresponds to a bit of z or e, together with the index of that bit in the original string. In this way, the CRS is gradually disclosed to the adversary while picking codeword locations.

Finally, our CRS sustains a substantial amount of tampering that depends on the codeword locations chosen by the attacker: an attacker that gets access to a sensitive codeword bit is allowed to modify the part of the CRS that defines the location of that bit in the codeword. The attacker is allowed to modify all but $O(k \log(|s|+k))$ bits of the CRS, that is of length $O(k^2 \log k \log(|s|+k))$. To our knowledge, this is the first construction that tolerates, even partial modification of the CRS. In contrast, existing constructions in the CRS model are either using NIZKs [LL12, FN17, FMNV14, DKS18], or they are based on the *knowledge of exponent assumption* [KLT16], thus tampering access to the CRS would compromise security.

3.4 Removing the CRS

In the present section we show how to construct an MD-NMC for partial functions, in the standard model.

A first approach would be to store the CRS of Construction 3.3.1, inside the codeword together with $P_\Sigma(z||e)$, and give to the attacker read/write access to it. However, the tampering function, besides getting direct (partial) access to the encoding of sk, it also gets indirect access to it by (partially) controlling the CRS. Then, it can modify the CRS in a way such that, during decoding, ciphertext locations of its choice will be treated as bits of the inner encoding, z, increasing the tampering rate against z significantly. This makes the task of protecting sk hard, if not impossible (unless we restrict the access rate significantly).

To handle this challenge, we embed a structure recovering mechanism inside the codeword and we emulate the CRS effect by increasing the size of the alphabet, giving rise to a block-wise structure.[15] Notice that, non-malleable codes with large alphabet size (i.e., $\text{poly}(k) + |s|$ bits) might be easy to construct, as we can embed in each codeword block the verification key of a signature scheme together with a secret share of the message, as

[15]Bigger alphabets have been also considered in the context of error-correcting codes, in which the codeword consists of symbols.

well as a signature over the share. In this way, partial access over the codeword does not compromise the security of the signature scheme while the message remains private, and the simulation is straightforward. This approach however, comes with a large overhead, decreasing the information rate and access rate of the scheme significantly. In general, and similar to error correcting codes, we prefer smaller alphabet sizes – the larger the size is, the more coarse access structure is required, i.e., in order to access individual bits we need to access the blocks that contain them. The present thesis aims at minimizing this restriction by using small alphabets, as described below.

Our approach on the problem is the following. We increase the alphabet size to $O(\log k)$ bits, and we consider two types of blocks: (i) *sensitive blocks*, in which we store the inner encoding, z, of the secret key, sk, and (ii) *non-sensitive* blocks, in which we store the ciphertext, e, that is fragmented into blocks of size $O(\log k)$. The first bit of each block indicates whether it is a sensitive block, i.e., we set it to 1 for sensitive blocks and to 0, otherwise. Our encoder works as follows: on input message s, it computes z, e, as in the previous scheme and then uses rejection sampling to sample the indices, $\rho_1, \ldots, \rho_{|z|}$, for the sensitive blocks. Then, for every $i \in \{1, \ldots, |z|\}$, C_{ρ_i} is a sensitive block, with contents $(1||i||z[i])$, while the remaining blocks keep ciphertext pieces of size $O(\log k)$. Decoding proceeds as follows: on input codeword $C = (C_1, \ldots, C_{\mathsf{bn}})$, for each $i \in [\mathsf{bn}]$, if C_i is a non-sensitive block, its data will be part of e, otherwise, the last bit of C_i will be part of z, as it is dictated by the index stored in C_i. If the number of sensitive blocks is not the expected, the decoder outputs $\perp$, otherwise, z, e, have been fully recovered and decoding proceeds as in the previous scheme. The proposed scheme is depicted in Figure 3.3.

The security of our construction is based on the fact that, due to our shuffling technique, the position mapping will not be completely overwritten by the attacker, and we prove later in this section, this suffices for protecting the inner encoding over sk. We prove security of the current scheme (cf. Theorem 3.4.8) by a reduction to the security of the scheme in the CRS model. Our instantiation yields a rate $1 - 1/\Omega(\log k)$ MD-NMC in the standard model, with access rate $1 - 1/\Omega(\log k)$ and codewords of length $|s|(1 + 1/O(\log k)) + O(k^2 \log^2 k)$, assuming one-way functions.

It is worth pointing out that the idea of permuting blocks containing sensitive and non-sensitive data was also considered by [SS16] in the context of list-decodable codes, however the similarity is only in the fact that a permutation is being used at some point in the encoding process, and our objective, construction and proof are different.

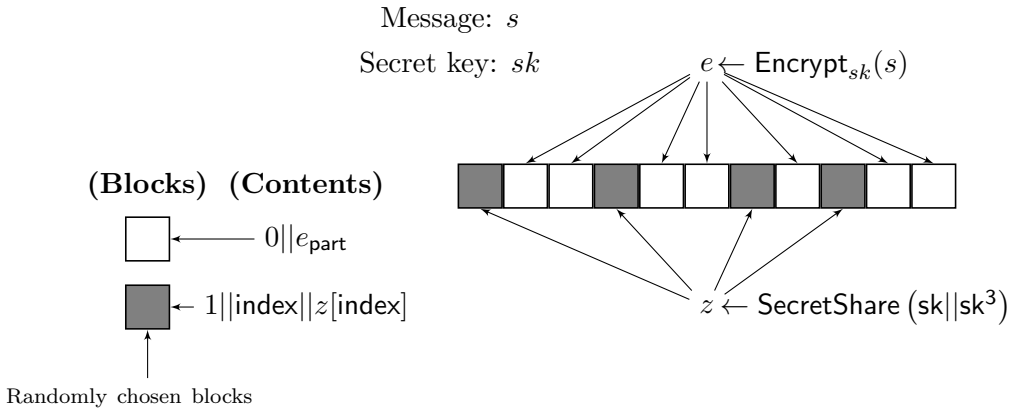

Figure 3.3: Description of the scheme in the standard model.

In what follows, we consider alphabets of size $O(\log(k))$ and we provide a computationally secure, rate $1 - 1/\Omega(\log k)$ encoding scheme in the standard model, tolerating modification of $(1 - o(1))\nu$ blocks, where ν is the total number of blocks in the codeword. The projection operation will be also used with respect to bigger alphabets, enabling the projection of blocks.

Our construction is defined below.

Construction 3.4.1. *Let k, $m \in \mathbb{N}$, let $(\mathsf{KGen}, \mathsf{E}, \mathsf{D})$ be a symmetric encryption scheme and $(\mathsf{SS}_m, \mathsf{Rec}_m)$ be an m-out-of-m secret sharing scheme. We define an encoding scheme $(\mathsf{Enc}^*, \mathsf{Dec}^*)$, as follows:*

- $\mathsf{Enc}^*(1^k, \cdot)$*: for input message s, sample $sk \leftarrow \mathsf{KGen}\left(1^k\right)$, $e \leftarrow \mathsf{E}_{sk}(s)$.*
  - **(Secret share)** *Sample $z \leftarrow \mathsf{SS}_m(sk||sk^3)$, where $z = ||_{i=1}^{2|sk|} z_i$, $z \in \{0,1\}^{2m|sk|}$, and for $i \in [|sk|]$, z_i (resp. $z_{|sk|+i}$) is an m-out-of-m secret sharing of $sk[i]$ (resp. $sk^3[i]$).*
  - **(Construct blocks & permute)** *Set $l \leftarrow 2m|sk|$, $\mathsf{bs} \leftarrow \log l + 2$, $d \leftarrow |e|/\mathsf{bs}$, $\mathsf{bn} \leftarrow l + d$, sample $\rho := (\rho_1, \ldots, \rho_l) \xleftarrow{\mathsf{rs}} \{0,1\}^{\log(\mathsf{bn})}$ and compute $C \leftarrow \Pi_\rho(z||e)$ as follows:*
    1. *Set $t \leftarrow 1$, $C_i \leftarrow 0^{\mathsf{bs}}$, $i \in [\mathsf{bn}]$.*
    2. **(Sensitive blocks)** *For $i \in [l]$, set $C_{\rho_i} \leftarrow (1||i||z[i])$.*

3. **(Ciphertext blocks)** *For* $i \in [\mathsf{bn}]$, *if* $i \neq \rho_j$, $j \in [l]$, $C_i \leftarrow (0||e[t : t + (\mathsf{bs} - 1)])$, $t \leftarrow t + (\mathsf{bs} - 1)$.[16]

Output $C := (C_1|| \ldots ||C_{\mathsf{bn}})$.

- $\mathsf{Dec}^*(1^k, \cdot)$*: on input* C, *parse it as* $(C_1|| \ldots ||C_{\mathsf{bn}})$, *set* $t \leftarrow 1$, $l \leftarrow 2m|sk|$, $z \leftarrow 0^l$, $e \leftarrow 0$, $\mathcal{L} = \emptyset$ *and compute* $(z||e) \leftarrow \Pi^{-1}(C)$ *as follows:*
  - *For* $i \in [\mathsf{bn}]$,
    - **(Sensitive block)** *If* $C_i[1] = 1$, *set* $j \leftarrow C_i[2 : \mathsf{bs} - 1]$, $z[j] \leftarrow C_i[\mathsf{bs}]$, $\mathcal{L} \leftarrow \mathcal{L} \cup \{j\}$.
    - **(Ciphertext block)** *Otherwise, set* $e[t : t + \mathsf{bs} - 1] = C_i[2 : \mathsf{bs}]$, $t \leftarrow t + \mathsf{bs} - 1$.
  - *If* $|\mathcal{L}| \neq l$, *output* $\perp$, *otherwise output* $(z||e)$.

  If $\Pi^{-1}(C) = \perp$, *output* $\perp$, *otherwise, compute* $(sk||sk') \leftarrow \mathsf{Rec}_m(z)$, *and if* $sk^3 = sk'$, *output* $\mathsf{D}_{sk}(e)$, *otherwise output* $\perp$.

The set of indices of the blocks in which z_i *is stored will be denoted by* Z_i.

We prove security for the above construction by a reduction to the security of Construction 3.3.1. We note that our reduction is non-black box with respect to the coding scheme in which security is reduced to; a generic reduction, i.e., non-malleable reduction [ADKO15], from the standard model to the CRS model is an interesting open problem and thus out of the scope of the present thesis.

In the following, we consider $\Gamma = \{0,1\}^{O(\log(k))}$.[17] The straightforward way to prove that $(\mathsf{Enc}^*, \mathsf{Dec}^*)$ is secure against $\mathcal{F}^{\alpha}_{\Gamma}$ by a reduction to the security of the bit-wise code of Section 3.3, would be as follows: for any $\alpha \in [0,1)$, $f \in \mathcal{F}^{\alpha}_{\Gamma}$ and any message s, we have to define α', $g \in \mathcal{F}^{\alpha'}$, such that the output of the tampered execution with respect to $(\mathsf{Enc}^*, \mathsf{Dec}^*)$, f, s, is indistinguishable from the tampered execution with respect to $(\mathsf{Init}, \mathsf{Enc}, \mathsf{Dec})$, g, s, and g is an admissible function for $(\mathsf{Init}, \mathsf{Enc}, \mathsf{Dec})$. However, this approach might be tricky as it requires the establishment of a relation between α and α' such that the sensitive blocks that f will receive access to, will be simulated using the sensitive bits accessed by g. Our approach is cleaner: for the needs of the current proof we leverage the power of Construction 3.3.1, by allowing the attacker to choose adaptively

[16] Here we assume that $\mathsf{bs} - 1$, divides the length of the ciphertext e. We can always achieve this property by padding the message s with zeros, if necessary.

[17] Recall that, whenever Γ is omitted from the notation, we assume that $\Gamma = \{0,1\}$.

the codeword locations, as long as it does not request to read all shares of the secret key. Then, for every block that is accessed by the block-wise attacker f, the bit-wise attacker g requests access to the locations of the bit-wise code that enable him to fully simulate the input to f. We formally present our ideas in the following sections. In Section 3.4.1 we introduce the function class $\mathcal{F}_{\mathsf{ad}}$ that considers adaptive adversaries with respect to the CRS and we prove security of Construction 3.3.1 in Corollary 3.4.3 against a subclass of $\mathcal{F}_{\mathsf{ad}}$, and then, we reduce the security of the block-wise code $(\mathsf{Enc}^*, \mathsf{Dec}^*)$ against $\mathcal{F}_{\Gamma}^{\alpha}$ to the security of Construction 3.3.1 against $\mathcal{F}_{\mathsf{ad}}$ (cf. Section 3.4.2).

3.4.1 Security against adaptive adversaries

In the current section we prove that Construction 3.3.1 is secure against the class of functions that request access to the codeword adaptively, i.e., depending on the CRS, as long as they access a bounded number of sensitive bits. Below, we formally define the function class $\mathcal{F}_{\mathsf{ad}}$, in which the tampering function picks up the codeword locations depending on the CRS, and we consider $\Gamma = \{0,1\}$.

Definition 3.4.2 (The function class $\mathcal{F}_{\mathsf{ad}}^{\nu}$ (or $\mathcal{F}_{\mathsf{ad}}$))**.** *Let* $(\mathsf{Init}, \mathsf{Enc}, \mathsf{Dec})$ *be an* (κ, ν)*-coding scheme and let* $\mathbb{\Sigma}$ *be the range of* $\mathsf{Init}(1^k)$*. For any* $g = (g_1, g_2) \in \mathcal{F}_{\mathsf{ad}}^{\nu}$*, we have* $g_1 : \mathbb{\Sigma} \to \mathcal{P}([\nu])$*,* $g_2^{\Sigma} : \{0,1\}^{|\mathsf{range}(g_1)|} \to \{0,1\}^{|\mathsf{range}(g_1)|} \cup \{\perp\}$*, and for any* $c \in \{0,1\}^{\nu}$*,* $g^{\Sigma}(c) = g_2\left(c_{|_{g_1(\Sigma)}}\right)$*. For brevity, the function class will be denoted as* $\mathcal{F}_{\mathsf{ad}}$*.*

Construction 3.3.1 remains secure against functions that receive full access to the ciphertext, as long as they request to read all but one shares for each bit of sk and sk^3. The result is formally presented in the following corollary.

Corollary 3.4.3. *Let* k*,* $m \in \mathbb{N}$*. Assuming* $(\mathsf{SS}_m, \mathsf{Rec}_m)$ *is an* m*-out-of-*m *secret sharing scheme and* $(\mathsf{KGen}, \mathsf{E}, \mathsf{D})$ *is* 1-IND-CPA *secure authenticated encryption scheme, the code of Construction 3.4.1 is an* MD-NMC *against any* $g = (g_1, g_2) \in \mathcal{F}_{\mathsf{ad}}$*, assuming that for all* $i \in [2|sk|]$*,* $(Z_i \cap g_1(\Sigma)) < m$*, where* $sk \leftarrow \mathsf{KGen}(1^k)$ *and* $\Sigma \leftarrow \mathsf{Init}(1^k)$*.*

Proof. Let $g = (g_1, g_2)$ be as stated above. For any message s, the tampered execution with respect to g and $(\mathsf{Init}, \mathsf{Enc}, \mathsf{Dec})$, is defined as follows.

$$\mathsf{Tamper}_s^g := \left\{ \begin{array}{l} \Sigma \leftarrow \mathsf{Init}(1^k), c \leftarrow \mathsf{Enc}(\Sigma, s), I \leftarrow g_1(\Sigma) \\ \tilde{c} \leftarrow g_2^{\Sigma}(c_{|_I}), \tilde{s} \leftarrow \mathsf{Dec}(\Sigma, \tilde{c}) \\ \\ \text{If } \tilde{s} = s, \text{ output } \mathsf{same}^*, \text{ otherwise, output } \tilde{s}. \end{array} \right\}$$

The proof is along the lines of the proof of Theorem 3.3.2, i.e., we prove that for any g having the properties stated above, and any pair of messages s_0, s_1, $\mathsf{Tamper}^g_{s_0} \approx \mathsf{Tamper}^g_{s_1}$, and the output of the tampered execution is either the original message, or $\perp$, with overwhelming probability. Below, we revisit the hybrids of Theorem 3.3.2 and we prove that the indistinguishability between adjacent hybrids, holds with respect to g.

- $\mathsf{Exp}^{g,s}_0$: For any f, s, the first experiment, $\mathsf{Exp}^{g,s}_0$, is identical to the experiment Tamper^g_s.

- $\mathsf{Exp}^{g,s}_1$: In the second experiment we have Z_i, $i \in [2|sk|]$, to be the set of indices in which z_i is stored, $|Z_i| = m$. The main difference from the previous experiment is that the current one outputs $\perp$, if there exists a bit of sk or sk^3 for which the tampering function reads all shares of it. However, by the definition of g we know that this happens with zero probability, thus we have that the following claim holds,

  Claim 3.4.4. *Let k, $m \in \mathbb{N}$. For any $g = (g_1, g_2) \in \mathcal{F}_{\mathsf{ad}}$, assuming that for all $i \in [2|sk|]$, $(Z_i \cap g_1(\Sigma)) < m$ and any message s, we have $\mathsf{Exp}^{g,s}_0 = \mathsf{Exp}^{g,s}_1$, where $sk \leftarrow \mathsf{KGen}(1^k)$, $\Sigma \leftarrow \mathsf{Init}(1^k)$.*

- $\mathsf{Exp}^{g,s}_2$: In the current experiment we unfold the encoding procedure, and in addition, we substitute the secret sharing procedure SS_m with $\bar{\mathsf{SS}}^g_m$, where $\bar{\mathsf{SS}}^g_m$ is defined as $\bar{\mathsf{SS}}^f_m$ does with respect to f, in Claim 3.3.4 of Theorem 3.3.2. From the above claim we have that for all i, $|I \cap Z_i| < m$, and we observe that the current experiment is identical to the previous one up to the point of computing $g(c_{|_I})$, as $c_{|_I}$ carries no information about sk and sk^3. Thus, the transition between the current experiment and the previous one is identical to that of Theorem 3.3.2: an attacker that partially modifies the shares of sk and sk^3, creates shares of $\tilde{sk}$ and $\tilde{sk}'$, such that $\tilde{sk}^3 = \tilde{sk}'$, with negligible probability in k, which is proved by a reduction to the 1-IND-CPA security of the encryption scheme in the presence of z^*. Thus, we have the following claim.

  Claim 3.4.5. *Assuming $(\mathsf{KGen}, \mathsf{E}, \mathsf{D})$ is* 1-IND-CPA *secure (cf. Definition 2.1.6), for any $g \in \mathcal{F}_{\mathsf{ad}}$ and any message s, $\mathsf{Exp}^{g,s}_1 \approx \mathsf{Exp}^{g,s}_2$, where the probability runs over the randomness used by* Init, Enc.

- $\mathsf{Exp}^{g,s}_3$: As in Theorem 3.3.2, the indistinguishability between the two experiments follows from the authenticity property of the encryption scheme in the presence of z^*. Thus, the following holds.

Claim 3.4.6. *Assuming the authenticity property of* $(\mathsf{KGen}, \mathsf{E}, \mathsf{D})$*, for any* $g \in \mathcal{F}_{\mathsf{ad}}$ *and any message* s*,* $\mathsf{Exp}_2^{f,s} \approx \mathsf{Exp}_3^{f,s}$*, where the probability runs over the randomness used by* Init*,* KGen *and* E*.*

- Finally, since g learns nothing about sk, we have that for any $g \in \mathcal{F}_{\mathsf{ad}}$, and message s, $\mathsf{Exp}_3^{g,s}$ is indistinguishable from $\mathsf{Exp}_3^{g,0}$, where 0 denotes the zero-message. This follows by the semantic security of the encryption scheme (Definition 2.1.6). Formally, we prove the following claim.

  Claim 3.4.7. *Assuming* $(\mathsf{KGen}, \mathsf{E}, \mathsf{D})$ *is semantically secure, for any* $g \in \mathcal{F}_{\mathsf{ad}}$ *and any message* s*,* $\mathsf{Exp}_3^{g,s} \approx \mathsf{Exp}_3^{g,0}$ *, where the probability runs over the randomness used by* Init*,* KGen*,* E*, "*$\approx$*" may refer to statistical or computational indistinguishability, and* 0 *is the zero-message.*

From the above claims we have that for any $g \in \mathcal{F}_{\mathsf{ad}}$ and any s_0, s_1, assuming that for all $i \in [2|sk|]$, $(Z_i \cap g_1(\Sigma)) < m$, $\mathsf{Exp}_0^{g,s_0} \approx \mathsf{Exp}_0^{g,s_1}$, and non-malleability with manipulation detection follows by Lemma 3.2.2, since $\mathsf{Exp}_0^{g,s}$ is identical to Tamper_s^g of Lemma 3.2.2, and by the indistinguishability between $\mathsf{Exp}_0^{g,s}$ and $\mathsf{Exp}_3^{g,s}$, the second property of Lemma 3.2.2 has been proven as the output of $\mathsf{Exp}_3^{g,s}$ is in $\{s, \perp\}$, with overwhelming probability. □

3.4.2 MD-NMC security of the block-wise code

In the current section we prove security of Construction 3.4.1 against $\mathcal{F}_\Gamma^\alpha$, for $\Gamma = \{0,1\}^{O(\log(k))}$.

Theorem 3.4.8. *Let* k*,* $m \in \mathbb{N}$*,* $\Gamma = \{0,1\}^{O(\log(k))}$ *and* $\alpha \in [0,1)$*. Assuming* $(\mathsf{SS}_m, \mathsf{Rec}_m)$ *is an* m*-out-of-*m *secret sharing scheme and* $(\mathsf{KGen}, \mathsf{E}, \mathsf{D})$ *is a* 1-IND-CPA *secure authenticated encryption scheme, the code of Construction 3.4.1 is an* MD-NMC *against* $\mathcal{F}_\Gamma^\alpha$*, for any* α*,* m*, such that* $(1-\alpha)m = \omega(\log(k))$*.*

Proof. Following Lemma 3.2.2, we prove that for any $f \in \mathcal{F}_\Gamma^\alpha$, and any pair of messages s_0, s_1, $\mathsf{Tamper}_{s_0}^f \approx \mathsf{Tamper}_{s_1}^f$, and for any s, $\Pr\left[\mathsf{Tamper}_s^f \notin \{\perp, s\}\right] \leq \mathsf{negl}(k)$, where Tamper denotes the experiment defined in Lemma 3.2.2 with respect to the encoding scheme of Construction 3.4.1, $(\mathsf{Enc}^*, \mathsf{Dec}^*)$. Our proof is given by a series of hybrids depicted in Figure 3.7. We reduce the security $(\mathsf{Enc}^*, \mathsf{Dec}^*)$, to the security of Construction 3.3.1, $(\mathsf{Init}, \mathsf{Enc}, \mathsf{Dec})$, against $\mathcal{F}_{\mathsf{ad}}$ (cf. Corollary 3.4.3). The idea is to move from the tampered execution with respect to $(\mathsf{Enc}^*, \mathsf{Dec}^*)$, f, to a tampered execution with respect to $(\mathsf{Init}, \mathsf{Enc}, \mathsf{Dec})$, g, such that the two executions are indistinguishable and $(\mathsf{Init}, \mathsf{Enc}, \mathsf{Dec})$ is secure against g.

Let I_{b} be the set of indices of the blocks that f chooses to tamper with, where $|I_{\mathsf{b}}| \leq \alpha\nu$, and let $l \leftarrow 2m|sk|$, $\mathsf{bs} \leftarrow \log l + 2$, $\mathsf{bn} \leftarrow l + |e|/\mathsf{bs}$. Below we describe the hybrids of Figure 3.7.

- $\mathsf{Exp}_0^{f,s}$: The current experiment is the experiment Tamper_s^f, of Lemma 3.2.2, with respect to $(\mathsf{Enc}^*, \mathsf{Dec}^*)$, f, s.
- $\mathsf{Exp}_1^{(g_1,g_2),s}$: The main difference between $\mathsf{Exp}_0^{f,s}$ and $\mathsf{Exp}_1^{(g_1,g_2),s}$, is that in the latter one, we introduce the tampering function (g_1, g_2), that operates over codewords of $(\mathsf{Init}, \mathsf{Enc}, \mathsf{Dec})$ and we modify the encoding steps so that the experiment creates codewords of the bit-wise code $(\mathsf{Init}, \mathsf{Enc}, \mathsf{Dec})$. (g_1, g_2) simulates partially the block-wise codeword C, while given partial access to the bit-wise codeword $c \leftarrow \mathsf{Enc}(s)$. As we prove in Claim 3.4.9, it simulates perfectly the tampering effect of f against $C \leftarrow \mathsf{Enc}^*(s)$.

  g_1 operates as follows (cf. Figure 3.4): it simulates perfectly the randomness for the permutation of the block-wise code, denoted as ρ, and constructs a set of indices I, such that g_2 will receive access to, and tamper with, $c_{|_I}$. The set I is constructed

$g_1(\Sigma = (r_1, \ldots, r_l))$:

- **(Simulate block shuffling)**:
  Sample $\rho := (\rho_1, \ldots, \rho_l) \xleftarrow{\mathsf{rs}} \{0,1\}^{\log(\mathsf{bn})}$
- **(Construct I)**: Set $I = \emptyset$,
  - **(Add ciphertext locations to I)**:
    For $j \in [|e| + l]$, if $j \notin \{r_i | i \in [l]\}$, $I \leftarrow (I \cup j)$.
  - **(Add sensitive bit locations to I according to I_{b})**:
    For $j \in [\mathsf{bn}]$, if $j \in I_{\mathsf{b}}$ and $\exists i \in [l]$ such that $j = \rho_i$, $I \leftarrow (I \cup r_i)$.
- **Output**: Output I.

Figure 3.4: The function g_1 that appears in the hybrid experiments of Figure 3.7.

  with respect to the set of blocks I_{b}, that f chooses to access, as well as Σ, that reveals the original bit positions, i.e., the ones before permuting $(z||e)$. g_2 receives $c_{|_I}$, reconstructs I, simulates partially the blocks of the block-wise codeword, C, and applies f on the simulated codeword. The program of g_2 is given in Figure 3.5. In Claim 3.4.9, we show that g_2, given $c_{|_I}$, simulates perfectly $C_{|_{I_{\mathsf{b}}}}$, which implies that $g_2^{\Sigma}(c_{|_I}) = f(C_{|_{I_{\mathsf{b}}}})$, and the two executions are identical.

$g_2^{\Sigma}(c_{|_I})$:

$t \leftarrow 1$, $C_i^* \leftarrow 0^{\mathsf{bs}}$, $i \in [\mathsf{bn}]$.

- **(Reconstruct I)**: Compute $I \leftarrow g_1(\Sigma)$.
- **(Simulate ciphertext blocks)**:
  For $i \in [\mathsf{bn}]$, if $i \neq \rho_j$ for $j \in [l]$, $C_i^* \leftarrow (0||e[t : t + (\mathsf{bs} - 1)])$, $t \leftarrow t + (\mathsf{bs} - 1)$.
- **(Simulate sensitive blocks)**:
  - For $i \in [|I|]$, if $\exists j \in [l]$, such that $\mathsf{Ind}(c_{|_I}[i]) = r_j$, set $C^*_{\rho_j} \leftarrow \big(1||j||c_{|_I}[i]\big)$.
  - Set $C^* := (C_1^*|| \dots ||C^*_{\mathsf{bn}})$ and $\tilde{C}^* := C^*$.
- **(Apply f)**: compute $\tilde{C}^*[I_{\mathsf{b}}] \leftarrow f(C^*_{|_{I_{\mathsf{b}}}})$.
- **(Output)**: Output $\tilde{C}^*_{|_{I_{\mathsf{b}}}}$.

Figure 3.5: The function g_2 that appears in the hybrid experiments of Figure 3.7.

- $\mathsf{Exp}_2^{(g_1,g_3),s}$: In the current experiment, we substitute the function g_2 with g_3, and Dec^* with Dec, respectively. By inspecting the code of g_2 and g_3 (cf. Figures 3.5,3.6, respectively), we observe that latter function executes the code of the former, plus the "Check labels and simulate $\tilde{c}[I]$" step. Thus the two experiments are identical up to the point of computing $f(C^*_{|_{I_{\mathsf{b}}}})$. The main idea here is that we want the current

$g_3^{\Sigma}(c_{|_I})$:

$t \leftarrow 1$, $C_i^* \leftarrow 0^{\mathsf{bs}}$, $i \in [\mathsf{bn}]$.

- **(Reconstruct I)**: Compute $I \leftarrow g_1(\Sigma)$.
- **(Simulate ciphertext blocks)**:
  For $i \in [\mathsf{bn}]$, if $i \neq \rho_j$, $j \in [l]$, $C_i^* \leftarrow (0||e[t : t + (\mathsf{bs} - 1)])$, $t \leftarrow t + (\mathsf{bs} - 1)$.
- **(Simulate sensitive blocks)**:
  - For $i \in [|I|]$, if $\exists j \in [l]$, such that $\mathsf{Ind}(c_{|_I}[i]) = r_j$, set $C^*_{\rho_j} \leftarrow \big(1||j||c_{|_I}[i]\big)$.
  - Set $C^* := (C_1^*|| \dots ||C^*_{\mathsf{bn}})$ and $\tilde{C}^* := C^*$.
- **(Apply f)**: compute $\tilde{C}^*[I_{\mathsf{b}}] \leftarrow f(C^*_{|_{I_{\mathsf{b}}}})$.
- **(Check labels and simulate $\tilde{c}[I]$)**: For $i \in \{\rho_j | j \in [l]\} \backslash I_{\mathsf{b}}$, $C_i^* \leftarrow (1||i||0)$. If $\Pi^{-1}(\tilde{C}^*) = \perp$, set $d \leftarrow 1$, otherwise set $(\tilde{z}^*||\tilde{e}) \leftarrow \Pi^{-1}(\tilde{C}^*)$, $\tilde{c}^* \leftarrow P_{\Sigma}(\tilde{z}^*||\tilde{e})$.
- **(Output)**: If $d = 1$ output $\perp$, otherwise output $\tilde{c}^*_{|_I}$.

Figure 3.6: The function g_3 that appears in the hybrid experiments of Figure 3.7.

execution to be with respect to $(\mathsf{Init}, \mathsf{Enc}, \mathsf{Dec})$ against (g_1, g_3). Thus, we substitute Dec^* with Dec, and we expand the function g_2 with some extra instructions/checks that are missing from Dec. We name the resulting function as g_3 and we prove that the two executions are identical.

- Finally, we prove that for any f and any s,

$$\Pr\left[\mathsf{Exp}_2^{(g_1,g_3),s} \notin \{\perp, s\}\right] \leq \mathsf{negl}(k).$$

We do so by proving that (g_1, g_3) is admissible for $(\mathsf{Init}, \mathsf{Enc}, \mathsf{Dec},)$, i.e., $(g_1, g_3) \in \mathcal{F}_{\mathsf{ad}}$, and g_3 will not request to access more that $m-1$ shares for each bit of sk, sk^3 (cf. Corollary 3.4.3). This implies security according to Lemma 3.2.2.

In what follows we prove indistinguishability between the hybrids.

Claim 3.4.9. *For any $f \in \mathcal{F}_\Gamma^\alpha$ and any s, $\mathsf{Exp}_0^{f,s} = \mathsf{Exp}_1^{(g_1,g_2),s}$.*[18]

Proof. The main difference between Exp_0 and Exp_1, is that in Exp_1, we introduce the tampering function $g = (g_1, g_2)$, that operates over codewords of $(\mathsf{Init}, \mathsf{Enc}, \mathsf{Dec})$, and simulates partially the block-wise code. We observe that g_1 simulates perfectly the randomness of the permutation for the block-wise code, denoted as ρ. Thus, the computation $C \leftarrow \Pi_\rho(z||e)$ does not induce any statistical difference between the two experiments. By the definition of g_1 we have that $c_{|_I}$ consists of all ciphertext bits, as well as the indices r_i, for which $\rho_i \in I_{\mathsf{b}}$, $i \in [l]$, i.e., if f requests access to the sensitive block with index ρ_i, containing $z[i]$, g_1 will request access to the r_i-th bit of c, which is $z[i]$. Thus, g_2 will receive as input the entire ciphertext and all the sensitive bits that f will request access to, with respect to I_{b}, thus it can fully simulate $C_{|_{I_{\mathsf{b}}}}$ while being consistent with the distribution of blocks in $C_{|_{I_{\mathsf{b}}^{\mathsf{c}}}}$, as ρ is generated by g_1. Thus we have that $g_2^{\Sigma}(c_{|_I})$ is identical to $f(C_{|_{I_{\mathsf{b}}}})$, and the proof of the claim is complete. □

Claim 3.4.10. *For any $f \in \mathcal{F}_\Gamma^\alpha$ and any s, $\mathsf{Exp}_1^{(g_1,g_2),s} = \mathsf{Exp}_2^{(g_1,g_3),s}$.*

Proof. In Exp_2 we substitute the function g_2 with g_3, and Dec^* with Dec, respectively. By inspecting the code of g_2 and g_3, we observe that latter function executes the code of the former, plus the "Check labels and simulate $\tilde{c}[I]$" step. Thus the two experiments are identical up to the point of computing $f(C_{|_{I_{\mathsf{b}}}})$. We unfold the code of the two experiments from that point of the computation and on (cf. Figure 3.8). They idea is that the consis-

[18] For random variables X, Y, $X = Y$ denotes that the random variables are identical.

$\mathsf{Exp}_0^{f,s}$:
$sk \leftarrow \mathsf{KGen}\left(1^k\right), e \leftarrow \mathsf{E}_{sk}(s)$
$z \leftarrow \mathsf{SS}_m(sk||sk^3)$

$\rho := (\rho_1, \ldots, \rho_l) \xleftarrow{\mathsf{rs}} \{0,1\}^{\log(\mathsf{bn})}$
$C \leftarrow \Pi_\rho(z||e), \tilde{C} \leftarrow C$
$\tilde{C}[I_\mathsf{b}] \leftarrow f(C_{|_{I_\mathsf{b}}})$

$\tilde{s} \leftarrow \mathsf{Dec}^*(\tilde{C})$

Output same^* if $\tilde{s} = s$ and $\tilde{s}$ otherwise.

$\mathsf{Exp}_1^{(g_1,g_2),s}$:
$sk \leftarrow \mathsf{KGen}\left(1^k\right), e \leftarrow \mathsf{E}_{sk}(s)$
$z \leftarrow \mathsf{SS}_m(sk||sk^3)$

$\Sigma \leftarrow \mathsf{Init}(1^k), c \leftarrow P_\Sigma(z||e)$
$I \leftarrow g_1(\Sigma)$
$C \leftarrow \Pi_\rho(z||e), \tilde{C} \leftarrow C$
$\tilde{C}[I_\mathsf{b}] \leftarrow g_2^\Sigma(c_{|_I})$

$\tilde{s} \leftarrow \mathsf{Dec}^*(\tilde{C})$

Output same^* if $\tilde{s} = s$ and $\tilde{s}$ otherwise.

$\mathsf{Exp}_2^{(g_1,g_3),s}$:
$\Sigma \leftarrow \mathsf{Init}(1^k)$
$sk \leftarrow \mathsf{KGen}\left(1^k\right), e \leftarrow \mathsf{E}_{sk}(s)$
$z \leftarrow \mathsf{SS}_m(sk||sk^3)$

$c \leftarrow P_\Sigma(z||e), \tilde{c} \leftarrow c$
$I \leftarrow g_1(\Sigma)$
$\tilde{c}[I] \leftarrow g_3^\Sigma(c_{|_I})$

$\tilde{s} \leftarrow \mathsf{Dec}(\Sigma, \tilde{c})$

Output same^* if $\tilde{s} = s$ and $\tilde{s}$ otherwise.

Figure 3.7: The hybrid experiments for the proof of Theorem 3.4.8.

tency check on the labels of the block-wise code is transferred from Dec^* in Exp_1 to g_3 in Exp_2, and Dec^* is substituted by Dec, so that $\mathsf{Exp}_2^{(g_1,g_3),s}$ is the tampering experiment of Lemma 3.2.2 with respect to $(\mathsf{Init}, \mathsf{Enc}, \mathsf{Dec})$ and (g_1, g_3).

In order to show that $\mathsf{Exp}_1^{(g_1,g_2),s} = \mathsf{Exp}_2^{(g_1,g_3),s}$, is suffices to prove that $\mathsf{Dec}^*(\tilde{C}) = \mathsf{Dec}(\tilde{c})$. By inspecting $\mathsf{Exp}_2^{(g_1,g_3),s}$, we have that $\tilde{c} = \perp$ if and only if $\Pi^{-1}(\tilde{C}^*) = \perp$. By the definition of Π^{-1} (cf. Construction 3.4.1), $\Pi^{-1}(\tilde{C}^*) = \perp$, if and only if the tampering function creates an inconsistent set of labels, an effect that can be decided by g_3 by only partially accessing C, since it fully simulates the labels for the block-wise code. By Claim

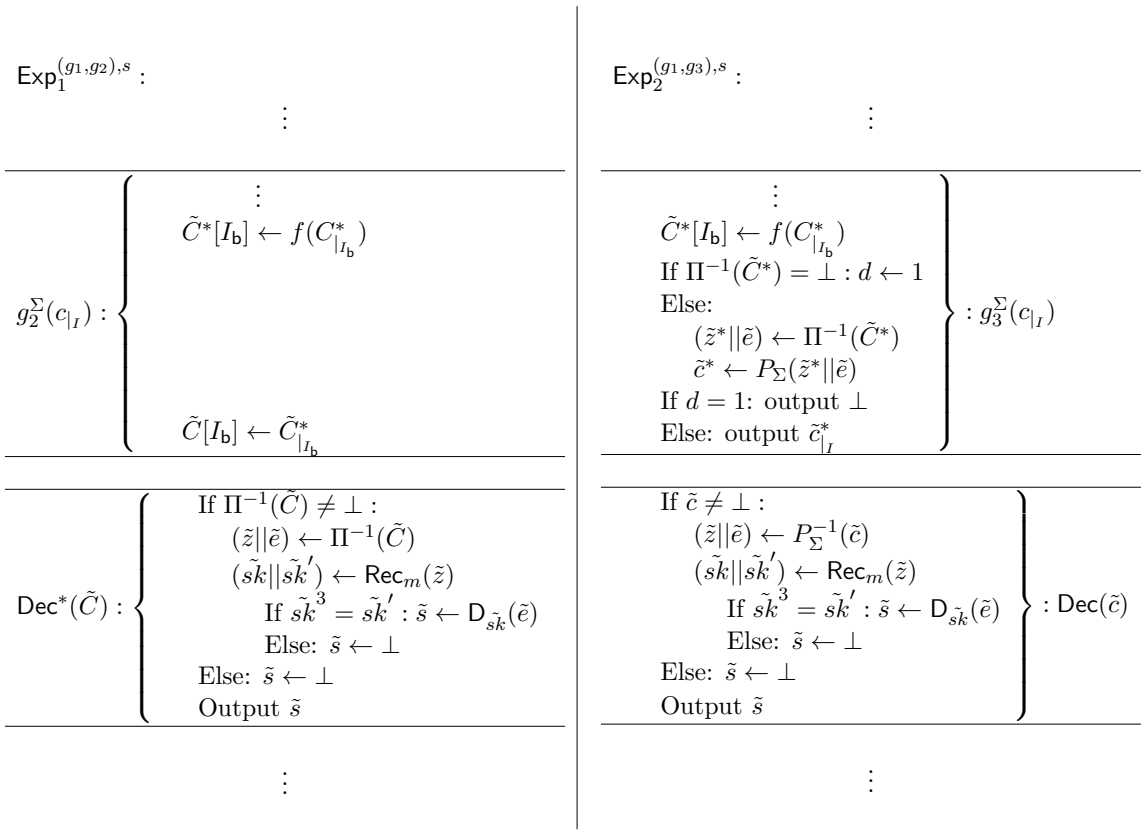

Figure 3.8: The unfolded code of Exp_1 and Exp_2.

3.4.9, $C_{|_{I_{\mathsf{b}}}} = C^*_{|_{I_{\mathsf{b}}}}$ and thus $\tilde{C}_{|_{I_{\mathsf{b}}}} = \tilde{C}^*_{|_{I_{\mathsf{b}}}}$, which implies that $\Pi^{-1}(\tilde{C}^*) = \perp$ if and only if $\Pi^{-1}(\tilde{C}) = \perp$. We conclude that $\tilde{c} = \perp$ if and only if $\Pi^{-1}(\tilde{C}) = \perp$. Let E be the event in which $\tilde{c} \neq \perp$. Clearly, conditioned on $\neg E$ the two experiments are identical, as both output $\perp$. It remains to prove the same conditioned on E.

By inspecting the two experiments, and conditioned on E, we have

$$\tilde{c}_{|_I} = \tilde{c}^*_{|_I} = \left[P_\Sigma\left(\Pi^{-1}(\tilde{C}^*)\right)\right]_{|_I} = \left[P_\Sigma\left(\Pi^{-1}(\tilde{C})\right)\right]_{|_I}, \tag{3.8}$$

where the last equality follows from the fact that $\left[P_\Sigma\left(\Pi^{-1}(\tilde{C}^*)\right)\right]_{|_I}$ is independent of the blocks of $\tilde{C}$ that Exp_2 does not have access to. Moreover,

$$\tilde{c}_{|_{I^{\mathsf{c}}}} = c_{|_{I^{\mathsf{c}}}} = \left[P_\Sigma\left(\Pi^{-1}(C)\right)\right]_{|_{I^{\mathsf{c}}}} = \left[P_\Sigma\left(\Pi^{-1}(\tilde{C})\right)\right]_{|_{I^{\mathsf{c}}}}, \tag{3.9}$$

where the last equality follows from the fact that $\tilde{c}_{|_{I^{\mathsf{c}}}}$ is not being accessed by the tampering function. From the above relations we have that $\tilde{c} = P_\Sigma\left(\Pi^{-1}(\tilde{C})\right)$, thus $P_\Sigma^{-1}(\tilde{c}) = \Pi^{-1}(\tilde{C})$, and the two executions are identical conditioned on E. □

Claim 3.4.11. *Assuming* $(1-\alpha)m = \omega(\log(k))$*, for any* $f \in \mathcal{F}_\Gamma^\alpha$ *and any* s*,*

$$\Pr\left[\mathsf{Exp}_2^{(g_1,g_3),s} \notin \{\perp, s\}\right] \leq \mathsf{negl}(k),$$

over the randomness of Exp_2*.*

Proof. Assuming $(1-\alpha)m = \omega(\log(k))$, it suffices to prove that for any $f \in \mathcal{F}_\Gamma^\alpha$, the function $(g_1, g_3) \in \mathcal{F}_{\mathsf{ad}}$ is admissible for $(\mathsf{Init}, \mathsf{Enc}, \mathsf{Dec},)$, i.e., g_1 will not request to access more that $m-1$ shares for each bit of sk, sk^3, and the proof of the claim will follow by Corollary 3.4.3 and Lemma 3.2.2. We prove that for any $f \in \mathcal{F}_\Gamma^\alpha$, the corresponding (g_1, g_3) will not access the entire z_i, for all $i \in [|2sk|]$, with overwhelming probability. Such an event takes place if and only if $\exists i : |(I_{\mathsf{b}} \cap Z_i)| = m$. We define by E_i the event in which f request access to all blocks in which z_i is stored. Assuming f reads n blocks, we have that for all $i \in [2|sk|]$,

$$\Pr_\rho[E_i] = \Pr_\rho[\ |I_{\mathsf{b}} \cap Z_i| = m\ ] = \prod_{j=0}^{m-1} \frac{n-j}{\nu-j} \leq \left(\frac{n}{\nu}\right)^m.$$

We have $n = \alpha\nu$ and assuming $\alpha = 1-\epsilon$ for $\epsilon \in (0,1]$, we have $\Pr[E_i] \leq (1-\epsilon)^m \leq 1/e^{m\epsilon}$ and

$$\Pr[E] = \Pr_\rho\left[\bigcup_{i=1}^{2|sk|} E_i\right] \leq \frac{2|sk|}{e^{m\epsilon}},$$

which is negligible when $(1-\alpha)m = \omega(\log(k))$. □

The security of the block-wise code follows from the above claims and the MD-NMC security of $(\mathsf{Init}, \mathsf{Enc}, \mathsf{Dec})$. □

3.5 Continuous MD-NMC with light updates

In this section, we enhance the block-wise scheme of Section 3.4 with an update mechanism, that uses only shuffling and refreshing operations. The resulting code is secure against continuous attacks, for a notion of security that is weaker than the original one [FMVW14], as we need to update the codeword after each round of execution. However, our update mechanism is using cheap operations, avoiding the full decoding and re-encoding of the message, which is the standard way to achieve continuous security [LL12, DPW10] using a one-time NMC. In addition, our solution avoids the usage of a self-destruction mechanism that produces $\perp$ in all subsequent rounds after the first round in which the attacker creates an invalid codeword, which was originally proposed by [FN17]. Avoiding

the self-destruction mechanism, was originally proposed by [FN17], and it is an important step towards practicality, as (i) the mechanism is subjective to denial of service attacks, and (ii) it renders the device useless in the presence of non-adversarial hardware faults. Our solution enables normal use of the device in the presence of such faults *and* provides security against malicious attacks.[19]

The update mechanism of the proposed scheme, works as follows: in each round, it randomly shuffles the blocks and refreshes the randomness of the inner encoding of sk. The idea here is that, due to the continual shuffling and refreshing of the inner encoding scheme, in each round the attacker learns nothing about the secret key, and every attempt to modify the inner encoding, results to an invalid key, with overwhelming probability. Our update mechanism can be made deterministic if we further encode the seed of a PRG together with the secret key, which is similar to the technique presented in [LL12].

Below we define the update mechanism, which is denoted as Update^*.

Construction 3.5.1. *Let k, $m \in \mathbb{N}$, and let $(\mathsf{KGen}, \mathsf{E}, \mathsf{D})$, $(\mathsf{SS}_m, \mathsf{Rec}_m)$, Enc^*, Dec^*, be as in Construction 3.4.1. We define the update procedure, Update^*, for the encoding scheme of Construction 3.4.1, as follows:*

- $\mathsf{Update}^*(1^k, \cdot)$*: on input C, parse it as $(C_1 || \dots || C_{\mathsf{bn}})$, set $l \leftarrow 2m|sk|$, $\hat{\mathcal{L}} = \emptyset$, and set $\hat{C} := (\hat{C}_1 || \dots || \hat{C}_{\mathsf{bn}})$ to zeros.*
  - **(Secret share $0^{2|sk|}$)**: *Sample $z \leftarrow \mathsf{SS}_m\left(0^{2|sk|}\right)$, where $z = ||_{i=1}^{2|sk|} z_i$, $z \in \{0,1\}^{2m|sk|}$, and for $i \in [2|sk|]$, z_i is an m-out-of-m secret sharing of the 0 bit.*
  - **(Shuffle & Refresh)**: *Sample $\rho := (\rho_1, \dots, \rho_l) \xleftarrow{\mathsf{rs}} \{0,1\}^{\log(\mathsf{bn})}$. For $i \in [\mathsf{bn}]$,*
    - **(Sensitive block)** *If $C_i[1] = 1$,*
      - **(Shuffle)**: *Set $j \leftarrow C_i[2 : \mathsf{bs} - 1]$, $\hat{C}_{\rho_j} \leftarrow C_i$.*
      - **(Refresh)**: *Set $\hat{C}_{\rho_j}[\mathsf{bs}] \leftarrow \hat{C}_{\rho_j}[\mathsf{bs}] \oplus z[j]$.*
    - **(Ciphertext block)**
      If $C_i[1] = 0$, set $j \leftarrow \min_n \left\{ n \in [\mathsf{bn}] \middle| n \notin \hat{\mathcal{L}}, n \neq \rho_i, i \in [l] \right\}$, and $\hat{C}_j \leftarrow C_i$, $\hat{\mathcal{L}} \leftarrow \hat{\mathcal{L}} \cup \{j\}$.

  Output $\hat{C}$.

[19] We assume that the attacks against the memory are non-persistent.

The following definition of security is along the lines of the one given in [FMVW14], adapted to the notion of non-malleability with manipulation detection. Also, after each invocation the codewords are updated, where in our case the update mechanism is only using shuffling and refreshing operations. In addition, there is no need for self-destruct after detecting an invalid codeword [FN17].

Definition 3.5.2 (Continuous MD-NMC with light updates)**.** *Let* $\mathsf{CS} = (\mathsf{Enc}, \mathsf{Dec})$ *be an encoding scheme,* $\mathcal{F}$ *be a function class and* $k, q \in \mathbb{N}$*. Then,* CS *is a* q*-continuously non-malleable code with manipulation detection (*q-MD-CNMC*) with light updates, if for every, sufficiently large* $k \in \mathbb{N}$*, any pair of messages* s_0*,* $s_1 \in \{0,1\}^{\mathrm{poly}(k)}$*, and any algorithm* $\mathcal{A}$*,*

$$\left\{\mathsf{Tamper}^{\mathcal{A}}_{s_0}(k)\right\}_{k\in\mathbb{N}} \approx \left\{\mathsf{Tamper}^{\mathcal{A}}_{s_1}(k)\right\}_{k\in\mathbb{N}},$$

where,

$\mathsf{Tamper}^{\mathcal{A}}_{s}(k)$:
$C \leftarrow \mathsf{Enc}(1^k, s), \tilde{s} \leftarrow 0$
For $\tau \in [q]$:
  $f \leftarrow \mathcal{A}(\tilde{s}), \tilde{C} \leftarrow f(C), \tilde{s} \leftarrow \mathsf{Dec}(\tilde{C})$
  If $\tilde{s} = s$: $\tilde{s} \leftarrow \mathsf{same}^*$
  $C \leftarrow \mathsf{Update}^*(1^k, C)$
$out \leftarrow \mathcal{A}(\tilde{s})$
Return : out

and for each round the output of the decoder is not in $\{s, \perp\}$ *with negligible probability in* k*, over the randomness of* $\mathsf{Tamper}^{\mathcal{A}}_{s}$*.*

Below we prove that the scheme of Construction 3.5.1 is continuously non-malleable with manipulation detection and light updates.

Theorem 3.5.3. *Let* q*,* k*,* m*,* $\in \mathbb{N}$*,* $\Gamma = \{0,1\}^{O(\log(k))}$ *and* $\alpha \in [0,1)$*. Assuming* $(\mathsf{SS}_m, \mathsf{Rec}_m)$ *is an* m*-out-of-*m *secret sharing scheme and* $(\mathsf{KGen}, \mathsf{E}, \mathsf{D})$ *is a* 1-IND-CPA*, authenticated encryption scheme, the scheme of Construction 3.5.1 is a* MD-CNMC *with light updates, against* $\mathcal{F}^{\alpha}_{\Gamma}$*, for any* α*,* m*, such that* $(1-\alpha)m = \omega(\log(k))$*.*

Proof. Let $\mathcal{A}$ be any adversary playing against $\mathsf{Tamper}^{\mathcal{A}}_{s}$, for any s. Let I_{b} be the set of indices chosen by the attacker in each round and $I^{\mathsf{c}} = [\nu] \backslash I$. The tampered components of the codeword will be denoted using the symbol "˜" on top of the original symbol. Our proof follows the strategy of the one given in Theorem 3.3.2, using a series of hybrid experiments that are depicted in Figure 3.9. Below, we describe the hybrids.

$\mathsf{Exp}_0^{\mathcal{A},s,q}$:
$C \leftarrow \mathsf{Enc}(s), \tilde{s} \leftarrow 0, \tilde{C} \leftarrow C$
For $\tau \in [q]$:
  $f \leftarrow \mathcal{A}(\tilde{s})$

  $\tilde{C}_{|_{I_b}} \leftarrow f(C_{|_{I_b}}), \tilde{s} \leftarrow \mathsf{Dec}(\tilde{C})$
  If $\tilde{s} = s$:
    $\tilde{s} \leftarrow \mathsf{same}^*$
  $C \leftarrow \mathsf{Update}^*(1^k, C)$

$out \leftarrow \mathcal{A}(\tilde{s})$
Return : out

$\mathsf{Exp}_1^{\mathcal{A},s,q}$:
$C \leftarrow \mathsf{Enc}(s), \tilde{s} \leftarrow 0, \tilde{C} \leftarrow C$
For $\tau \leq [q]$:
  $f \leftarrow \mathcal{A}(\tilde{s}),\ \tilde{s} \leftarrow \perp$
  If $\forall i : |(I_b \cap Z_i)| < m$:
    $\tilde{C}_{|_{I_b}} \leftarrow f(C_{|_{I_b}}), \tilde{s} \leftarrow \mathsf{Dec}(\tilde{C})$
    If $\tilde{s} = s$:
      $\tilde{s} \leftarrow \mathsf{same}^*$
  $C \leftarrow \mathsf{Update}^*(1^k, C)$

$out \leftarrow \mathcal{A}(\tilde{s})$
Return : out

$\mathsf{Exp}_2^{\mathcal{A},s,q}$:
$\rho \xleftarrow{\mathsf{rs}} \{0,1\}^{\log(\mathsf{bn})}$
$sk \leftarrow \mathsf{KGen}(1^k), e \leftarrow \mathsf{E}_{sk}(s)$
$\tilde{s} \leftarrow 0$
For $\tau \in [q]$:
  $f \leftarrow \mathcal{A}(\tilde{s}), \tilde{s} \leftarrow \perp$
  $z^* \leftarrow \bar{\mathsf{SS}}_m^{f,\rho}(sk), C \leftarrow \Pi_\rho(z^*||e)$
  If $\forall i : |(I_b \cap Z_i)| < m$:
    $\tilde{C}_{|_{I_b}} \leftarrow f(C_{|_{I_b}})$
    $w_i \leftarrow \bigoplus_{j \in (I_b \cap Z_i)} C_j[\mathsf{bs}]$
    $\tilde{w}_i \leftarrow \bigoplus_{j \in (I_b \cap \tilde{Z}_i)} \tilde{C}_j[\mathsf{bs}]$

    If $\forall i : w_i = \tilde{w}_i$:
      $\tilde{s} \leftarrow \mathsf{D}_{sk}(\tilde{e})$

    If $\tilde{s} = s$:
      $\tilde{s} \leftarrow \mathsf{same}^*$
  $C \leftarrow \mathsf{Update}^*(1^k, C)$

$out \leftarrow \mathcal{A}(\tilde{s})$
Return : out

$\mathsf{Exp}_3^{\mathcal{A},s,q}$:
$\rho \xleftarrow{\mathsf{rs}} \{0,1\}^{\log(\mathsf{bn})}$
$sk \leftarrow \mathsf{KGen}(1^k), e \leftarrow \mathsf{E}_{sk}(s)$
$\tilde{s} \leftarrow 0$

For $\tau \in [q]$:
  $f \leftarrow \mathcal{A}(\tilde{s}), \tilde{s} \leftarrow \perp$
  $z^* \leftarrow \bar{\mathsf{SS}}_m^{f,\rho}(sk), C \leftarrow \Pi_\rho(z^*||e)$
  If $\forall i : |(I_b \cap Z_i)| < m$:
    $\tilde{C} \leftarrow f(C_{|_{I_b}})$
    $w_i \leftarrow \bigoplus_{j \in (I_b \cap Z_i)} C_j[\mathsf{bs}]$
    $\tilde{w}_i \leftarrow \bigoplus_{j \in (I_b \cap \tilde{Z}_i)} \tilde{C}_j[\mathsf{bs}]$

    If $\forall i : w_i = \tilde{w}_i$:
      If $\tilde{e} = e$:
        $\tilde{s} \leftarrow \mathsf{same}^*$

  $C \leftarrow \mathsf{Update}^*(1^k, C)$

$out \leftarrow \mathcal{A}(\tilde{s})$
Return : out

Figure 3.9: Hybrids for the proof of Theorem 3.5.3. The gray part signifies the portion of the code of an experiment that differs from the previous one.

- $\mathsf{Exp}_0^{\mathcal{A},s,q}$: For any $\mathcal{A}$, s, q, the experiment $\mathsf{Exp}_0^{\mathcal{A},s,q}$, is the experiment $\mathsf{Tamper}_s^{\mathcal{A}}$, of Definition 3.5.2.

- $\mathsf{Exp}_1^{\mathcal{A},s,q}$: In the second experiment and for each round of the execution, we define Z_i, $i \in [2|sk|]$, to be the set of indices in which z_i is stored, $|Z_i| = m$. Intuitively, in each round, by calling the Update^* procedure that permutes the blocks using a fresh permutation key and updates the shares of sk and sk^3, we achieve the following: in each round, the attacker finds all shares for a bit of sk, and sk^3, with negligible probability in k, thus the tampering function is not accessing any bit of sk and sk^3, even if it is given access to $(1 - o(1))\nu$ blocks of the codeword. Thus, the indistinguishability between the current experiment and the previous one comes from a claim analogous to Claim 3.3.3, made in the proof of Theorem 3.3.2. In particular, we have the following claim.

  Claim 3.5.4. *For k, q, $m \in \mathbb{N}$, assume $(1-\alpha)m = \omega(\log(k))$. Then, for any $\mathcal{A}$ that chooses its tampering strategy from $\mathcal{F}_\Gamma^\alpha$, and any message s, we have $\mathsf{Exp}_0^{\mathcal{A},s,q} \approx \mathsf{Exp}_1^{\mathcal{A},s,q}$, where the probability runs over the randomness used by Enc^*, Update^*.*

- $\mathsf{Exp}_2^{\mathcal{A},s,q}$: In the third experiment we define by $\tilde{Z}_i$ to be the set of indices in which $\tilde{z}_i$ is stored, $|\tilde{Z}_i| = m$. The main difference with the previous experiment is that we unfold the encoding procedure, and in addition, we substitute the secret sharing procedure SS_m with $\bar{\mathsf{SS}}_m^{f,\rho}$, defined as follows:

  $\bar{\mathsf{SS}}_m^{f,\rho}(sk)$:

  1. Sample $\left(z_1^*, \ldots, z_{2|sk|}^*\right) \leftarrow \mathsf{SS}_m\left(sk||sk^3\right)$.
  2. For $i \in [2|sk|]$:
     $$l_i := \max_d \left\{d \in [m] \wedge \mathsf{Ind}\left(z_i[d]\right) \notin I_\mathsf{b}\right\},$$
     where Ind returns the index of $z_i[d]$ in C, i.e., l_i is the largest index in $[m]$ such that the codeword block containing $z_i[l_i]$, is not accessed by f.
  3. (**Output**): For all i set $z_i^*[l_i] = *$, and output $z^* := \|_{i=1}^{2|sk|} z_i^*$.

  In $\mathsf{Exp}_1^{\mathcal{A},s,q}$, we have $z = \|_{i=1}^{2|sk|} z_i$, and each z_i is an m-out-of-m secret sharing for a bit of sk or sk^3. From the first transition we have that for all i, $|I_\mathsf{b} \cap Z_i| < m$ with overwhelming probability, and the current experiment is identical to the previous one up to the point of computing $f(C_{|_{I_\mathsf{b}}})$, as $C_{|_{I_\mathsf{b}}}$ and $f(C_{|_{I_\mathsf{b}}})$ depend only on z^*, that gives no information about sk and sk^3.

  Another difference between the two experiments is that, after applying the tampering function, $\mathsf{Exp}_1^{\mathcal{A},s,q}$ makes a call on the decoder while $\mathsf{Exp}_2^{\mathcal{A},s,q}$, checks if the tampering

function has modified the shares in a way such that the reconstruction procedure will give $\tilde{sk} \neq sk$ or $\tilde{sk}' \neq sk'$. In case modification is detected the current experiments sends $\perp$ to the attacker. The main idea here is that, a tampering function that modifies the shares of sk and sk^3, creates shares of $\tilde{sk}$ and $\tilde{sk}'$, such that $\tilde{sk}^3 = \tilde{sk}'$, with negligible probability in k. We prove this by a reduction to the 1-IND-CPA security of the encryption scheme in the presence of z^*, that as we have already stated, it gives no information about the secret key. The indistinguishability between the two experiments comes from the following claim, whose proof is similar to the one given in Claim 3.3.4.

Claim 3.5.5. *Assuming* $(\mathsf{KGen}, \mathsf{E}, \mathsf{D})$ *is* 1-IND-CPA *secure (cf. Definition 2.1.6), for any* $\mathcal{A}$ *choosing its tampering strategy from* $\mathcal{F}_{\Gamma}^{\alpha}$*, and any message* s*,* $\mathsf{Exp}_1^{\mathcal{A},s,q} \approx \mathsf{Exp}_2^{\mathcal{A},s,q}$*, where the probability runs over the randomness used by* Enc^**,* Update^**.*

- $\mathsf{Exp}_3^{\mathcal{A},s,q}$: In the final experiment, in each round of the execution, instead of calling the decryption $\mathsf{D}_{sk}(\tilde{e})$, we first check if the attacker has modified the ciphertext, in which case the current experiment outputs $\perp$, otherwise it outputs same^*. This part of the program is reached only if the tampering function does not modify the secret key. Thus, the indistinguishability between the two experiments follows from the authenticity property of the encryption scheme in the presence of z^*, which is updated in each round depending on the set I_{b}. Clearly, requesting z^* adaptively in each round does not compromise the security of the encryption scheme, as z^* carries no information about sk. Thus, in each round, given that $\tilde{sk} = sk$ and $\tilde{sk}' = sk'$, we have that if the attacker modifies the ciphertext, then with overwhelming probability $\mathsf{D}_{sk}(\tilde{e}) = \perp$, otherwise, $\mathsf{D}_{sk}(\tilde{e}) = s$, and the current experiment correctly sends $\tilde{s} = \mathsf{same}^*$ to the attacker. Thus, we have the following claim.

  Claim 3.5.6. *Assuming the authenticity property of* $(\mathsf{KGen}, \mathsf{E}, \mathsf{D})$*, for any* $\mathcal{A}$ *choosing its tampering strategy from* $\mathcal{F}_{\Gamma}^{\alpha}$*, and any message* s*,* $\mathsf{Exp}_2^{\mathcal{A},s,q} \approx \mathsf{Exp}_3^{\mathcal{A},s,q}$*, where the probability runs over the randomness used by* KGen*,* E *and* Update^**.*

- Finally, we have that for any $\mathcal{A}$ choosing its tampering strategy from $\mathcal{F}_{\Gamma}^{\alpha\nu}$, and any message s, $\mathsf{Exp}_3^{\mathcal{A},s,q}$ is indistinguishable from $\mathsf{Exp}_3^{\mathcal{A},0,q}$, where $\mathsf{0}$ denotes the zero-message. This follows by the semantic security of the encryption scheme in the presence of z^*, for the multi-round case.

Claim 3.5.7. *Assuming* $(\mathsf{KGen}, \mathsf{E}, \mathsf{D})$ *is semantically secure, for any* $\mathcal{A}$ *choosing its tampering strategy from* $\mathcal{F}_\Gamma^\alpha$, $\mathsf{Exp}_3^{\mathcal{A},s,q} \approx \mathsf{Exp}_3^{\mathcal{A},0,q}$ *, where the probability runs over the randomness used by* KGen, E, Update, *"$\approx$" may refer to statistical or computational indistinguishability, and* 0 *denotes the zero-message.*

The above claims conclude our proof. Clearly, the manipulation detection property follows from the fact that the output of Exp_3 is in $\{\mathsf{same}^*, \perp\}$, with overwhelming probability. □

In the above theorem, q can be polynomial (resp. exponential) in k, assuming the underlying encryption scheme is computationally (resp. unconditionally) secure.

3.6 Instantiations

Below, we define a rate 1, computationally secure authenticated encryption scheme.

Instantiation 3.6.1 (IND-CPA secure authenticated encryption (computational))**.** *Let* F_r *be a pseudo-random function,* $F_r : \{0,1\}^k \to \{0,1\}^k$*, let* PRG *be a pseudo-random generator,* $\mathsf{PRG} : \{0,1\}^k \to \{0,1\}^{|s|}$*, and let* $(\mathsf{MKGen}, \mathsf{Mac}, \mathsf{Vrfy})$ *be a message authentication code that outputs tags of length* k *(cf. [KL14]). We define a symmetric encryption scheme* $(\mathsf{KGen}, \mathsf{E}, \mathsf{D})$*, as follows:*

- $\mathsf{KGen}(1^k)$*: sample* $r \leftarrow \{0,1\}^k$*,* $mk \leftarrow \mathsf{Mac}(1^k)$ *and output* $sk = (r, mk)$*.*
- $\mathsf{E}_{sk}(\cdot)$*: On input* s*, sample* $\tau \leftarrow \{0,1\}^k$*, set* $e = (\mathsf{PRG}(F_r(\tau)) \oplus s, \tau)$*,* $t = \mathsf{Mac}_{mk}(e)$*, and output* (e, t)*.*
- $\mathsf{D}_{sk}(\cdot)$*: On input* (e, t)*, if* $\mathsf{Vrfy}_{mk}(e, t) = 1$*, parse* e *as* (e', τ) *and output* $s = (\mathsf{PRG}(F_r(\tau)) \oplus e')$*, otherwise output* $\perp$*.*

It is not hard to see that the scheme defined above is a rate 1, computationally secure authenticated encryption scheme [KL14]. By instantiating Construction 3.3.1 with the above scheme, Theorem 3.3.2, for $m = k \log k$, $\alpha = 1 - 1/\Omega(\log k)$, yields a rate 1 MD-NMC, with access rate $1 - 1/\Omega(\log k)$ and codewords of length $|s| + O(k^2 \log k)$, assuming one-way functions. In addition, by instantiating Constructions 3.4.1, 3.5.1 with the above scheme, Theorems 3.4.8, 3.5.3, for $m = k \log k$, $\alpha = 1 - 1/\Omega(\log k)$, yield rate $1 - 1/\Omega(\log k)$ MD-NMC, with access rate $1 - 1/\Omega(\log k)$ and codewords of length $|s|(1 + 1/O(\log k)) + O(k^2 \log^2 k)$, assuming one-way functions.

Below we define a generic, one-time secure, information theoretic construction.

Instantiation 3.6.2 (1-IND-CPA secure authenticated encryption (information theoretic))**.** *Let $\mathcal{H}$, $\bar{\mathcal{H}}$, be pair-wise independent hash function families, such that for any $h \in \mathcal{H}$, $h : \{0,1\}^{O(|s|)} \to \{0,1\}^{|s|}$ and for any $\bar{h} \in \bar{\mathcal{H}}$, $\bar{h} : \{0,1\}^{O(|s|)} \to \{0,1\}^{|s|}$. We define a symmetric encryption scheme $(\mathsf{KGen}, \mathsf{E}, \mathsf{D})$, as follows:*

- $\mathsf{KGen}(1^k)$*: sample $h \leftarrow \mathcal{H}$, $\bar{h} \leftarrow \bar{\mathcal{H}}$ and set $sk = (h, \bar{h})$.*
- $\mathsf{E}_{sk}(\cdot)$*: On input s, sample $r \leftarrow \{0,1\}^{|s|}$, set $e = (r||(h(r) + s))$ and output $(e, \bar{h}(e))$.*
- $\mathsf{D}_{sk}(\cdot)$*: On input (e, t), if $\bar{h}(e) = t$, parse e as $(r||e')$ and output $s = h(r) + e'$, otherwise output $\perp$.*

The security of the above scheme comes from the pair-wise independence of $\mathcal{H}$, $\bar{\mathcal{H}}$. By instantiating Construction 3.3.1 with the above scheme, Theorem 3.3.2, for $m = |s| \log |s|$, $\alpha = 1 - 1/O(\log(|s|))$, yields an unconditionally secure MD-NMC in the CRS model, with concrete information rate $1/O(|s| \log(|s|))$, access rate $1 - 1/\Omega(\log(|s|))$ and codewords of length $O(|s|^2 \log |s|)$. In addition, by instantiating Constructions 3.4.1, 3.5.1 with the above scheme, Theorems 3.4.8, 3.5.3, for $m = |s| \log |s|$, $\alpha = 1 - 1/O(\log(|s|))$, yield unconditionally secure, rate $1/O(|s| \log^2(|s|))$ MD-NMC in the standard model, with access rate $\Omega(1 - 1/\log(|s|))$ and codewords of length $O(|s|^2 \log^2 |s|)$.

Chapter 4

Extractable hash function families

4.1 Introduction

The notion of extractable collision-resistant hash functions (ECRHs) was originally proposed by [BCCT12, GLR11, BCC+14] as a tool for building efficient succinct non-interactive arguments of knowledge (SNARKs). Informally, a family of functions, $\mathcal{H}$, is extractable, if for a uniform $h \in \mathcal{H}$, sampling an element $v \in \mathsf{Image}(h)$ without actually evaluating the function on a pre-image s, i.e., by computing $h(s) = v$, is infeasible. This concept is formalized in the following way: for any algorithm $\mathcal{A}$ that produces some $v \in \mathsf{Image}(h)$, there exists an extractor that, possibly depending on the code of $\mathcal{A}$, outputs a preimage s, such that $h(s) = v$. Typically, such families are interesting only if they posses some sort of hardness property, like one-wayness, or otherwise the problem can be trivial. In [BCC+14] the authors propose two constructions for ECRH: the first one is based on a variant of the Knowledge of Exponent assumption, called t-KEA(cf. Assumption 2.1.7), and the hardness of the discrete logarithm problem (cf. Definition 2.1.5), while the latter uses a lattice based knowledge assumption, called Knowledge of Knapsack [BCCT12].

An important observation regarding the setting described above is that ECRHs provide no guarantee against attackers that receive access to precomputed hash values, prior to producing their own. However, there are applications of the primitive that can deviate from the above setting. For instance, the attacker might be given access to a number of valid hash values for which it does not know the pre-images, prior to delivering its own hash value. In this setting, creating a new valid hash value could be achieved by mauling the received ones without knowing the pre-image. To tackle this issue, the present thesis introduces the notion of ℓ-more extractable hash functions. Briefly speaking, ℓ-more extractable hash function families capture the following idea: if an adversary is given access

to $\ell \in \mathbb{N}$ precomputed hash values $v_1, \ldots, v_\ell$, and produces a new valid hash value $\tilde{v}$, then it must know a pre-image of $\tilde{v}$. As we prove later, this notion is not implied by the one by Bitansky et al. [BCCT12] and Goldwasser et al. [GLR11], which considers adversaries that get no access to precomputed hash values prior to producing their own value, assuming the hardness of the discrete logarithm problem. Moreover, by requiring the attacker not only to produce some $\tilde{v} \in \mathsf{Image}(h)$ but also to come up with a valid pre-image for $\tilde{v}$, we prove that ℓ-more extractable hash functions are feasible under the same assumptions used in [GLR11, BCCT12]. This puts forth a weaker form of extractability (we refer to is as wECRH) in the sense that the extractor is allowed to fail in case a pre-image exists but is not somehow efficiently computable based on the view of the adversary. It should be noted that there is no contradiction in terms here: this extra requirement does not trivialize the notion of ℓ-more wECRH, since the extractor is required to depend only on the adversarial program that produces $\tilde{v}$ and is independent of the program that produces the valid pre-image for $\tilde{v}$.

In this chapter, we show how to construct efficient, leakage-resilient ℓ-more wECRHs, and in subsequent chapters, we show that this weaker form of extractability is sufficient for constructing *efficient, computationally secure, non-malleable codes* (cf. Chapter 5) and *continuous non-malleable codes* (cf. Chapter 6), against split-state attackers, as well as *efficient non-interactive, non-malleable commitments, with respect to opening* (cf. Chapter 7). The next section summarizes (informally) our results on ℓ-more wECRH.

4.1.1 Results

In this chapter, a new cryptographic primitive is introduced and constructed, called *ℓ-more extractable hash function families*. Briefly speaking, ℓ-more extractable hash function families capture the idea that, if an adversary, that given $\ell \in \mathbb{N}$ precomputed hash values $v_1, \ldots, v_\ell$, manages to produce a new valid hash value $\tilde{v}$, then it must know a pre-image of $\tilde{v}$. This is a generalization of the notion of extractable hash functions by Bitansky et al. [BCCT12] and Goldwasser et al. [GLR11], which corresponds to the $\ell = 0$ case (i.e., the adversary gets no access to valid hash values, prior to producing its own value), and is somewhat reminiscent of the strengthening of simulation-soundness in the context of zero-knowlege proofs [Sah99].

The proposed generalization is strict, as the following, informally stated theorem, is proved later in this chapter.

Theorem 4.1.1 (Informal)**.** *Extractable hash* $\not\Longrightarrow$ *1-more extractable hash.*

The subtlety comes from the fact that the ℓ-more attacker might get an "unfair advantage" in producing a valid hash value, for which it does not possess a pre-image, because of the ℓ additional inputs; e.g., by modifying the v_i's in some suitable way. Indeed, as it is proven below, the hash function family of Bitansky et al. [BCCT12] is easily malleable, and thus exploitable by "1-more" attackers. This demonstrates that ℓ-more extractability is strictly stronger than the original notion [BCCT12].

The next step is to construct this notion. We prove that, by requiring the attacker not only to produce a valid hash $\tilde{v}$, but also to come up with a valid pre-image for $\tilde{v}$, ℓ-more extractability can be achieved under the same assumptions used by the construction of Bitansky et al. [BCCT12], i.e., a variant of the Knowledge of Exponent Assumption KEA and DLOG. As a conclusion, KEA and DLOG are still sufficient to achieve ℓ-more extractable hash functions with a *weaker* form of extractability (wECRH). In detail, the following, informally stated theorem can be derived.

Theorem 4.1.2 (Informal)**.** DLOG *and* t-KEA *imply* ℓ*-more* wECRH*.*

At this point, it should be noted that KEA is non-falsifiable (cf. [Nao03]), and it is indeed a strong assumption. However, one can argue that non-falsifiability might be inherent for extractable hash functions, and thus for ℓ-more extractability, by recalling the results of Bitansky et al. [BCCT12] and Gentry and Wichs [GW11]: in [BCCT12] the authors showed that extractable hash function families imply succinct non-interactive arguments of knowledge (SNARKs), while Gentry and Wichs [GW11] showed that SNARKs are unlikely to be constructed based on falsifiable assumptions. Thus, non-falsifiable assumptions are likely to be inherent for achieving (ℓ-more) extractability. It should be also noted that some variants of KEA were shown to contradict (public-coin) differing-inputs obfuscation and indistinguishability obfuscation [BCPR14, BP15], however the variant used in the present thesis is suitably defined to circumvent this contradiction.

Next, we prove that any hash function that is modeled as a random oracle, is an ℓ-more wECRH, receiving the following informal theorem.

Theorem 4.1.3 (Informal)**.** *Let* h *be a hash function. If* h *is modeled as a random oracle, then it is an* ℓ*-more* wECRH*.*

We note that, even though any function h that is modeled as a random oracle is an ℓ-more wECRH, it cannot be extractable according to [BCCT12, BCC$^+$14], as the range of the function is dense, thus an attacker can just output a random element in the range

of the hash, and then there is no knowledge to extract, as the extractor needs to invert a uniform valid hash value [BCCT12].

Finally, as an extension of the notion of ℓ-more wECRH, we formalize the notion of *leakage-resilient, ℓ-more, weakly extractable hash function families*, considering attackers that, in addition to receiving access to ℓ precomputed hash values, they also receive *bounded leakage* over the randomness used to compute those values. Then, we prove security for the construction of the Informal Theorem 4.1.2, in the presence of leakage over the randomness that is used to compute the hash. In particular, we prove the following, informally stated, theorem.

Theorem 4.1.4 (Informal)**.** DLOG *and* t-KEA *imply leakage-resilient* ℓ*-more* wECRH*.*

For the construction of the Informal Theorem 4.1.3, leakage-resilience is straightforward, as the adversary receives black box access to the hash function.

4.1.2 KEAs and previous work

In [Dam92], Damgård introduces KEA to construct a CCA-secure encryption scheme. In [HT98, BP04], the authors extend the assumption of [Dam92], and construct three-round, zero-knowledge arguments. Abe and Fehr [AF07] construct the first perfect NIZK for NP with adaptive soundness, by extending the assumption of [BP04]. Prabhakaran and Xue [PX09] constructed statistically-hiding sets for trapdoor DDH groups, by introducing a new knowledge assumption. Gennaro et al. [GKR10] proved that a modified version of the Okamoto-Tanaka key-agreement protocol [Oka88] satisfies perfect forward secrecy against fully active attackers, by introducing a new knowledge assumption. In [BCCT12, BCC$^+$17, Gro10, BCCT13, GGPR13], the authors construct succinct, non-interactive, arguments of knowledge (SNARKs), and NIZKs, while in [CD08, CD09], Canetti and Dakdouk provide an extensive study on extractable functions. In [PHGR13], Parno et al. show how to perform verifiable computation, efficiently.

In [BCPR14, BP15], the authors show that, assuming indistinguishability obfuscation [BGI$^+$01], extractable one-way functions, and thus ECRHs, do not exist against adversaries receiving arbitrary, polynomial-size, auxiliary input, if the code of the extractor is fixed before the attacker's auxiliary input. On the other hand, they show that, under standard assumptions, extractable one-way functions, may exist against adversaries with bounded auxiliary input.

In this work, and as it is suggested by [BCPR14], we consider individual auxiliary input, i.e., we allow the auxiliary input of the extractor to depend on the attacker's auxiliary input, and therefore, we do not contradict the impossibility results of [BCPR14, BP15].

4.2 ℓ-more weakly extractable hash function families

In this section we define the notion of *ℓ-more weakly extractable hash function families*, and we provide a general discussion on the primitive.

Definition 4.2.1 (ℓ-more weakly extractable hash function families)**.** *For $\ell \in \mathbb{N}$, an efficiently samplable hash function ensemble $\mathcal{H} = \{\mathcal{H}_k\}_{k\in\mathbb{N}}$, is ℓ-more extractable, if for any* PPT *algorithm $\mathcal{A}_v$ and any $z_v \in \{0,1\}^{\mathrm{poly}(k)}$, there exist a* PPT *extractor $\mathcal{E}^{\mathcal{H}}_{\mathcal{A}_v}$ and $z_{\mathcal{E}} \in \{0,1\}^{\mathrm{poly}(k)}$, such that for all* PPT *algorithms $\mathcal{A}_s$, any large $k \in \mathbb{N}$ and any vector of messages $\mathbf{s} = (s_1, \ldots, s_\ell)$,*

$$\Pr_{h\leftarrow\mathcal{H}_k}\left[\mathsf{Exp}^{\mathbf{s},h}_{\mathcal{A}_v,\mathcal{A}_s,\mathcal{E}^{\mathcal{H}}_{\mathcal{A}_v}}(\ell, z_v, z_{\mathcal{E}}) = 1\right] \leq \mathsf{negl}(k),$$

where,

$\mathsf{Exp}^{\mathbf{s},h}_{\mathcal{A}_v,\mathcal{A}_s,\mathcal{E}^{\mathcal{H}}_{\mathcal{A}_v}}(\ell, z_v, z_{\mathcal{E}})$:
$\forall i \in [\ell], \tau_i \leftarrow U_{\{0,1\}^{\mathrm{poly}(k)}}, v_i \leftarrow h(s_i;\tau_i)$ *(* **hash computation** *)*
$\mathbf{t} := (\tau_1, \ldots, \tau_\ell), \mathbf{v} := (v_1, \ldots, v_\ell)$
$(\tilde{v}, st) \leftarrow \mathcal{A}_v(h, \mathbf{v}, z_v)$ *(* **hash tampering** *)*
$(\hat{\tau}, \hat{s}) \leftarrow \mathcal{E}^{\mathcal{H}}_{\mathcal{A}_v}(h, \mathbf{v}, z_{\mathcal{E}})$ *(* **pre-image extraction** *)*
$(\tilde{\tau}, \tilde{s}) \leftarrow \mathcal{A}_s\,(h, \mathbf{t}, \mathbf{s}, st)$ *(* **pre-image tampering** *)*

If $h(\tilde{s};\tilde{\tau}) = \tilde{v} \wedge \forall i : \tilde{v} \neq v_i \wedge h(\hat{s};\hat{\tau}) \neq \tilde{v}$, return 1
otherwise, return 0

The main steps in the above experiment are the following. Initially, we sample the randomness that is required for computing the hashes, and we perform the hash computation over $\ell \in \mathbb{N}$, pre-images. For deterministic hash function families we just omit randomness sampling, and we compute the hashes directly over the messages. The challenge for the adversary $\mathcal{A}_v$, is to produce a valid hash value $\tilde{v}$, given ℓ has values, denoted as $\mathbf{v}$, and auxiliary information z_v; it also produces state information, denoted as st. Then, the extractor $\mathcal{E}^{\mathcal{H}}_{\mathcal{A}_v}$ is executed, given $\mathbf{v}$ and its own auxiliary input $z_{\mathcal{E}}$. Notice, that, we allow

the auxiliary input of the extractor to depend on the attacker's auxiliary input, thus our definition is not contradicting the impossibility results of [BCPR14, BP15]. Finally, the adversary $\mathcal{A}_s$ is required to produce a valid pre-image for $\tilde{v}$, while given all information generated during the execution. The output of the experiment is 1, if $\mathcal{A}_v$ produces a valid hash value $\tilde{v}$, $\mathcal{A}_s$ produces a valid pre-image for $\tilde{v}$, while the extractor fails.

Leaving aside the fact that the above definition considers randomized function families, the major difference between the current definition and the one given by Bitansky et al. [BCCT12, BCC$^+$14] (cf. Definition 2.1.8), is two-fold: first the "ℓ-more" generalization that allows the attacker to have access to $\ell \in \mathbb{N}$ precomputed hash values for which it does not know the corresponding randomness, prior to delivering its own hash value. Second, the introduction of the algorithm $\mathcal{A}_s$, that takes the place of the existential quantifier that appears in the original definition. This is in fact a *weakening* of the original definition, in the sense that the extractor is allowed to fail in case a pre-image exists but is not efficiently computable based on the view of the adversary, which would not be allowed in the original definition.

Note that, weaker extractability does not hurt the applicability of the primitive, as there are many settings in which the attacker, is not only required to produce a valid hash value $\tilde{v}$, but also to provide a valid pre-image for it. For instance, in our application on non-malleable codes later in the thesis, our codeword stores a secret key and its hash value, and any attacker that modifies the hash value, also needs to come up with a valid pre-image, otherwise it creates an invalid codeword, assuming the collision resistance property of the hash function family. In addition, the existence of $\mathcal{A}_s$ does not trivialize the problem for the extractor since the extractor is challenged to produce a valid pre-image for $\tilde{v}$, given only the code of $\mathcal{A}_v$ and its own auxiliary input, and in particular it lacks access to the state and the program of $\mathcal{A}_s$.

It is easy to see that, constructing ℓ-more extractable hash function families that are non-compressing, can be achieved using existing tools, such as robust NIZKs [DDO$^+$01]. In the present thesis, we construct ℓ-more weakly extractable, collision resistant, hash function families (wECRH), achieving length-efficiency comparable to that of a regular hash function.

In the following lemma we prove that, for any ℓ-more wECRH function family, the output of the extractor should match the output of $\mathcal{A}_s$, in case both of them output valid pre-images, otherwise we break collision resistance.

Lemma 4.2.2. *Let $\mathcal{H} = \{\mathcal{H}_k\}_{k\in\mathbb{N}}$ be a collision resistant, ℓ-more weakly extractable,*

efficiently samplable, hash function ensemble. Then, for any $\mathcal{A}_v$, z_v, $\mathcal{E}^{\mathcal{H}}_{\mathcal{A}_v}$, $z_{\mathcal{E}}$, $\mathcal{A}_s$, $\mathbf{s} = (s_1, \ldots, s_\ell)$, ℓ, *as they were defined in Definition 4.2.1, we have*

$$\Pr_{h \leftarrow \mathcal{H}_k} \left[\begin{array}{c} \mathsf{Exp}^{\mathbf{s},h}_{\mathcal{A}_v,\mathcal{A}_s,\mathcal{E}^{\mathcal{H}}_{\mathcal{A}_v}}(\ell, z_v, z_{\mathcal{E}}) = 0, h(\tilde{s};\tilde{\tau}) = \tilde{v}, \tilde{v} \neq v_i, i \in [\ell] : \\ (\hat{\tau}, \hat{s}) \neq (\tilde{\tau}, \tilde{s}) \end{array} \right] \leq \mathsf{negl}(k).$$

Proof. We are given that the output of $\mathsf{Exp}^{\mathbf{s},h}_{\mathcal{A}_v,\mathcal{A}_s,\mathcal{E}^{\mathcal{H}}_{\mathcal{A}_v}}(\ell, z_v, z_{\mathcal{E}})$ is 0, and $\mathcal{A}_s$ succeeds in producing a valid pre-image $(\tilde{\tau}, \tilde{s})$ for a new hash, $\tilde{v}$. Since the output of the experiment is 0, we know that $\mathcal{E}^{\mathcal{H}}_{\mathcal{A}_v}$ produces a valid pre-image $(\hat{\tau}, \hat{s})$ for $\tilde{v}$. Assuming, $(\hat{\tau}, \hat{s}) \neq (\tilde{\tau}, \tilde{s})$, we break the collision resistance property of $\mathcal{H}_k$.

Concretely, assume there exist $\mathcal{A}_v$ with auxiliary input z_v, extractor $\mathcal{E}^{\mathcal{H}}_{\mathcal{A}_v}$ with auxiliary info $z_{\mathcal{E}}$, algorithm $\mathcal{A}_s$, and vector of messages $\mathbf{s}$, such that

$$\Pr_{h \leftarrow \mathcal{H}_k} \left[\begin{array}{c} \mathsf{Exp}^{\mathbf{s},h}_{\mathcal{A}_v,\mathcal{A}_s,\mathcal{E}^{\mathcal{H}}_{\mathcal{A}_v}}(\ell, z_v, z_{\mathcal{E}}) = 0, h(\tilde{\tau}, \tilde{s}) = \tilde{v}, \tilde{v} \neq v_i, i \in [\ell] : \\ (\hat{\tau}, \hat{s}) \neq (\tilde{\tau}, \tilde{s}) \end{array} \right] > \epsilon, \tag{4.1}$$

for $\epsilon = 1/\mathrm{poly}(k)$. We define a PPT adversary $\mathcal{A}$ who simulates $\mathsf{Exp}^{\mathbf{s},h}_{\mathcal{A}_v,\mathcal{A}_s,\mathcal{E}^{\mathcal{H}}_{\mathcal{A}_v}}(\ell, z_v, z_{\mathcal{E}})$ while playing against the collision finding experiment $\mathsf{Hcoll}_{\mathcal{A},\mathcal{H}_k}$ (cf. Definition 2.1.4): $\mathcal{A}$, after receiving the key of the hash function, it simulates $\mathsf{Exp}^{\mathbf{s},h}_{\mathcal{A}_v,\mathcal{A}_s,\mathcal{E}^{\mathcal{H}}_{\mathcal{A}_v}}(\ell, z_v, z_{\mathcal{E}})$, and outputs $\mathbf{x}_1 = (\tilde{\tau}, \tilde{s})$ and $\mathbf{x}_2 = (\hat{\tau}, \hat{s})$, i.e., $\mathbf{x}_1$ is the output of $\mathcal{A}_s$ and $\mathbf{x}_2$ is the output of $\mathcal{E}^{\mathcal{H}}_{\mathcal{A}_v}$, computed while executing the experiment. Then, assuming Relation 4.1 holds, we have $h(\mathbf{x}_1) = h(\mathbf{x}_2)$, and the collision resistance property of h breaks, with non-negligible probability. □

Next, we show a separation between extractability and general ℓ-more extractability, as we discussed in earlier. In particular, we prove that the extractable hash of [BCCT12] is not 1-more (weakly) extractable. Before doing so, we first revisit their construction, and t-KEA (Assumption 2.1.7), upon which the constructions is based.

Assuming a group $\mathbb{G}$, of prime order p, the Knowledge of Exponent Assumption KEA, introduced by Damgård [Dam92], states the following: any adversary that is given a generator, g, of $\mathbb{G}$, and a random group element g^a, produces the pair (g^s, g^{as}), only if it "knows" the exponent s. The assumption was later extended by [HT98, BP04], by requiring that, given g^{r_1}, g^{ar_1}, g^{r_2}, g^{ar_2}, it is infeasible to produce $v = g^{r_1 s_1 + r_2 s_2}$ and v^a, without "knowing" s_1, s_2. This assumption, generalized for $t = \mathrm{poly}(\log |\mathbb{G}|)$ pairs g^{r_i}, g^{ar_i}, is referred to as t-KEA by [BCCT12].

An element from the hash function family of [BCCT12] is described by the pair $(g^{\mathbf{r}}, g^{a\mathbf{r}})$, for uniformly random vector $\mathbf{r}$, and element a. Note that, $g^{\mathbf{r}}$ denotes the value $(g^{r_1}, \ldots, g^{r_t})$, where $\mathbf{r} = (r_1, \ldots, r_t)$. The hash of a message $\mathbf{s} = (s_1, \ldots, s_t)$, is the pair $(g^{\langle \mathbf{r},\mathbf{s}\rangle}, g^{a\langle \mathbf{r},\mathbf{s}\rangle})$, where $\langle \mathbf{r}, \mathbf{s}\rangle$ denotes the inner product of $\mathbf{r}$, $\mathbf{s}$. It is not hard to see that the hash value can be computed efficiently given the message and the description of the hash function, and assuming the t-KEA, the above hash function family is extractable according to [BCCT12]. We formally define the construction below.

Construction 4.2.3 (Extractable hash from t-KEA [BCCT12])**.** *Let $\mathcal{G}$ be a group-generation algorithm. An instance of a $(kt, 2k)$-compressing, hash function family, $\mathcal{H}^* = (\mathsf{Gen}^*, h^*)$, with respect to $\mathcal{G}$, is defined as follows:*

1. $\mathsf{Gen}^*(1^k)$*: sample* $(\mathbb{G}, g, p) \leftarrow \mathcal{G}(1^k)$*,* $p \in (2^{k-1}, 2^k)$*,* $(a, \mathbf{r}) \leftarrow \mathbb{Z}_p \times \mathbb{Z}_p^t$*, where* $p = |\mathbb{G}|$*, and output* $z := (\mathbb{G}, g^{\mathbf{r}}, g^{a\mathbf{r}})$*.*

2. ***Hashing computation****: on input* $\mathbf{s}$*, compute* $h_z^*(\mathbf{s}) := (g^{\langle \mathbf{r},\mathbf{s}\rangle}, g^{\langle a\mathbf{r},\mathbf{s}\rangle})$*.*

For brevity, in what follows $\mathbb{G}$ will be omitted from the description of the hash, and we will use h^* to refer both to the program of the hash and the key of a specific element $(g^{\mathbf{r}}, g^{a\mathbf{r}})$, i.e., z is omitted.

In [BCCT12] the authors prove that Construction 4.2.3 is collision resistant, which is revisited in the following lemma.

Lemma 4.2.4 (Collision resistance for Construction 4.2.3 [BCCT12])**.** *Assuming the hardness of the discrete logarithm problem, with respect to a group $\mathbb{G}$, Construction 4.2.3 is collision resistant, with respect to $\mathbb{G}$.*

The intuition behind the fact that Construction 4.2.3 is not 1-more extractable, is the following. Suppose the adversary receives a hash value $v := h(\mathbf{s}) = (g^{\langle \mathbf{r},\mathbf{s}\rangle}, g^{a\langle \mathbf{r},\mathbf{s}\rangle})$, for some unknown message $\mathbf{s}$, and then computes $v' := v^x = (g^{\langle \mathbf{r},x\mathbf{s}\rangle}, g^{a\langle \mathbf{r},x\mathbf{s}\rangle})$, for some non-zero x, of its choice. Clearly, the new hash value v' equals $h(x\mathbf{s})$, and thus, it is valid. Then, assuming an extractor for the current family, under the "1-more" setting, the original message $\mathbf{s}$ can be retrieved, by first extracting $x\mathbf{s}$ and then dividing each coordinate of $\mathbf{s}$ by x. This idea can be turned into a DLOG solver, and thus, assuming the hardness of DLOG with respect to $\mathbb{G}$, it is shown in Lemma 4.2.5, that the above construction is not 1-more extractable.

Lemma 4.2.5 (Construction 4.2.3 is not 1-more extractable)**.** *Let $\mathcal{H}^*$ be the hash function family of Construction 4.2.3, with respect to a group generation algorithm $\mathcal{G}$. Then, assuming the hardness of the discrete logarithm problem for $\mathcal{G}$ (Definition 2.1.5), $\mathcal{H}^*$ is not* 1*-more weakly extractable.*

Proof. Let $k \in \mathbb{N}$, $t = O(\text{poly}(k))$, and let $\mathcal{G}$ be a group-generation algorithm, for which the discrete logarithm problem is hard. Assuming the hash function family $\mathcal{H}^*$ is 1-more extractable with respect to $\mathcal{G}$, we define a PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ that breaks the hardness assumption on the discrete logarithm problem, with non-negligible probability in k. $\mathcal{A}$ executes the following steps

1. (**Define $\mathcal{A}_v$, z_v**): $\mathcal{A}_v(h^*, v, z_v) = (v^x, st)$, where x is a fixed, non-zero, element in $\mathbb{Z}_p$, and z_v, st, are zero-length strings.

2. (**Define $\mathcal{A}_s$**): $\mathcal{A}_s(h^*, s, st) = xs$.

3. $\mathcal{A}$ executes the following steps while playing against $\mathsf{DLog}_{\mathcal{A},\mathcal{G}}$:

   a) On input $(\mathbb{G}, g, p, w)$, where $w = g^{s'}$ and s' is uniform over $\mathbb{Z}_p$, sample $(a, \mathbf{r}, \mathbf{s}) \leftarrow \mathbb{Z}_p \times \mathbb{Z}_p^t \times \mathbb{Z}_p^{t-1}$, $\mathbf{s} = (s_1, \ldots, s_{t-1})$, i.e., it sample a hash function from $\mathcal{H}^*$ and a vector message $\mathbf{s}$ with $t-1$ coordinates. Then sets $h^* := (g^{\mathbf{r}}, g^{a\mathbf{r}})$ and partially simulates $\mathsf{Exp}^{\mathbf{s}',h^*}_{\mathcal{A}_v,\mathcal{A}_s,\mathcal{E}^{\mathcal{H}^*}_{\mathcal{A}_v}}(1, z_v, z_{\mathcal{E}})$, where $\mathbf{s}' = (s', s_1, \ldots, s_{t-1})$, without accessing s'. Here, $\mathcal{E}^{\mathcal{H}^*}_{\mathcal{A}_v}$ and $z_{\mathcal{E}}$ are totally defined by $\mathcal{A}_v$ and z_v, since we assume that $\mathcal{H}^*$ is 1-more extractable.

   b) Compute $h^*(s', s_1, \ldots, s_{t-1})$ while not having access to s', i.e., compute $v = ((g^{s'})^{r_1} \cdot g^d, (g^{s'})^{ar_1} \cdot g^{ad})$, where $d = \langle [\mathbf{r}]_{(2:t)}, \mathbf{s} \rangle$.

   c) Sample $(\tilde{v}, st) \leftarrow \mathcal{A}_v(h^*, v, z_v)$, where by definition

   $$\tilde{v} = v^x = \left(\left(g^{(r_1 s' + d)} \right)^x, \left(g^{a(r_1 s' + d)} \right)^x \right).$$

   d) Sample $\hat{\mathbf{s}} \leftarrow \mathcal{E}^{\mathcal{H}^*}_{\mathcal{A}_v}(h^*, v, z_{\mathcal{E}})$, and output $s = r_1^{-1}(x^{-1}\langle \mathbf{r}, \hat{\mathbf{s}} \rangle - d)$.

It is not hard to see that v and $\tilde{v}$ are valid hash values with respect to $\mathcal{H}^*$, and the execution of $\mathcal{A}_s(h^*, \mathbf{s}', st)$ would yield $x\mathbf{s}' = (xs', xs_1, \ldots, xs_{t-1})$, which is a valid pre-image for $\tilde{v}$. Moreover, $\mathcal{A}$, that is not having access to s', does not need to fully simulate $\mathsf{Exp}^{\mathbf{s}',h^*}_{\mathcal{A}_v,\mathcal{A}_s,\mathcal{E}^{\mathcal{H}^*}_{\mathcal{A}_v}}(1, z_v, z_{\mathcal{E}})$, since the extractor, $\mathcal{E}^{\mathcal{H}^*}_{\mathcal{A}_v}$, does not depend on $\mathcal{A}_s$. In other words, with overwhelming probability, over the execution of $\mathsf{Exp}^{\mathbf{s}',h^*}_{\mathcal{A}_v,\mathcal{A}_s,\mathcal{E}^{\mathcal{H}^*}_{\mathcal{A}_v}}(1, z_v, z_{\mathcal{E}})$, $\mathcal{A}_s$ would output the right pre-image for $\tilde{v}$ while having access to $\mathbf{s}'$, still this event does not need to

be triggered in order for $\mathcal{E}_{\mathcal{A}_v}^{\mathcal{H}^*}$ to output a valid pre-image for $\tilde{v}$. Concretely, and assuming $\mathcal{H}^*$ is 1-more extractable we have

$$\Pr_{h^* \leftarrow \mathcal{H}_k^*}[h^*(\hat{\mathbf{s}}) = \tilde{v}] \geq 1 - \mathsf{negl}(k), \tag{4.2}$$

where

$$\begin{aligned}
\Pr_{h^* \leftarrow \mathcal{H}_k^*}[h^*(\hat{\mathbf{s}}) = \tilde{v}] &= \Pr\left[\left(g^{<\mathbf{r},\hat{\mathbf{s}}>}, g^{<a\mathbf{r},\hat{\mathbf{s}}>}\right) = \left(g^{x(r_1 s' + d)}, g^{ax(r_1 s' + d)}\right)\right] \\
&= \Pr[< \mathbf{r}, \hat{\mathbf{s}} >= x(r_1 s' + d)] \\
&= \Pr[r_1^{-1}(x^{-1} < \mathbf{r}, \hat{\mathbf{s}} > -d) = s'] = \Pr[s = s'].
\end{aligned} \tag{4.3}$$

By Relations (4.2) and (4.3) we have

$$\Pr[s = s'] \geq 1 - \mathsf{negl}(k),$$

and $\mathsf{DLog}_{\mathcal{A},\mathcal{G}}(k) = 1$, i.e., $g^s = g^{s'}$, with non-negligible probability in k. □

From the above lemma, we derive that the notion of extractability according to [BCCT12], does not imply 1-more, weak extractability. By inspecting the above proof, we can also derive a similar relation between extractability [BCCT12] and 1-more extractability (not the weaker form), i.e., for the general notion of 1-more extractability that does not require $\mathcal{A}_s$. This is due to the fact that, in the above proof, the attacker against the hardness of DLOG does not require access to $\mathcal{A}_s$, thus a similar proof holds for proving that extractability [BCCT12] does not imply 1-more extractability, and thus the latter notion is strictly stronger than the one by [BCCT12]. Later in the thesis, we present a construction which is 1-more weakly extractable, but it is not extractable according to [BCCT12], which gives us a separation between the notions.

4.3 Constructing 1-more weakly extractable hash functions from KEA

In the current section we construct 1-more wECRH under t-KEA, but before doing so, we present the main idea behind our technique. Having in mind Lemma 4.2.5 and its proof, the main observation is that, even though the hash function family of Construction 4.2.3 is malleable, the modified hash value, $\tilde{v}$, has some structure: it is the hash value of the message yielded after applying an affine transformation on the original message, $\mathbf{s}$ (in the above case, the affine transformation was $x\mathbf{s}$). Interestingly, we prove later in this section,

that under t-KEA, applying an affine transformation is the only thing the adversary can do! In particular, it is shown that, if the adversary outputs a valid, new hash value, $\tilde{v}$, then there exists an extractor that extracts an affine transformation on the underlying message. So, in order to make the hash function family non-malleable (and then 1-more extractable), first we encode the message $\mathbf{s}$ (i.e., we compute $\mathbf{c} \leftarrow \mathsf{Enc}(\mathbf{s})$), using a non-malleable encoding scheme, $(\mathsf{Enc}, \mathsf{Dec})$, against affine functions, and then $v = (g^{\langle \mathbf{r}, \mathbf{c} \rangle}, g^{a\langle \mathbf{r}, \mathbf{c} \rangle})$ is computed as the output of the hash function. This approach can be viewed as a computational analogue of a non-malleable reduction, as previously used by [ADL14], and then formally presented by [ADKO15] (both works are in the information-theoretic setting).

It turns out that, in order to apply the methodology described above, a slightly stronger flavor of non-malleability is required for the underlying code, formalized in the present thesis as *randomness simulatable non-malleable codes* (RSS-NMC). The results of the present section are presented as follows: (1) we first construct 1-more wECRH assuming any randomness simulatable non-malleable encoding scheme, against affine tampering functions, and (2) we show how to construct such a code. Finally, we present Corollary 4.3.10 to summarize our overall construction, by putting all things together in a single statement.

4.3.1 1-more weakly extractable hash functions from RSS-NMC codes against affine functions

In this section we construct a collision resistant, 1-more extractable hash function family (wECRH). Before doing so, we present the notion of *randomness simulatable, strongly non-malleable codes* (RSS-NMC). This notion is stronger than strong non-malleability (cf. Definition 2.3.3) in the sense that besides simulating the tampered message, $\tilde{s}$, the simulator also needs to produce the randomness of the encoder, $\tilde{s}_{\mathsf{r}}$, such that the encoding of $\tilde{s}$ with randomness $\tilde{s}_{\mathsf{r}}$, produces the tampered codeword. To ease the presentation of RSS-NMC, we modify the syntax of non-malleable codes, so that the decoder, Dec, returns, not only the decoded message, $\tilde{s}$, but also the randomness string, $\tilde{s}_{\mathsf{r}}$, for the encoder Enc.[1]

Definition 4.3.1 (Randomness simulatable, strongly non-malleable code)**.** *Let* $(\mathsf{Enc}, \mathsf{Dec})$ *be a* (κ, ν)*-coding scheme and* $\mathcal{F}$ *be a family of functions* $f : \{0,1\}^{\nu} \to \{0,1\}^{\nu}$*. For every*

[1] It is possible to define RSS-NMC without modifying the operation of Dec at the expense of slightly complicating the definition of non-malleability. Due to the fact that our RSS-NMC construction conforms to the modified syntax, we opt for the simpler alternative.

$f \in \mathcal{F}$ and $s \in \{0,1\}^\kappa$, define the tampering experiment

$$\mathsf{Tamper}_s^f := \left\{ \begin{array}{c} c \leftarrow \mathsf{Enc}(s), \tilde{c} \leftarrow f(c), (\tilde{s}_\mathsf{r}, \tilde{s}) \leftarrow \mathsf{Dec}(\tilde{c}) \\ \textit{Output } \mathsf{same}^* \textit{ if } \tilde{c} = c, \textit{ and } (\tilde{s}_\mathsf{r}, \tilde{s}) \textit{ otherwise.} \end{array} \right\}$$

which is a random variable over the randomness of Enc *and* Dec*. An encoding scheme* $(\mathsf{Enc}, \mathsf{Dec})$ *is randomness simulatable, strongly non-malleable (*RSS-NM*), with respect to the function family $\mathcal{F}$, if for every $f \in \mathcal{F}$ and any s_0, $s_1 \in \{0,1\}^\kappa$, we have:*

$$\left\{\mathsf{Tamper}_{s_0}^f\right\}_{k\in\mathbb{N}} \approx \left\{\mathsf{Tamper}_{s_1}^f\right\}_{k\in\mathbb{N}}.$$

Here, "$\approx$" may refer to statistical, or computational, indistinguishability. For encoding schemes in the common reference string model, the definition is analogous.

Next we present our construction.

Construction 4.3.2 (1-more weakly extractable hash)**.** *Let $\mathcal{G}$ be a group-generation algorithm and let* $(\mathsf{Enc}, \mathsf{Dec})$ *be a (kt, kt')-encoding scheme, $t, t' = \mathrm{poly}(k)$. An instance of a $(kt, 2k)$-compressing hash function family $\mathcal{H} = (\mathsf{Gen}, h)$ is defined as follows:*

1. $\mathsf{Gen}(1^k)$*: sample* $(\mathbb{G}, g, p) \leftarrow \mathcal{G}(1^k)$, $(a, \mathbf{r}) \leftarrow \mathbb{Z}_p \times \mathbb{Z}_p^{t'}$*, where $p = |\mathbb{G}|$, and output* $z := (\mathbb{G}, g^{\mathbf{r}}, g^{a\mathbf{r}})$.

2. ***Hashing computation****: on input message* $\mathbf{s} = (s_1, \ldots, s_t)$*, sample* $\mathbf{s}_\mathsf{r} \leftarrow U_{\{0,1\}^{\mathrm{poly}(k)}}$*, and compute* $h_z(\mathbf{s}; \mathbf{s}_\mathsf{r}) := (g^{\langle \mathbf{r}, \mathbf{c}\rangle}, g^{\langle a\mathbf{r}, \mathbf{c}\rangle})$*, where* $\mathbf{c} \leftarrow \mathsf{Enc}(\mathbf{s}; \mathbf{s}_\mathsf{r})$.

For encoding schemes $(\mathsf{Init}, \mathsf{Enc}, \mathsf{Dec})$ *in the CRS model,* $\mathsf{Gen}(1^k)$ *outputs* (z, Σ)*, where* $\Sigma \leftarrow \mathsf{Init}(1^k)$.

For brevity, in what follows $\mathbb{G}$ will be omitted from the description of the hash, and we will use h to refer both to the program of the hash and the key of a specific element $(g^{\mathbf{r}}, g^{a\mathbf{r}})$, i.e., z is omitted. Also, notice that, the description of the hash function family defined above matches the one of Construction 4.2.3.

In what follows, we prove that Construction 4.3.2, which is a composition of an encoding scheme $(\mathsf{Enc}, \mathsf{Dec})$, with construction 4.2.3 (the extractable hash function by Bitansky et al. [BCCT12]), is a 1-more wECRH, assuming that $(\mathsf{Enc}, \mathsf{Dec})$, satisfies certain properties. Then, in Section 4.3.2, we instantiate $(\mathsf{Enc}, \mathsf{Dec})$ with the desired properties. Below, we prove that Construction 4.3.2 is collision resistant.

Lemma 4.3.3. *Let $\mathcal{G}$ be any group generation algorithm. Then, assuming the hardness of the discrete logarithm problem on $\mathcal{G}$, and the underlying encoding algorithm is injective, Construction 4.3.2 is collision resistant with respect to $\mathcal{G}$.*

Proof. In [BCCT12], the authors prove that the hash function family of Construction 4.2.3, $\mathcal{H}^*$, is collision resistant, assuming the hardness of the discrete logarithm problem. We recall that Construction 4.3.2 is a composition of $\mathsf{Enc}(\cdot)$ and $\mathcal{H}^*$. Following a simple fact that *any injective function composed with a collision resistant hash function still results in a collision resistant hash function* (composition in any order), we can conclude that the hash function family of Construction 4.3.2 is collision resistant, under the same assumption. □

In the following theorem, we prove that, under certain assumptions, Construction 4.3.2, is a 1-more wECRH.

Theorem 4.3.4. *Let $t(k), t'(k) = \text{poly}(k)$ and let $(\mathsf{Enc}, \mathsf{Dec})$ be an* RSS-NMC *(kt, kt')-encoding scheme, against $\mathcal{F}_{\mathsf{aff}}$ (cf. Definition 2.2.2), such that that for any message $\mathbf{s}$, $H_\infty(\mathsf{Enc}(\mathbf{s})) \geq k + \omega(\log k)$. Then, assuming $(t'+1)$-*KEA *and the hardness of* DLOG*, the hash function family of Construction 4.3.2, $\mathcal{H}$, is 1-more weakly extractable, with respect to $(\mathsf{Enc}, \mathsf{Dec})$.*

Proof. For $k \in \mathbb{N}$, let $(\mathsf{Enc}, \mathsf{Dec})$ be an RSS-NMC (kt, kt')-coding scheme, against $\mathcal{F}_{\mathsf{aff}}$, $t(k), t'(k) = \text{poly}(k)$, and let $\mathcal{H}$ be the $(kt, 2k)$-compressing, collision-resistant, hash function family of Construction 4.3.2, with respect to $(\mathsf{Enc}, \mathsf{Dec})$. Following Definition 4.2.1, we need to prove that for any PPT algorithm $\mathcal{A}_v$ with auxiliary input z_v, there exist extractor $\mathcal{E}^{\mathcal{H}}_{\mathcal{A}_v}$ and auxiliary input $z_{\mathcal{E}}$, such that for any PPT algorithm $\mathcal{A}_s$, any large k and every message $\mathbf{s} = (s_1, \ldots, s_t) \in \mathbb{Z}_p^t$,

$$\Pr_{h \leftarrow \mathcal{H}_k}\left[\mathsf{Exp}^{\mathbf{s},h}_{\mathcal{A}_v,\mathcal{A}_s,\mathcal{E}^{\mathcal{H}}_{\mathcal{A}_v}}(1, z_v, z_{\mathcal{E}}) = 1\right] \leq \mathsf{negl}(k). \tag{4.4}$$

Clearly, if $\mathcal{A}_v$ fails to produce a new valid hash value, or, if $\mathcal{A}_s$ fails to produce a valid pre-image for the new hash value, the experiment simply outputs 0, and there is no challenge for the extractor. Therefore, the interesting case is when $\mathcal{A}_v$ produces a valid hash value, say $\tilde{v}$, while having access to an element in the range of the hash, say v, and $\mathcal{A}_s$ produces a valid pre-image for $\tilde{v}$, while having access to the message $\mathbf{s}$,[2] the randomness used to compute the hash, $\tilde{\mathbf{s}}_{\mathsf{r}}$, and any other state information produced by $\mathcal{A}_v$, denoted as *st*. Hence, for the rest of the proof we assume $\tilde{v} \neq v$, and $(\tilde{\mathbf{s}}_{\mathsf{r}}, \tilde{\mathbf{s}})$ is a valid pre-image for $\tilde{v}$, i.e., $h(\tilde{\mathbf{s}}; \tilde{\mathbf{s}}_{\mathsf{r}}) = \tilde{v}$.

[2]Since we prove 1-extractability, $\mathbf{s}$ denotes a single message, which is a vector over $\mathbb{Z}_p^t$.

The description of an element in $\mathcal{H}$ is $h = (g^{r_1}, \ldots, g^{r_{t'}}, g^{ar_1}, \ldots, g^{ar_{t'}})$, while a hash value with respect to h is a pair of the form $v = (g^{r'}, g^{ar'})$, for some $r' \in \mathbb{Z}_p$; $\mathcal{A}_v$ receives as input (h, v) and the auxiliary input z_v. Having these in mind, the main steps in our proof are the following: (1) we prove that, by appending appropriately v to the description h, we receive a description h^* which is statistically close to an element in $\mathcal{H}^*$ (cf. Construction 4.2.3), for messages with $t'+1$ coordinates, (2) given $\mathcal{A}_v$ we define an attacker $\bar{\mathcal{A}}_v$ against $\mathcal{H}^*$, (3) since $\mathcal{H}^*$ is an extractable hash function family [BCCT12] assuming $(t'+1)$-KEA, we can use the extractor of $\mathcal{H}^*$ against $\bar{\mathcal{A}}_v$, to extract a valid pre-image of $\tilde{v}$ with respect to $\mathcal{H}^*$, (4) we prove that the extracted pre-image yields an affine transformation, which is the one that $\mathcal{A}_v$ applied to the original pre-image in order to construct $\tilde{v}$, and we use the RSS-NMC simulator to extract a valid pre-image for $\tilde{v}$, this time with respect to $\mathcal{H}$.

First we define an adversary, $\bar{\mathcal{A}}_v$, against the hash function family $\mathcal{H}^*$, of Construction 4.2.3, for vector messages with $t'+1$ coordinates. Concretely, we have,

$\bar{\mathcal{A}}_v\,(h^* = (g^{r_1}, \ldots, g^{r_{t'+1}}, g^{ar_1}, \ldots, g^{ar_{t'+1}}), z_v)$:

1. Set $h := (g^{r_1}, \ldots, g^{r_{t'}}, g^{ar_1}, \ldots, g^{ar_{t'}})$ and $v := (g^{r_{t'+1}}, g^{ar_{t'+1}})$.
2. **Output**: Execute $\mathcal{A}_v(h, v, z_v)$ and output $\mathcal{A}_v$'s output.

Here, $\bar{\mathcal{A}}_v$, first interprets the description of the hash function h^*, as (h, v), i.e., as a description of a hash function and hash value, with respect to $\mathcal{H}$, and then executes $\mathcal{A}_v(h, v, z_v)$. Under $(t'+1)$-KEA, $\mathcal{H}^*$ is an extractable hash function family [BCCT12]. Then, assuming h^* is indistinguishable from an element in $\mathcal{H}^*$ (we prove it below), there exists an extractor $\bar{\mathcal{E}}^{\mathcal{H}^*}_{\bar{\mathcal{A}}_v}$ with its auxiliary input $z_{\mathcal{E}}$, that extracts a valid pre-image for $\tilde{v}$, with respect to h^* (see Claim 4.3.5). We define the auxiliary input $z_{\mathcal{E}} := z_{\bar{\mathcal{E}}}$.

The extractor is defined below.

The extractor $\mathcal{E}^{\mathcal{H}}_{\mathcal{A}_v}$:
Input: $(h = (g^{\mathbf{r}}, g^{a\mathbf{r}}), v = (g^{r'}, g^{ar'}), z_{\mathcal{E}})$.

1. Set $h^* := (g^{\mathbf{r}}, g^{r'}, g^{a\mathbf{r}}, g^{ar'})$. Here, we interpret h^* as a description of hash function $h^* \in \mathcal{H}^*$, for vector messages with $t'+1$ coordinates.
2. Sample $(b_1, \ldots, b_{t'}, d) \leftarrow \bar{\mathcal{E}}^{\mathcal{H}^*}_{\bar{\mathcal{A}}_v}(h^*, z_{\mathcal{E}})$ and set $f := ((b_1, \ldots, b_{t'}), d) = (\mathbf{b}, d) \in \mathbb{Z}_p^{t'} \times \mathbb{Z}_p$.

3. Interpret f as an affine function that on input $(x_1, \ldots, x_{t'})$ outputs $(dx_1 + b_1, dx_2 + b_2, \ldots, dx_{t'} + b_{t'})$, and then sample $(\hat{\mathbf{s}}_\mathsf{r}, \hat{\mathbf{s}}) \leftarrow D_f^{\mathsf{aff}}$, where D_f^{aff} is the simulator of the underlying RSS-NMC, $(\mathsf{Enc}, \mathsf{Dec})$, parameterized by the affine function f.[3]

4. **Output**: $(\hat{\mathbf{s}}_\mathsf{r}, \hat{\mathbf{s}})$.

The extractor is defined with respect to any input v, still by the definition of the ℓ-more experiment, v is always a valid hash value, i.e., $v = h(\mathbf{s}) = (g^{\langle \mathbf{r}, \mathbf{c} \rangle}, g^{a\langle \mathbf{r}, \mathbf{c} \rangle})$, where $\mathbf{c} \leftarrow \mathsf{Enc}(\mathbf{s})$, for some message $\mathbf{s}$. Then, for any $\mathcal{A}_s$, and message $\mathbf{s}$, we are going to analyze the execution of $\mathsf{Exp}^{\mathbf{s},h}_{\mathcal{A}_v,\mathcal{A}_s,\mathcal{E}^{\mathcal{H}}_{\mathcal{A}_v}}(1, z_v, z_\mathcal{E})$. We first prove that with overwhelming probability, the following events happen:

- E_1: $h^*(b_1, \ldots, b_{t'}, d) = \tilde{v}$, where $\tilde{v}$ is the output of $\mathcal{A}_v$ on input (h, v).
- E_2: $\mathsf{Enc}(\tilde{\mathbf{s}}; \tilde{\mathbf{s}}_\mathsf{r}) = f(\mathbf{c})$, where $(\tilde{\mathbf{s}}_\mathsf{r}, \tilde{\mathbf{s}})$ is the output of $\mathcal{A}_s$.

We formalize those ideas in the following claims.

Claim 4.3.5. *Let h, $\mathcal{A}_v$, z_v, $\mathcal{E}^{\mathcal{H}}_{\mathcal{A}_v}$ and $z_\mathcal{E}$, be as they where defined above. Then, for any $\mathcal{A}_s$ and message $\mathbf{s}$, assuming $(t'+1)$-KEA and $H_\infty(\mathsf{Enc}(\mathbf{s})) \geq k + \omega(\log k)$, we have that $\Pr[\neg E_1] < \mathsf{negl}(k)$, over the randomness of the experiment $\mathsf{Exp}^{\mathbf{s},h}_{\mathcal{A}_v,\mathcal{A}_s,\mathcal{E}^{\mathcal{H}}_{\mathcal{A}_v}}(1, z_v, z_\mathcal{E})$.*

Proof. We recall that the experiment selects a uniform element $h = (g^{\mathbf{r}}, g^{a\mathbf{r}}) \in \mathcal{H}$, and then computes $v = h(\mathbf{s}) = (g^{\langle \mathbf{r}, \mathbf{c} \rangle}, g^{a\langle \mathbf{r}, \mathbf{c} \rangle})$, where $\mathbf{c} \leftarrow \mathsf{Enc}(\mathbf{s}; \mathbf{s}_\mathsf{r})$. In order to show that $(b_1, \ldots, b_{t'}, d)$ is a valid pre-image for $\tilde{v}$ with respect to h^*, we prove that $h^* := (g^{\mathbf{r}}, g^{\langle \mathbf{r}, \mathbf{c} \rangle}, g^{a\mathbf{r}}, g^{a\langle \mathbf{r}, \mathbf{c} \rangle})$ is indistinguishable from an element in $\mathcal{H}^*$. Concretely, by assumption we have that $H_\infty(\mathsf{Enc}(\mathbf{s})) \geq k + \omega(\log k)$. Since the randomness of the encoder is independent of $Z = (z_v, h)$, (those values are fixed before sampling randomness for the hash), we have $H_\infty(\mathsf{Enc}(\mathbf{s}) \mid Z) \geq k + \omega(\log k)$, and therefore, $\tilde{H}_\infty(\mathsf{Enc}(\mathbf{s}) \mid Z) \geq k + \omega(\log k)$. By the above argument, the Left-Over Hash Lemma (Lemma 2.1.11) and the universality of the inner product function (Lemma 2.1.13), the distribution $\langle \mathbf{r}, \mathbf{c} \rangle$ is statistically close to uniform, under the partial execution of the "1-more" experiment, i.e., up to the point we execute the extractor. This implies that the distribution $(g^{\mathbf{r}}, g^{\langle \mathbf{r}, \mathbf{c} \rangle}, g^{a\mathbf{r}}, g^{a\langle \mathbf{r}, \mathbf{c} \rangle})$ is statistically close to $(g^{\mathbf{r}}, g^{r'}, g^{a\mathbf{r}}, g^{ar'})$, for a uniformly random r', and thus $\Pr[E_1]$ differs by a negligible quantity under the two distributions, denoted as D_1 and D_2, respectively.

[3] Given that the underlying encoding scheme is an RSS-NMC, defining the simulator is straightforward: given f, the simulator executes Tamper^f_0, where 0 denotes the zero message, and outputs the output of the experiment.

In [BCCT12] the authors showed that $\mathcal{H}^*$ is extractable assuming $(t'+1)$-KEA, which implies that the extractor $\bar{\mathcal{E}}^{\mathcal{H}^*}_{\bar{\mathcal{A}}_v}$ extracts a pre-image for $\tilde{v}$, with respect to h^*, i.e., the event E_1 happens with overwhelming probability, under the distribution D_2. Therefore, by the statistical closeness between D_1 and D_2, we conclude that the event E_1 happens with overwhelming probability, under the distribution D_1. This completes the proof of the claim. □

Claim 4.3.6. *Let h, $\mathcal{A}_v$, z_v, $\mathcal{E}^{\mathcal{H}}_{\mathcal{A}_v}$ and $z_{\mathcal{E}}$, be as they where defined above. Then, for any $\mathcal{A}_s$ and message $\mathbf{s}$, $\Pr[\neg E_2] < \mathsf{negl}(k)$ over the randomness of the experiment $\mathsf{Exp}^{\mathbf{s},h}_{\mathcal{A}_v,\mathcal{A}_s,\mathcal{E}^{\mathcal{H}}_{\mathcal{A}_v}}(1, z_v, z_{\mathcal{E}})$.*

Proof. We have $\Pr[\neg E_2] = \Pr[\neg E_2 \wedge E_1] + \Pr[\neg E_2 \wedge \neg E_1]$, and by the previous claim we have $\Pr[\neg E_1] < \mathsf{negl}(k)$. Therefore, it suffices to show that $\Pr[\neg E_2 \wedge E_1] < \mathsf{negl}(k)$. Thus, below we focus on the event $\neg E_2 \wedge E_1$.

Let $\bar{h}^* := (g^{\mathbf{r}}, g^{a\mathbf{r}}) = h$. We treat $\bar{h}^*$ as an element in $\mathcal{H}^*$ operating over in $\mathbb{Z}_p^{t'}$, i.e., for any message $\mathbf{s} \in \mathbb{Z}_p^{t'}$, $\bar{h}^*(\mathbf{s}) = (g^{\langle \mathbf{r},\mathbf{s}\rangle}, g^{a\langle \mathbf{r},\mathbf{s}\rangle})$. By the definition of $\mathcal{H}$ and $\mathcal{H}^*$, for any message $\mathbf{s} \in \mathbb{Z}_p^{t'}$, we have that,

$$h(\mathbf{s}; \mathbf{s}_{\mathsf{r}}) = \bar{h}^*(\mathsf{Enc}(\mathbf{s}; \mathbf{s}_{\mathsf{r}})). \tag{4.5}$$

Now, recall that, in order to exclude the trivial cases, we have assumed that the output of $\mathcal{A}_s$, $(\tilde{\mathbf{s}}_{\mathsf{r}}, \tilde{\mathbf{s}})$, is a valid pre-image for $\tilde{v}$, i.e., $h(\tilde{\mathbf{s}}; \tilde{\mathbf{s}}_{\mathsf{r}}) = \tilde{v}$, which by Equation 4.5 implies that $\bar{h}^*(\mathsf{Enc}(\tilde{\mathbf{s}}; \tilde{\mathbf{s}}_{\mathsf{r}})) = \tilde{v}$. Whenever E_1 takes place we have that $h^*(b_1, \ldots, b_{t'}, d) = \tilde{v}$, and by the definition of $h^* = (g^{\mathbf{r}}, g^{\langle \mathbf{r},\mathbf{c}\rangle}, g^{a\mathbf{r}}, g^{a\langle \mathbf{r},\mathbf{c}\rangle})$, it is implied that $\left(g^{\langle \mathbf{r}, d\mathbf{c}+\mathbf{b}\rangle}, g^{a\langle \mathbf{r}, d\mathbf{c}+\mathbf{b}\rangle}\right) = \left(g^{\langle \mathbf{r}, f(\mathbf{c})\rangle}, g^{a\langle \mathbf{r}, f(\mathbf{c})\rangle}\right) = \tilde{v}$. From the last relation we receive that $\bar{h}^*(f(\mathbf{c})) = \tilde{v}$. Now, recall that, by Lemma 4.2.4, the family $\mathcal{H}^*$ is collision resistant, assuming the hardness of DLOG. Assuming that $\Pr[\neg E_2 \wedge E_1]$ happens with non-negligible probability, we have that $\mathsf{Enc}(\tilde{\mathbf{s}}; \tilde{\mathbf{s}}_{\mathsf{r}}) \neq f(\mathbf{c})$, while $\bar{h}^*(\mathsf{Enc}(\tilde{\mathbf{s}}_{\mathsf{r}}, \tilde{\mathbf{s}})) = \bar{h}^*(f(\mathbf{c}))$, with non-negligible probability. Thus, by simulating the 1-more experiment, we find such a collision with non-negligible probability, as long as $\neg E_2 \wedge E_1$ happens. This reaches a contradiction and completes the proof of the claim. □

Finally, we argue that, with overwhelming probability the output of the extractor, $(\hat{\mathbf{s}}_{\mathsf{r}}, \hat{\mathbf{s}})$, is a valid pre-image for $\tilde{v}$, i.e., $h(\hat{\mathbf{s}}; \hat{\mathbf{s}}_{\mathsf{r}}) = \tilde{v}$. Here, recall that $(\hat{\mathbf{s}}_{\mathsf{r}}, \hat{\mathbf{s}})$ is the output of the simulator D^{aff}_f, of the underlying RSS-NMC. Concretely, we prove the following claim.

Claim 4.3.7. *Let h, $\mathcal{A}_v$, z_v, $\mathcal{E}^{\mathcal{H}}_{\mathcal{A}_v}$ and $z_{\mathcal{E}}$, be as they where defined above, and assume that* $(\mathsf{Enc}, \mathsf{Dec})$ *is an* RSS-NMC*. Then, for any $\mathcal{A}_s$ and message* $\mathbf{s}$*,* $\Pr[(\tilde{\mathbf{s}}_{\mathsf{r}}, \tilde{\mathbf{s}}) \neq (\hat{\mathbf{s}}_{\mathsf{r}}, \hat{\mathbf{s}})] < \mathsf{negl}(k)$*, over the randomness of the experiment* $\mathsf{Exp}^{\mathbf{s},h}_{\mathcal{A}_v,\mathcal{A}_s,\mathcal{E}^{\mathcal{H}}_{\mathcal{A}_v}}(1, z_v, z_{\mathcal{E}})$*.*

Proof. As above, we can upper bound $\Pr[(\tilde{\mathbf{s}}_{\mathsf{r}}, \tilde{\mathbf{s}}) \neq (\hat{\mathbf{s}}_{\mathsf{r}}, \hat{\mathbf{s}})]$, by $\Pr[(\tilde{\mathbf{s}}_{\mathsf{r}}, \tilde{\mathbf{s}}) \neq (\hat{\mathbf{s}}_{\mathsf{r}}, \hat{\mathbf{s}}) \wedge E_1 \wedge E_2] + \Pr[(\tilde{\mathbf{s}}_{\mathsf{r}}, \tilde{\mathbf{s}}) \neq (\hat{\mathbf{s}}_{\mathsf{r}}, \hat{\mathbf{s}}) \wedge \neg E_1 \wedge E_2] + \Pr[(\tilde{\mathbf{s}}_{\mathsf{r}}, \tilde{\mathbf{s}}) \neq (\hat{\mathbf{s}}_{\mathsf{r}}, \hat{\mathbf{s}}) \wedge E_1 \wedge \neg E_2] + \Pr[(\tilde{\mathbf{s}}_{\mathsf{r}}, \tilde{\mathbf{s}}) \neq (\hat{\mathbf{s}}_{\mathsf{r}}, \hat{\mathbf{s}}) \wedge \neg E_1 \wedge \neg E_2]$. By the above claims, we have $\Pr[\neg E_1], \Pr[\neg E_2] < \mathsf{negl}(k)$. Therefore, in order to show the current claim, it suffices to prove that $\Pr[(\tilde{\mathbf{s}}_{\mathsf{r}}, \tilde{\mathbf{s}}) \neq (\hat{\mathbf{s}}_{\mathsf{r}}, \hat{\mathbf{s}}) \wedge E_1 \wedge E_2] < \mathsf{negl}(k)$. Thus, in the rest of the proof we focus on the event $(\tilde{\mathbf{s}}_{\mathsf{r}}, \tilde{\mathbf{s}}) \neq (\hat{\mathbf{s}}_{\mathsf{r}}, \hat{\mathbf{s}}) \wedge E_1 \wedge E_2$.

By Claims 4.3.5, 4.3.6, we have that $\mathsf{Enc}(\tilde{\mathbf{s}}; \tilde{\mathbf{s}}_{\mathsf{r}}) = f(\mathbf{c})$ and $\bar{h}^*(f(\mathbf{c})) = \tilde{v}$, with overwhelming probability. Moreover, in order to exclude the trivial cases with respect to the task that needs to accomplished by the extractor, we have assumed that $\tilde{v} \neq v = \bar{h}^*(\mathbf{c})$. Since $v \neq \tilde{v}$, we have $\mathbf{c} \neq f(\mathbf{c})$; since $f(\mathbf{c}) = \mathsf{Enc}(\tilde{\mathbf{s}}; \tilde{\mathbf{s}}_{\mathsf{r}})$, $f(\mathbf{c})$ is a valid codeword. Thus, by the previous observations we have that $\mathsf{Dec}(f(\mathbf{c})) \neq \perp$. Moreover, by the security of the RSS-NMC scheme, we know that with overwhelming probability $\Pr[\mathsf{Enc}(\hat{\mathbf{s}}_{\mathsf{r}}; \hat{\mathbf{s}}) \neq f(\mathbf{c})] < \mathsf{negl}(k)$. Since Enc is injective, this implies that $\Pr[(\tilde{\mathbf{s}}_{\mathsf{r}}, \tilde{\mathbf{s}}) \neq (\hat{\mathbf{s}}_{\mathsf{r}}, \hat{\mathbf{s}}) \wedge E_1 \wedge E_2] < \mathsf{negl}(k)$, and the proof of the claim is complete. □

Note, that, for any $\mathcal{A}_s$ and message $\mathbf{s}$, the experiment $\mathsf{Exp}^{\mathbf{s},h}_{\mathcal{A}_v,\mathcal{A}_s,\mathcal{E}^{\mathcal{H}}_{\mathcal{A}_v}}(1, z_v, z_{\mathcal{E}})$ outputs 1 if $h(\tilde{\mathbf{s}}; \tilde{\mathbf{s}}_{\mathsf{r}}) = \tilde{v} \ \wedge \ \tilde{v} \neq v \ \wedge \ h(\hat{\mathbf{s}}; \hat{\mathbf{s}}_{\mathsf{r}}) \neq \tilde{v}$. By the above claims, $h(\hat{\mathbf{s}}; \hat{\mathbf{s}}_{\mathsf{r}}) \neq \tilde{v}$, with negligible probability, assuming that $h(\tilde{\mathbf{s}}; \tilde{\mathbf{s}}_{\mathsf{r}}) = \tilde{v}$ and $\tilde{v} \neq v$. Therefore, we conclude that

$$\Pr_{h \leftarrow \mathcal{H}_k}\left[\mathsf{Exp}^{\mathbf{s},h}_{\mathcal{A}_v,\mathcal{A}_s,\mathcal{E}^{\mathcal{H}}_{\mathcal{A}_v}}(1, z_v, z_{\mathcal{E}}) = 1\right] \leq \mathsf{negl}(k).$$

This completes the proof of the theorem. □

4.3.2 Constructing RSS-NM codes

In the present section, we construct an RSS-NMC as required by the 1-more wECRH of Construction 4.3.2. A simplified version of the proposed construction is presented in the next paragraph.

For any message s, our encoder secret shares s into (s_1, s_2), using a two-out-of-two, additive, secret sharing scheme, and outputs $\mathbf{c} = (s_1, s_2, s_1^2, s_2^2)$. Then, for any codeword $\mathbf{c} = (s_1, s_2, s_1', s_2')$, decoding proceeds as follows: if $s_i^2 = s_i'$, for $i \in \{1, 2\}$, the decoder outputs $s_1 + s_2$, otherwise, it outputs $\perp$. An affine tampering function, f, against the code is described by the pair $(\mathbf{b}, d)$, where $\mathbf{b} = (b_1, b_2, b_3, b_4)$, and the application of f on a codeword $\mathbf{c}$, yields the codeword $d \cdot \mathbf{c} + \mathbf{b}$. Briefly, security of the scheme is proven

by considering the following cases. If $d = 0$, then the tampered codeword is completely overwritten by $\mathbf{b}$, and clearly, the output of the decoder depends only on $\mathbf{b}$. If $d \neq 0$, then, we prove that, either the attack leaves the codeword intact, i.e., $d = 1, \mathbf{b} = \mathbf{0}$, or the decoding of the tampered codeword is equal to $\perp$, with overwhelming probability.

Construction 4.3.8 (An RSS-NMC against $\mathcal{F}_{\mathsf{aff}}$)**.** *For any $k \in \mathbb{N}$, $t = \mathrm{poly}(k)$, we define a $(kt, (2t+4)k)$-encoding scheme $(\mathsf{Init}, \mathsf{Enc}, \mathsf{Dec})$ in the CRS model,*[4] *as follows:*

- $\mathsf{Init}(1^k)$*: sample a k-bit prime $p \in (2^{k-1}, 2^k)$ and set $\Sigma := p$.*
- $\mathsf{Enc}(\Sigma, \cdot)$*: let $\mathbf{s} = (s_1, \ldots, s_t) \in \mathbb{F}_p^t$ be the input to* Enc*. Sample two random field elements $v, r \leftarrow \mathbb{F}_p$, and then output*

$$\mathbf{c} = \left(v, v^2, r, r^2, u_1, u_1^2, \ldots, u_t, u_t^2\right) \in \mathbb{F}_p^{2t+4},$$

  where $u_i = s_i - r$ for $i \in [t]$.
- $\mathsf{Dec}(\Sigma, \cdot)$*: on input $\mathbf{c} = (v, \bar{v}, r, \bar{r}, u_1, \bar{u}_1, \ldots, u_t, \bar{u}_t)$, the decoder checks whether $\bar{v} = v^2, \bar{r} = r^2$, and $\bar{u}_i = u_i^2$ for all $i \in [t]$. If so, then it outputs $(v, r, u_1 + r, u_2 + r, \ldots, u_t + r)$, otherwise, outputs $\perp$.*

All operations are performed modulo p. We also consider the deterministic version of Enc *by allowing the randomness to be given on the input. In that case we have $\mathbf{c} \leftarrow \mathsf{Enc}(\Sigma, \mathbf{s}_\mathsf{r}, \mathbf{s})$, where $\mathbf{s}_\mathsf{r} = (v, r)$.*

Notice, that, the randomness employed by the above construction is $2k$, independently of the message length.

Theorem 4.3.9. *The code of Construction 4.3.8 is randomness simulatable, strongly non-malleable (Definition 4.3.1), against $\mathcal{F}_{\mathsf{aff}}$. In addition, for any message $\mathbf{s}$, $H_\infty(\mathsf{Enc}(\mathbf{s})) \geq k + \omega(\log k)$.*

Proof. Let $f \in \mathcal{F}_{\mathsf{aff}}$ be a tampering function against the code, defined by the pair $(\mathbf{b}, d) \in \mathbb{F}_p^{2t+4} \times \mathbb{F}_p$, where $\mathbf{b} = (b_v, b'_v, b_r, b'_r, b_1, b'_1, \ldots, b_t, b'_t)$ and $f(\mathbf{c}) = d\mathbf{c} + \mathbf{b}$. Following the definition of RSS-NMC, we need to show that for any pair of messages $\mathbf{s}_0$, $\mathbf{s}_1$,

$$\left\{\left(\Sigma, \mathsf{Tamper}_{\mathbf{s}_0}^{\Sigma,f}\right)\right\}_{k\in\mathbb{N}} \approx \left\{\left(\Sigma, \mathsf{Tamper}_{\mathbf{s}_1}^{\Sigma,f}\right)\right\}_{k\in\mathbb{N}},$$

[4] Note that the CRS is not essential for this encoding, but for simplicity we describe the code in this model.

i.e., the output of the tampering experiment is independent of the message and decidable only by inspecting the function f.

Now, recall that any codeword has the following form

$$\mathbf{c} = \left(v, v^2, r, r^2, u_1, u_1^2, \ldots, u_t, u_t^2\right).$$

We then consider the following cases:

1. ($d = 0$): such an attack completely overwrites $\mathbf{c}$ with $\mathbf{b}$, which is independent of the original message, $\mathbf{s}$. Therefore, for any of the two messages $\mathbf{s}_0$ and $\mathbf{s}_1$, the tampering experiment outputs $\mathsf{Dec}(\tilde{\mathbf{c}}) = \mathsf{Dec}(\mathbf{b})$, independently of the message.

2. ($d = 1$ and $\mathbf{b} = \mathbf{0}$): this attack leaves the codeword intact for both experiments, and the output of the experiment should be same^*, independently of the message.

3. ($d = 1$ and $\mathbf{b} \neq \mathbf{0}$): assume that $(b_z, b'_z) \neq (0, 0)$ for some $z \in [t] \cup \{v, r\}$.[5] We know that the tampering experiment outputs a non-bottom value only if the following equation is satisfied: $(du_z + b_z)^2 = du_z^2 + b'_z$. (For simplicity, and in order to cover the cases of r, v, we denote $u_r := r, u_v := v$). We argue that this happens with negligible probability, which implies that both experiments output $\perp$, with overwhelming probability.

   By expanding the equation and plugging in $d = 1$, we have $2b_z u_z + b_z^2 - b'_z = 0$. We consider two cases with respect to (b_z, b'_z): (a) $b_z \neq 0$; (b) $b_z = 0, b'_z \neq 0$. For case (b), clearly, the output in both experiments is $\perp$. Regarding case (a), the equation is linear, and therefore, it possesses at most one solution. By our choice of u_z (recall that $u_z = s_z - r$ where r is a random field element), we know that its marginal distribution is uniform, and thus the probability that the equation is satisfied is at most $1/p$.

4. $d \in \mathbb{F}_p \setminus \{0, 1\}$: as above, we argue that the two experiments output $\perp$ with overwhelming probability. As we discussed above, the tampering experiments output non-bottom values only if $(du_z + b_z)^2 = du_z^2 + b'_z$ is satisfied, for all $z \in [t] \cup \{r, v\}$. This probability can be upper bounded by the probability that a particular equation is satisfied. Without loss of generality, we consider the equation $d(d-1)u_1^2 + 2db_1u_1 + b_1^2 - b'_1 = 0$. Since $a \in \mathbb{F}_p \setminus \{0, 1\}$, the above equation is of degree 2, and by the Schwartz-Zippel lemma, it possesses at most two solutions. As

[5] Here, we treat v, r, as special symbols, not as integers.

we argued above, the marginal distribution of u_1 is uniformly random. Thus, the probability that the equation is satisfied is at most $2/p$.

The above case analysis covers all possibilities for $(\mathbf{b}, d)$, and the proof of the first part of the theorem is complete. For the second part, by construction we have that any codeword consists of two random field elements (r, v), having length $2k$, and thus $H_\infty(\mathsf{Enc}(\mathbf{s})) = 2k > k + \omega(\log k)$. This completes the proof of the theorem. □

4.3.3 Our resulting instantiation

By plugging Construction 4.3.8, as the underlying encoding scheme to Construction 4.3.2, we receive the following corollary.

Corollary 4.3.10. *Under the* DLOG *assumption and* t-KEA*, there exists an explicit* 1-*more* wECRH*.*

Proof. Let $(\mathsf{Init}, \mathsf{Enc}, \mathsf{Dec})$ be the $(kt, (2t+4)k)$ RSS-NMC of Construction 4.3.8. Then we construct $\mathcal{H}_k$ by plugging in $(\mathsf{Init}, \mathsf{Enc}, \mathsf{Dec})$, as the underlying encoding scheme to the hash function family of Construction 4.3.2. Clearly, by Lemma 4.3.3, $\mathcal{H}_k$ is collision resistant as the underlying encoder is injective. By Theorem 4.3.9, the underlying encoding scheme is an RSS-NMC against $\mathcal{F}_{\mathsf{aff}}$, and moreover, for any message $\mathbf{s}$, $H_\infty(\mathsf{Enc}(\mathbf{s})) \geq k + \omega(\log k)$. Thus, by Theorem 4.3.4, $\mathcal{H}_k$ is a 1-more wECRH. This concludes the proof of this corollary. □

4.3.4 Constructing ℓ-more weakly extractable hash under KEA

In the "ℓ-more" setting, the attacker is given $v_1, \ldots, v_\ell$, precomputed hash values, and produces a new hash value $\tilde{v}$. Having the techniques from the "1-more" setting, we can argue that any attack against $\tilde{v}$ (in the ℓ-more setting), can be reduced to an affine attack against the codewords $c_1, \ldots, c_\ell$, that are related to $v_1, \ldots, v_\ell$, respectively. In order to construct ℓ-more wECRH, for $\ell > 1$, we generalize the notion of RSS-NMC, for multiple codewords. The generalization is a straightforward extension of Definition 4.3.1, where the tampering function receives $\ell \in \mathbb{N}$ codewords and the simulator needs to recover the message and the randomness, in case the output of the tampering function is not among the given codewords. The formal is definition is given below.

Definition 4.3.11 (Multi-codeword RSS-NMC)**.** *Let* $(\mathsf{Enc}, \mathsf{Dec})$ *be a* (κ, ν)*-coding scheme and* $\mathcal{F}$ *be a family of functions* $f : \{0,1\}^\nu \to \{0,1\}^\nu$*. For every* $f \in \mathcal{F}$ *and* $\mathbf{s} = (s_1, \ldots, s_\ell) \in$

$(\{0,1\}^\kappa)^\ell$, *define the tampering experiment*

$$\mathsf{MultiTamper}^f_{\mathbf{s}} := \left\{ \begin{array}{c} c_i \leftarrow \mathsf{Enc}(s_i), i \in [\ell], \tilde{c} \leftarrow f(c_1, \ldots, c_\ell), (\tilde{s}_\mathsf{r}, \tilde{s}) = \mathsf{Dec}(\tilde{c}) \\ \textit{Output } \mathsf{same}^* \textit{ if } \exists i : \tilde{c} = c_i, \textit{ and } (\tilde{s}_\mathsf{r}, \tilde{s}) \textit{ otherwise.} \end{array} \right\}$$

which is a random variable over the randomness of Enc *and* Dec. *An encoding scheme* $(\mathsf{Enc}, \mathsf{Dec})$ *is multi-codeword, randomness simulatable, strongly non-malleable, with respect to the function family* $\mathcal{F}$, *if for every* $f \in \mathcal{F}$ *and any* $\mathbf{s}_0$, $\mathbf{s}_1 \in (\{0,1\}^\kappa)^\ell$, *we have*

$$\left\{\mathsf{MultiTamper}^f_{\mathbf{s}_0}\right\}_{k\in\mathbb{N}} \approx \left\{\mathsf{MultiTamper}^f_{\mathbf{s}_1}\right\}_{k\in\mathbb{N}}.$$

Here, "$\approx$*" may refer to statistical, or computational, indistinguishability. For encoding schemes in the common reference string model, the definition is analogous.*

Clearly, for $\ell = 1$, the notion of multi-codeword RSS-NMC matches the one presented in Definition 4.3.1. In order to construct, ℓ-more wECRH, for $\ell > 1$, we need an RSS-NMC, for the following function class.

Definition 4.3.12 (The function class $\bar{\mathcal{F}}^\ell_{\mathsf{aff}}$)**.** *We define the following function class*

$$\bar{\mathcal{F}}^\ell_{\mathsf{aff}} = \{f(x_1, \ldots, x_\ell) = f_1(x_1) + \ldots + f_\ell(x_\ell) \mid f_i \in \mathcal{F}_{\mathsf{aff}}\}.$$

We present the following lemma.

Lemma 4.3.13. *The encoding scheme of Construction 4.3.8,* $(\mathsf{Enc}, \mathsf{Dec})$*, is a multi-codeword* RSS-NMC *against* $\bar{\mathcal{F}}^\ell_{\mathsf{aff}}$, *for* $\ell > 1$.

Proof. Recall that, any $f \in \mathcal{F}_{\mathsf{aff}}$ produces a valid, new codeword, $\tilde{c}$, with respect to $(\mathsf{Enc}, \mathsf{Dec})$, only when $\tilde{c}$ is independent of the original codeword. Moreover, the output of Tamper^f_s is same^*, only if $f = (\mathbf{0}, 1)$, where Tamper is the tampering experiment of Definition 4.3.1 (for brevity we omit the CRS). Based on those facts and using similar arguments with the proof of Theorem 4.3.9, it is not hard to see that when considering multiple codewords/messages and the encoding $(\mathsf{Enc}, \mathsf{Dec})$ against $\mathcal{F}_{\mathsf{aff}}$, any tampering function $f = (f_1, \ldots, f_\ell) \in \bar{\mathcal{F}}^\ell_{\mathsf{aff}}$, makes the tampering experiment of Definition 4.3.11 to output same^*, only if a single $f_m = (\mathbf{b}_m, d_m)$ is the identity function, i.e., $\mathbf{b}_m = \mathbf{0}$, $d_m = 1$, while the remaining functions are the zero-functions, i.e., $\mathbf{b}_j = \mathbf{0}$, $d_j = 0$, $j \in [\ell]\backslash\{m\}$. We will refer to such a tampering function using the term *projection function*. Now, given a tampering function $f = (f_1, \ldots, f_\ell) \in \bar{\mathcal{F}}^\ell_{\mathsf{aff}}$, and messages $\mathbf{s} = (s_1, \ldots, s_\ell)$, we can easily construct $f' \in \mathcal{F}_{\mathsf{aff}}$, s', for which $\mathsf{MultiTamper}^f_{\mathbf{s}} = \mathsf{Tamper}^{f'}_{s'}$, where MultiTamper is the tampering experiment of Definition 4.3.11. If f' is a projection function with respect to index

m, we set $f' = (\mathbf{0}, 1)$ and $s' = s_m$, and clearly, both experiments output same^*. Otherwise, assuming $f_1 = (\mathbf{b}_1, d_1)$, we compute $\mathbf{b} = \sum_{i=2}^{\ell} f_i(\mathsf{Enc}(s_i))$ and we set $f' = (\mathbf{b} + \mathbf{b}_1, d_1)$, $s' = s_1$, and we can prove, exactly as in the proof of Theorem 4.3.9, that either the tampered codeword is independent of the original, and the outputs for both experiments are decidable by inspecting the tampering function, or both experiments output $\perp$. Thus, assuming that we can distinguish between $\mathsf{MultiTamper}^{f}_{\mathbf{s}_0}$ and $\mathsf{MultiTamper}^{f}_{\mathbf{s}_1}$, for some $f \in \bar{\mathcal{F}}^{\ell}_{\mathsf{aff}}$ and messages $\mathbf{s}_0$, $\mathbf{s}_1$, we can construct $f' \in \mathcal{F}_{\mathsf{aff}}$, s'_0, s'_1, and distinguish between $\mathsf{Tamper}^{f'}_{s'_0}$ and $\mathsf{Tamper}^{f'}_{s'_1}$, breaking the security of the RSS-NMC of Construction 4.3.8. $\square$

In the "ℓ-more" setting the attacker receives $v_i = (g^{\langle \mathbf{r}, \mathbf{c}_i \rangle}, g^{a\langle \mathbf{r}, \mathbf{c}_i \rangle})$, $i \in [\ell]$, and constructs a valid hash $\tilde{v}$. The proof of Theorem 4.3.4 easily extends to the "ℓ-more" setting by proving that $\tilde{v} = \left(g^{\langle \mathbf{r}, \sum_{i=1}^{\ell} f_i(\mathbf{c}_i) \rangle}, g^{a\langle \mathbf{r}, \sum_{i=1}^{\ell} f_i(\mathbf{c}_i) \rangle}\right)$, where $(f_1, \ldots, f_\ell) \in \bar{\mathcal{F}}^{\ell}_{\mathsf{aff}}$, and we achieve extractability using the simulator of the underlying RSS-NMC, for multiple codewords. Thus, we are able to show the following theorem.

Theorem 4.3.14. *Under the* DLOG *assumption and* t-KEA*, Construction 4.3.2, instantiated with the encoding scheme of Construction 4.3.8, is an* ℓ*-more* wECRH*.*

The proof is essentially the same as that of Theorem 4.3.4, as we discussed above.

4.3.5 Leakage-resilient ℓ-more wECRH under KEA

In the current section, we present the notion of *leakage-resilient* ℓ-more wECRH and we prove that Construction 4.3.2 satisfies this notion. The proposed definition is along the lines of Definition 4.2.1.

Definition 4.3.15 (ℓ-more weakly extractable, leakage-resilient hash function families)**.** *Let* $\ell, \lambda \in \mathbb{N}$ *and let* $g : \{0,1\}^* \to \{0,1\}^{\lambda}$*. An efficiently samplable hash function ensemble* $\mathcal{H} = \{\mathcal{H}_k\}_{k \in \mathbb{N}}$*, is* ℓ*-more weakly extractable against* λ *bits of leakage, if for any* PPT *algorithm* $\mathcal{A}_v$ *and any* $z_v \in \{0,1\}^{\mathrm{poly}(k)}$*, there exist a* PPT *extractor* $\mathcal{E}^{\mathcal{H}}_{\mathcal{A}_v}$ *and* $z_{\mathcal{E}} \in \{0,1\}^{\mathrm{poly}(k)}$*, such that for all* PPT *algorithms* $\mathcal{A}_s$*, any large* $k \in \mathbb{N}$ *and any vector of messages* $\mathbf{s} = (s_1, \ldots, s_\ell)$*,*

$$\Pr_{h \leftarrow \mathcal{H}_k}\left[\mathsf{Exp}^{\mathbf{s},h}_{\mathcal{A}_v, \mathcal{A}_s, \mathcal{E}_{\mathcal{A}_v}}(\ell, \lambda, z_v, z_{\mathcal{E}}) = 1\right] \leq \mathsf{negl}(k),$$

where,

$\mathsf{Exp}^{\mathbf{s},h}_{\mathcal{A}_v,\mathcal{A}_s,\mathcal{E}_{\mathcal{A}_v}}(\ell,\lambda,z_v,z_{\mathcal{E}})$:
$\tau_i \leftarrow \{0,1\}^{\mathrm{poly}(k)}, v_i = h(s_i;\tau_i), i \in [\ell]$ *(* **hash computation** *)*
$\mathbf{t} = (\tau_1,\ldots,\tau_\ell), \mathbf{v} = (v_1,\ldots,v_\ell)$
$(\tilde{v}, st) \leftarrow \mathcal{A}_v(h,\mathbf{v},g(\mathbf{t}),z_v)$ *(* **hash tampering** *)*
$(\hat{\tau},\hat{s}) \leftarrow \mathcal{E}_{\mathcal{A}_v}(h,\mathbf{v},z_{\mathcal{E}})$ *(* **pre-image extraction** *)*
$(\tilde{\tau},\tilde{s}) \leftarrow \mathcal{A}_s(h,\mathbf{t},\mathbf{s},st)$ *(* **pre-image tampering** *)*

If $h(\tilde{s};\tilde{\tau}) = \tilde{v} \wedge \forall i : \tilde{v} \neq v_i \wedge h(\hat{s};\hat{\tau}) \neq \tilde{v}$, *return 1*
otherwise, return 0

The main requirement for proving security of Construction 4.3.2, is that, the underlying encoding scheme, $(\mathsf{Enc},\mathsf{Dec})$, is an RSS-NMC against affine functions, and $\mathsf{Enc}(s)$ has sufficient entropy. Given a scheme that satisfies such properties, we prove that Construction 4.3.2 is an ℓ-more wECRH under the t-KEA and DLOG (cf. Theorem 4.3.2).

In the following theorem, we reduce the ℓ-more extractability in the presence of leakage, to standard, i.e., ℓ-more extractability without leakage, under the same assumptions. Briefly, the main idea behind the proof presented below, is that if $\mathsf{Enc}(s;\tau)$ has sufficient entropy even given bounded leakage over τ, then v is indistinguishable from uniform due to the universality property (cf. Definition 2.1.13) of the inner product, and we manage to reduce ℓ-more extractability in the presence of leakage, to ℓ-more extractability without leakage, using a series of hybrids. Our result is formally presented in the following theorem.

Theorem 4.3.16. *For $k \in \mathbb{N}$, let $\mathcal{H}$ be the ℓ-more* wECRH *family of Construction 4.3.2 instantiated with an* RSS-NMC *encoding scheme,* $(\mathsf{Enc},\mathsf{Dec})$*, such that for any message s, $\mathbf{H}_\infty(\mathsf{Enc}(s)) \geq \lambda + k + \omega(\log k)$. Then, $\mathcal{H}$ is an ℓ-more* wECRH *against λ bits of non-adaptive leakage.*

Proof. We prove the needed for the 1-more case (the ℓ-more case is identical) using a series of hybrid experiments that we describe below and they are depicted in Figure 4.1.

- For any $g : \{0,1\}^* \to \{0,1\}^\lambda$, PPT $\mathcal{A}_v$ with auxiliary input z_v, any $\mathcal{A}_s$, any message s and $h \in \mathcal{H}$, Exp_0 is the ℓ-more experiment under leakage, $\mathsf{Exp}^{\mathbf{s},h}_{\mathcal{A}_v,\mathcal{A}_s,\mathcal{E}_{\mathcal{A}_v}}(\ell,\lambda,z_v,z_{\mathcal{E}})$, of Definition 4.3.15. In order to fully define the experiment we need to define $\mathcal{E}$, $z_{\mathcal{E}}$: for any g, $\mathcal{A}_v$, z_v, we define the non-leakage attacker $\mathcal{A}'_v$, such that $\mathcal{A}'_v(h,v,z_v)$ samples

independent randomness $\tau' \leftarrow \{0,1\}^{\text{poly}(k)}$ and executes $(\tilde{v}, st) \leftarrow \mathcal{A}_v(h, v, g(\tau'), z_v)$. By the 1-more extractability of $\mathcal{H}$ (without leakage) there exists extractor $\mathcal{E}^{\mathcal{H}}_{\mathcal{A}'_v}$ with hardcoded auxiliary information $z_{\mathcal{E}}$, for $\mathcal{A}'_v$, and we define $\mathcal{E} := \mathcal{E}^{\mathcal{H}}_{\mathcal{A}'_v}$. We prove that for any $\mathcal{A}_v$, z_v, $\mathcal{A}_s$ and any message s, $\Pr[\mathsf{Exp}_0 = 1] \leq \mathsf{negl}(k)$, otherwise we break 1-more extractability, without leakage, of $\mathcal{H}$, with non-negligible probability.

- In Exp_1 we modify Exp_0 in two ways. First the leakage is computed over the independent randomness τ', instead of τ, which is used to compute v. As we prove, if the adversary is not leaking more than λ bits in total, v is statistically close to a uniform element in the range of the hash, even if the attacker receives leakage over τ, and this modification does not induce any statistical difference between the two experiments. Thus, $\mathcal{A}_v$ cannot distinguish between the two experiments. However, $\mathcal{A}_s$ might do so since for some leakage query g we might have $g(\tau) \neq g(\tau')$. Hence, for any $\mathcal{A}_s$, we define an $\mathcal{A}'_s$, that given τ, s, computes the output of $\mathcal{A}_s$, exactly as it happens in Exp_0, i.e., $\mathcal{A}'_s(h, \tau, s, st)$ computes (i) $v \leftarrow h(s;\tau)$, (ii) $(\tilde{v}, st) \leftarrow \mathcal{A}_v(h, v, g(\tau), z_v)$, (iii) outputs $(\tilde{\tau}, \tilde{s}) \leftarrow \mathcal{A}_s(h, \tau, s, st)$, and clearly, the output of $\mathcal{A}_s$ in Exp_0, matches the output of $\mathcal{A}'_s$ in Exp_1.
- In Exp_2, for any $\mathcal{A}_v$, we substitute $\mathcal{A}_v$ with $\mathcal{A}'_v$ such that $\mathcal{A}'_v(h, v, z_v)$ samples $\tau' \leftarrow \{0,1\}^{\text{poly}(k)}$ and outputs $(\tilde{v}, st) \leftarrow \mathcal{A}_v(h, v, g(\tau'), z_v)$. Exp_2 is the original ℓ-more experiment (without leakage) and it is not hard to see that $\mathsf{Exp}_1 \approx \mathsf{Exp}_2$.

In the following claims we prove statistical indistinguishability between the hybrids. The statistical distance between Exp_0 and Exp_1 is bounded by the distance of the input/output variables to $\mathcal{A}_v$, $\mathcal{A}_s$, $\mathcal{A}'_s$ and $\mathcal{E}$.

Claim 4.3.17. *For any any leakage function g, any s, $\mathcal{A}_v$, $\mathcal{A}_s$, z_v, $(h(s;\tau), g(\tau)) \approx (h(s;\tau), g(\tau'))$, over the randomness of Exp_0, Exp_1.*

Proof. By assumption we have that for any s, $\mathbf{H}_\infty(\mathsf{Enc}(s)) \geq \lambda + k + \omega(\log k)$. Moreover, each leakage query g can leak at most λ bits of τ (as all queries cannot leak more than λ bits, in total). Since the randomness of the encoder is independent of z_v, h, we have that for $Z = (z_v, h, g(h))$, $\mathbf{H}_\infty(\mathsf{Enc}(s) \mid Z) \geq k + \omega(\log k)$. Thus, $\tilde{\mathbf{H}}_\infty(\mathsf{Enc}(s) \mid Z) \geq k + \omega(\log k)$. By the Left-Over Hash Lemma (Lemma 2.1.11) and the universality of the inner product function (Lemma 2.1.13), the distribution $\langle \mathbf{r}, \mathsf{Enc}(s;\tau) \rangle$ is statistically close to uniform over

Exp_0 :
$\tau \leftarrow \{0,1\}^{\mathrm{poly}(k)}, v = h(s;\tau)$

$(\tilde{v}, st) \leftarrow \mathcal{A}_v(h, v, g(\tau), z_v)$
$(\hat{\tau}, \hat{s}) \leftarrow \mathcal{E}(h, v)$
$(\tilde{\tau}, \tilde{s}) \leftarrow \mathcal{A}_s\,(h, \tau, s, st)$

If $h(\tilde{s};\tilde{\tau}) = \tilde{v} \wedge \tilde{v} \neq v \wedge h(\hat{s};\hat{\tau}) \neq \tilde{v}$:
return 1
otherwise, return 0

Exp_1 :
τ, $\tau' \leftarrow \{0,1\}^{\mathrm{poly}(k)}$, $v = h(s;\tau)$

$(\tilde{v}, st) \leftarrow \mathcal{A}_v(h, v,$ $g(\tau')$ $, z_v)$
$(\hat{\tau}, \hat{s}) \leftarrow \mathcal{E}(h, v)$
$(\tilde{\tau}, \tilde{s}) \leftarrow \mathcal{A}'_s\,(h, \tau, s, st)$

If $h(\tilde{s};\tilde{\tau}) = \tilde{v} \wedge \tilde{v} \neq v \wedge h(\hat{s};\hat{\tau}) \neq \tilde{v}$:
return 1
otherwise, return 0

Exp_2 :
$\tau \leftarrow \{0,1\}^{\mathrm{poly}(k)}, v = h(s;\tau)$

$(\tilde{v}, st) \leftarrow \mathcal{A}'_v(h, v, z_v)$
$(\hat{\tau}, \hat{s}) \leftarrow \mathcal{E}(h, v)$
$(\tilde{\tau}, \tilde{s}) \leftarrow \mathcal{A}'_s\,(h, \tau, s, st)$

If $h(\tilde{s};\tilde{\tau}) = \tilde{v} \wedge \tilde{v} \neq v \wedge h(\hat{s};\hat{\tau}) \neq \tilde{v}$:
return 1
otherwise, return 0

Figure 4.1: The hybrid experiments for the proof of Theorem 4.3.16. The gray part signifies the portion of the code of an experiment that differs from the previous one.

$\mathbb{Z}_p$ and we have

$$(h(s;\tau), g(\tau)) = \left(\left(g^{\langle \mathbf{r}, \mathsf{Enc}(s;\tau)\rangle}, g^{a\langle \mathbf{r}, \mathsf{Enc}(s;\tau)\rangle}\right), g(\tau)\right) \approx ((g^r, g^{ar}), g(\tau))\,,$$

for uniform r, τ. Since τ, τ', r, are uniform and independent we have $((g^r, g^{ar}), g(\tau))$ $\approx ((g^r, g^{ar}), g(\tau'))$, and thus $(h(s;\tau), g(\tau)) \approx (h(s;\tau), g(\tau'))$. This concludes the proof of the claim and the input and output distributions for $\mathcal{A}_v$ and $\mathcal{E}$ in both experiments are the same. □

By the above claim and the fact that the input and output distributions of $\mathcal{A}_s$ and $\mathcal{A}'_s$ are the same (by the definition of $\mathcal{A}'_s$), we have that $\mathsf{Exp}_0 \approx \mathsf{Exp}_1$.

Finally, in Exp_2, $\mathcal{A}'_v$ is just sampling τ' internally and then executes $\mathcal{A}_v$. Again the output distributions of $\mathcal{A}_v$ and $\mathcal{A}'_v$ are the same, thus $\mathsf{Exp}_1 \approx \mathsf{Exp}_2$.

From the above we have that $\mathsf{Exp}_0 \approx \mathsf{Exp}_2$ and $|\Pr[\mathsf{Exp}_0 = 1] - \Pr[\mathsf{Exp}_2 = 1]| \leq \mathsf{negl}(k)$. Thus, assuming $\Pr[\mathsf{Exp}_0 = 1] > \epsilon$, for $\epsilon = 1/\mathrm{poly}(k)$, we receive that $\Pr[\mathsf{Exp}_2 = 1] > \epsilon' = \epsilon - \mathsf{negl}(k)$, and the 1-more extractability of $\mathcal{H}$ breaks with non-negligible probability (recall that Exp_2 is the 1-more experiment without leakage). Thus, $\Pr[\mathsf{Exp}_0 = 1] \leq \mathsf{negl}(k)$, and 1-more extractability under leakage for $\mathcal{H}$ follows. □

By plugging the non-malleable encoding scheme, $(\mathsf{Init}, \mathsf{Enc}, \mathsf{Dec})$, against affine functions (cf. Construction 4.3.8), with $|p| = k + \lambda$, as the underlying encoding scheme to Construction 4.3.2, we have that for any message s, $\mathbf{H}_\infty(\mathsf{Enc}(s)) \geq \lambda + k + \omega(\log k)$, and Construction 4.3.2 is an ℓ-more wECRH against λ bits of leakage (cf. Definition 4.3.15).

4.4 ℓ-more weakly extractable hash functions in the random oracle model

In the current section, we prove that any hash function is an ℓ-more wECRH (cf. Definition 4.2.1) when it is modeled as a random oracle. Since in the random oracle model, the randomness comes from the oracle, we do not hash the message using independent randomness, τ, and in addition, the adversary and the extractor receive black box access to the hash function, h. In this setting, leakage resilience (cf. Definition 4.3.15) is straightforward.

In what follows, we briefly discuss the main idea behind the proof of Theorem 4.4.1, presented below. Any adversary, $(\mathcal{A}_v, \mathcal{A}_s)$, against the ℓ-more extractability, is required (i) to produce a new valid hash value $\tilde{v}$ (this value is produced by $\mathcal{A}_v$) and (ii) to produce a valid pre-image for $\tilde{v}$ (this value is produced by $\mathcal{A}_s$). The extractor, who is given access only to the queries made by $\mathcal{A}_v$, checks if there is any query (pre-image) that hashes to $\tilde{v}$, and if so, it correctly outputs that pre-image, otherwise it outputs $\perp$. In the latter case, the extractor fails only if $\mathcal{A}_s$ manages to output a valid pre-image for $\tilde{v}$, which happens with negligible probability, as for any $\tilde{s}$ output by $\mathcal{A}_s$, when querying the oracle with $\tilde{s}$, the probability of receiving $\tilde{v}$ as a reply, is negligible. This idea is formally presented in the following theorem.

Theorem 4.4.1. *Let $k, \ell \in \mathbb{N}$ and let h be a function, $h : \{0,1\}^* \to \{0,1\}^k$. Assuming h is modeled as a random oracle, then h is an ℓ-more* wECRH *(cf. Definition 4.2.1).*

Proof. Let h be a random function that will be accessed by the extractor and the attacker in a black-box way. For any $\mathcal{A}_v$ with auxiliary input z_v we define $z_{\mathcal{E}} := z_v$ and $\mathcal{E}^{h(\cdot)}$ as follows:

$\mathcal{E}^{h(\cdot)}(\mathbf{v}, z_{\mathcal{E}})$:

1. (**Initialization**): Set $Q_v := \emptyset$.
2. (**Execute** $\mathcal{A}_v$): Execute $\mathcal{A}_v(\mathbf{v}, z_v)$ and for each oracle query q of $\mathcal{A}_v$, query the oracle with q, set $Q_v = Q_v \cup \{\mathsf{q}\}$ and send $h(\mathsf{q})$ to $\mathcal{A}_v$. At the end of the execution receive $\tilde{v}$ from $\mathcal{A}_v$.
3. (**Output**): If there exists $\mathsf{q} \in Q_v$, such that $h(\mathsf{q}) = \tilde{v}$, set $\hat{s} := \mathsf{q}$, otherwise, set $\hat{s} := \perp$. Output $\hat{s}$.

Clearly, the running time of $\mathcal{E}^{h(\cdot)}$ is linear in the running time of $\mathcal{A}_v$. According to Definition 4.2.1, we need to prove that for any any PPT algorithm $\mathcal{A}_v$, any $z_v \in \{0,1\}^{\mathrm{poly}(k)}$, all PPT algorithms $\mathcal{A}_s$, any large $k \in \mathbb{N}$ and any vector of messages $\mathbf{s} = (s_1, \ldots, s_\ell)$,

$$\Pr_h\left[\mathsf{Exp}^{\mathbf{s},h}_{\mathcal{A}_v,\mathcal{A}_s,\mathcal{E}}(\ell, z_v, z_{\mathcal{E}}) = 1\right] \leq \mathsf{negl}(k),$$

where,

$$\begin{array}{l}
\mathsf{Exp}^{\mathbf{s},h}_{\mathcal{A}_v,\mathcal{A}_s,\mathcal{E}}(\ell, z_v, z_{\mathcal{E}}) : \\
v_i = h(s_i), i \in [\ell] \\
\mathbf{v} = (v_1, \ldots, v_\ell) \\
(\tilde{v}, st) \leftarrow \mathcal{A}_v^{h(\cdot)}(\mathbf{v}, z_v) \\
\hat{s} \leftarrow \mathcal{E}^{h(\cdot)}(\mathbf{v}, z_{\mathcal{E}}) \\
\tilde{s} \leftarrow \mathcal{A}_s^{h(\cdot)}(\mathbf{s}, st) \\
\\
\text{If } h(\tilde{s}) = \tilde{v} \wedge \forall i : \tilde{v} \neq v_i \wedge h(\hat{s}) \neq \tilde{v}, \text{ return } 1 \\
\text{otherwise, return } 0
\end{array}$$

Let $\mathcal{Q} = \{s_i \mid i \in [\ell]\}$. We define the following events,

$$B\text{: } \mathsf{Exp}^{\mathbf{s},h}_{\mathcal{A}_v,\mathcal{A}_s,\mathcal{E}}(\ell, z_v, z_{\mathcal{E}}) = 1,\ E\text{: } h(\tilde{s}) = \tilde{v} \wedge \forall i : \tilde{v} \neq v_i.$$

Clearly, $\Pr[B \wedge \neg E] = 0$, thus we only need to bound $\Pr[B \wedge E]$.

$$\begin{aligned}
\Pr[B \wedge E] &= \Pr[h(\tilde{s}) = \tilde{v} \wedge \forall i : \tilde{v} \neq v_i \wedge h(\hat{s}) \neq \tilde{v}] \\
&= \Pr[h(\tilde{s}) = \tilde{v} \wedge \forall i : \tilde{s} \neq s_i \wedge \hat{s} = \perp] \\
&\leq \Pr[h(\tilde{s}) = \tilde{v} \wedge \tilde{s} \notin (\mathcal{Q} \cup Q_v)] = \frac{1}{2^k} \leq \mathsf{negl}(k),
\end{aligned}$$

where the last inequality follows from the fact that $\tilde{s}$ does not belong to the set of queries made to h, thus $h(\tilde{s})$ is completely random over $\{0,1\}^k$. This completes the proof of the theorem. $\square$

Chapter 5

Non-malleable codes in the split-state model

5.1 Introduction

The *split-state* model was originally introduced and studied in the context of non-malleable codes, by Dziembowski et al. [DPW10] and Liu and Lysyanskaya [LL12], that considered the case of *two split states*. Briefly speaking, in the split-state model with two states, private memory is split into two parts, c_0, c_1, and the attacker may apply any function $f = (f_0, f_1)$ that results in a tampered memory equal to $(f_0(c_0), f_1(c_1))$. The critical point here is that each f_i, for $i \in \{0, 1\}$, tampers with c_i independently of c_{1-i}. This is a plausible model to assume since there are many scenarios in which sensitive data may be split into two storage devices that are physically separated for security reasons. In this setting, an adversary that receives tampering access over the two devices modifies the contents of each memory independently of the contents of the other. Note that, the model can generalize to multiple split states, with the two-state variant being the hardest to achieve. In the present chapter, we consider the problem of constructing practically efficient non-malleable codes against split-state attackers with two states, and any reference to the split-state model is with respect to the two-state variant. Before presenting the contributions of the present chapter, the state of the art of split-state non-malleable codes is reviewed.

Broadly speaking, explicit constructions of non-malleable codes in the split-state model can be categorized into information-theoretic and computational.[1] In a recent break-

[1]The work of [DPW10] showed that in the random oracle model, there exist efficient non-malleable codes against split-state tampering functions. However, their approach uses a probabilistic argument thus providing only a proof of existence and not an explicit construction.

through result, Aggarwal et al. [ADL14], provide the first polynomial-time, information-theoretic, non-malleable code for multi-bit messages, significantly improving over the work by Dziembowski et al. [DKO13], which only supports single-bit messages. The encoder of [ADL14] produces codewords of length $O((|s|+k)^7)$, where $|s|$ denotes the length of the encoded message, s, and k is the security parameter. Later, Aggarwal et al. [ADKO15] proposed another construction that achieves codeword length roughly $O(|s|)$, for sufficiently large $|s|$, however as the authors point in the conclusion of their work, the hidden constants may be "astronomical", as they depend on results from additive combinatorics.

The first computationally secure non-malleable code is presented by the work of Liu and Lysyanskaya [LL12], that construct an efficient non-malleable code for split-state attackers, using cryptographic tools such as leakage resilient public-key encryption [NS09], and robust non-interactive zero-knowledge (NIZK) proofs [DDO$^+$01]. The rate of their construction is not provided in the original paper and a textbook instantiation with public-key encryption combined with NIZKs, does not yield a rate 1 code, since the length of the NIZK proof is proportional to the message length. However, using state of the art tools, a better instantiation of [LL12] could be provided, by combining the results of [LL12] together with the compiler of [AAG$^+$16], the public-key leakage resilient encryption of [NS09] and the efficient NIZKs of [GS08], yielding a scheme with codeword length $|s|+O(k^2)$ (cf. Table 5.1). In another work, Aggarwal et al. [AAG$^+$16] presented a compiler that transforms any low rate, non-malleable code, to a rate 1, computationally secure, non-malleable code. The underlying encoding must satisfy a notion, strictly stronger than non-malleability, called *augmented non-malleability*, which, as it is stated in [AAG$^+$16], can be satisfied by the construction of [ADL14]. By instantiating the compiler of [AAG$^+$16] with the construction of [ADL14], the codeword's length becomes $|s|+O(k^7)$.

Although the above constructions achieve rate 1 asymptotically, i.e., the ratio of message to codeword length is 1, as the message length, $|s|$, goes to infinity, in practice, the induced overhead can still be too large when considering short messages, e.g., a 160-bit cryptographic key, even without counting the potentially *large hidden constants* in the asymptotic notation. Thus, even though the problem of "optimal-rate" has been solved in theory, it is still unclear what the practical implications of those constructions are. Given the current state of the art, as discussed above, constructing codes with very small overhead, including the hidden constant, remains still one of the most important open questions in the area. Note, that the natural lower bound for code length is merely $|s|+k$, and none of the existing, computational or information-theoretic, constructions, match it,

even asymptotically.

5.1.1 Contributions

In the present chapter we tackle the problem of constructing truly efficient non-malleable codes in the split-state model, based on the notion of *ℓ-more weakly extractable hash function families* (wECRH), that was presented in Chapter 4. Recall that, ℓ-more extractable hash function families capture the idea that, if an adversary, that given $\ell \in \mathbb{N}$ precomputed hash values $v_1, \ldots, v_\ell$, manages to produce a new valid hash value $\tilde{v}$, then it must know a pre-image of $\tilde{v}$.

The main result, which is informally presented in the next theorem, states that non-malleable codes against split-state attackers are implied from ℓ-more weakly extractable hash functions. The crux of the underlying methodology is to adapt the "public-key-encrypt-and-prove" method of [LL12], using the notion of ℓ-more weakly extractable hash function families, yielding effectively a "(one-time-symmetric-key-encrypt)-and-hash" approach for obtaining non-malleable codes. In particular, the following, informally stated theorem is proved:

Theorem 5.1.1 (Informal)**.** *Assuming* 1*-more* wECRH*,* DLOG*, and one-time leakage-resilient authenticated encryption, there exists an explicit, information rate 1, non-malleable code in the split-state model.*

Assuming the KEA-based 1-more wECRH of Section 4.3.1, the proposed scheme produces codewords of length $|s| + 9k + 2\log^2(k)$ (or $|s| + 18k$ depending on the instantiation, while for the random oracle based one, the codeword length is $|s| + 6k + 2\log^2(k)$ (cf. Section 5.3). In Table 5.1, a comparison of the new scheme with the current state of the art on the split-state setting, is provided. The new construction is truly efficient in terms of codeword length, and it is one order of magnitude better than the combination of [LL12] + [AAG$^+$16] + [NS09] + [GS08], which is the most competitive scheme that can be constructed,[2] based on the current state of the art. It should be noted that, existing constructions in the information-theoretic setting, such as [ADL14, ADKO15], and the works built on top of them, e.g., [AAG$^+$16], might require very large constants, inherited by the results in additive combinatorics (cf. the conclusion of [ADKO15]).

[2]For the sake of this comparison, [LL12] is instantiated with the efficient zero-knowledge proofs of [GS08] and the leakage resilient public-key encryption of [NS09]; moreover it can be observed that the resulting encoding scheme is compatible with the compiler of [AAG$^+$16] (it satisfies "augmented non-malleability",

Scheme	Codeword length	Model	Assumption
[ADL14]	$O\left((\lvert s\rvert + k)^7 \log^7(\lvert s\rvert + k)\right)$	IT	N/A
[ADKO15][3]	$O(\max\{\lvert s\rvert, k\})$	IT	N/A
[ADL14] + [AAG+16]	$\lvert s\rvert + O\left(k^7\right)$	Comp.	AE
[LL12] + [AAG+16] + [NS09] + [GS08]	$\lvert s\rvert + O\left(k^2\right)$	Comp., CRS[4]	LR-PKE + robust NIZK
This thesis	$\lvert s\rvert + 9k + 2\log^2(k)$	Comp., CRS	1-time LR-AE + KEA
This thesis	$\lvert s\rvert + 6k + 2\log^2(k)$	Comp., RO	1-time LR-AE

Table 5.1: Comparison of multi-bit NMCs in the split-state model. k is the security parameter, "IT" stands for information-theoretic security, "Comp." for computational security, "AE" for authenticated encryption, and "LR" for leakage-resilient, respectively. In the information-theoretic setting, typically security breaks with probability $\epsilon = 2^{-\Omega(k)}$; in the computational setting, we have $\epsilon = \mathsf{negl}(k)$, e.g., $\epsilon = k^{-\omega(1)}$ or $2^{-\Omega(k)}$, depending on how strong the underlying computational assumption is.

5.1.2 Technical overview

The proposed NMC construction against split-state attackers is inspired by the one of Liu and Lysyanskaya [LL12], so we first revisit their construction. To encode a message s, the encoder of [LL12] outputs $(sk, (pk, \mathsf{E}_{pk}(s), \pi))$, where E is the encryption algorithm of a leakage-resilient, semantically secure, public-key encryption scheme $(\mathsf{KGen}, \mathsf{E}, \mathsf{D})$, sk, pk, denote the secret key and public key, respectively, and π is a non-interactive proof of knowledge (robust NIZK), that proves the existence of a valid secret key, decrypting the ciphertext to the message s.

The construction proposed in the present thesis significantly improves the efficiency of [LL12] by refining their approach in two ways: (1) by replacing the leakage-resilient public-key encryption scheme, with a one-time, symmetric-key, leakage-resilient authenticated encryption scheme, and (2) by replacing the (robust) NIZK proof with a 1-more wECRH, which was introduced in Chapter 4. The encoder works as follows: to encode a message s, the encoder outputs $\Big((\tau, sk), (e = \mathsf{E}_{sk}(s), v = h(sk; \tau))\Big)$, where E is the encryption algorithm of a symmetric, leakage-resilient, authenticated encryption scheme, sk is the corresponding secret key, h is a randomized 1-more wECRH, and τ denotes its randomness.

Here the reader can observe that, using a function h that is extractable according to [BCCT12], i.e., not 1-more extractable, is not a good idea. Since generic authenticated encryption schemes guarantee security only if the secret key remains the same, it is possible

a property defined in the latter paper) and thus the resulting system is of rate 1. This provides codeword length $\lvert s\rvert + O(k^2)$, cited in Table 5.1.

[4]The size of the CRS is $O(k)$, see [GS08]. The size of the CRS in the constructions presented in the current chapter, is roughly $32k$ bits, cf. Section 5.2, and it is independent of $\lvert s\rvert$.

to break security if the adversary modifies sk as well. In fact, it is possible to construct an authenticated encryption scheme such that it becomes insecure if the secret key is modified. Therefore, if the hash function family is malleable, then the tampering function may compute $(\tilde{e} = \mathsf{E}_{\tilde{sk}}(s+1), \tilde{v} = h(\tilde{sk}))$, where $\tilde{sk}$ is a bad key that does not provide security. In this setting, the tampered codeword decodes to a related message, and non-malleability is impossible to prove. The 1-more extractability property resolves this issue: even if the attacker is given access to a valid hash value v, it cannot produce a valid hash value $\tilde{v}$, unless it knows a valid pre-image. Proving security for the above construction requires to handle multiple subtleties, and the reader is referred to Section 5.2 for further details.

The proposed one-time, symmetric, leakage-resilient authenticated encryption scheme, in order to sustain $2k + \log^2 k$ bits of leakage, it requires key and ciphertext length $|s| + 5k + 2\log^2(k)$ (cf. Section 5.3). In addition, the KEA based 1-more wECRH satisfies $|\tau| = |v| = 2k$ (see Constructions 4.3.8 and 4.3.2). Therefore, the total codeword length is $|s| + 9k + 2\log^2(k)$ (or $|s| + 18k$, cf. Section 5.3). The encoding and decoding procedures require 128 group operations (64 exponentiations plus 64 multiplications), independently of the message length, plus the cost of one-time authenticated encryption and decryption, respectively. The random oracle-based construction produces codewords of length $|s|+6k+2\log^2(k)$, while the encoding and decoding procedures require the computation of a hash function, plus the cost of one-time authenticated encryption and decryption, respectively.

5.1.3 Related work on split-state NMC

The first explicit non-malleable code in the split-state model, for the information-theoretic setting was proposed by [DKO13], yet their scheme can only encode single-bit messages. Subsequent constructions for multi-bit messages are discussed in the previous section. Non-malleable codes for other function classes have been extensively studied, e.g., bit-wise independent tampering [DPW10], bounded-size function classes [FMVW14], the k-split setting [CZ14], block-wise tampering [CKM11, CGM+15], and bounded depth and fan-in circuits [BDKM16]. The work of [ADKO15] develops beautiful connections among different function classes.

Other aspects of non-malleable codes have also been studied, such as rate-function class tradeoff, in the information-theoretic setting [CG14]. Other variants of non-malleable codes have been proposed, such as continuous non-malleable codes [FMNV14, JW15], augmented non-malleable codes [AAG+16], locally decodable/updatable non-malleable

codes [DLSZ15, DKS17b, FMNV15, CKR16], which were used to secure the implementation of RAM computation, and non-malleable codes with split-state refresh [FN17]. Leakage resilience was also considered as an additional feature, e.g., [LL12, DLSZ15, CKR16, FN17].

A related line of work in tamper resilience aims to protect circuit computation against tampering attacks on circuit wires [IPSW06, FPV11, DK12, DK14] or gates [KT13]. In this setting, using non-malleable codes for protecting the circuit's private memory is an option, still in order to achieve security the encoding and decoding procedures should be protected against fault injection attacks using the techniques from [IPSW06, FPV11, DK12, DK14, KT13].

5.2 A non-malleable code against split-state tampering

In this section, we present our construction of non-malleable codes against split-state tampering functions. Our construction requires (i) a one-time, authenticated, symmetric-key encryption scheme that is also leakage resilient, and (ii) a 1-more wECRH.

Construction 5.2.1. *Let $\mathcal{H}_k = (\mathsf{Gen}, h)$ be a hash function family, and let $(\mathsf{KGen}, \mathsf{E}, \mathsf{D})$ be a symmetric encryption scheme. We define a coding scheme $(\mathsf{Init}, \mathsf{Enc}, \mathsf{Dec})$, as follows:*

- $\mathsf{Init}(1^k)$*: sample $z \leftarrow \mathsf{Gen}(1^k)$ and set $\Sigma := z$.*
- $\mathsf{Enc}(\Sigma, \cdot)$*: let s be the input to the encoder. The encoder samples $sk \leftarrow \mathsf{KGen}(1^k)$, $\tau \leftarrow \{0,1\}^{\mathrm{poly}(k)}$, $e \leftarrow \mathsf{E}_{sk}(s)$, and outputs $\Big(\tau, sk, e, h_z(sk;\tau)\Big)$. In particular, the left part of the codeword is (τ, sk), while the right part is $(e, h_z(sk;\tau))$.*
- $\mathsf{Dec}(\Sigma, \cdot)$*: let (τ, sk, e, v) be the input to* Dec*. If $h_z(sk;\tau) = v$, the decoder outputs $\mathsf{D}_{sk}(e)$, otherwise, it outputs $\perp$.*

Since the input message to h_z, sk, possesses sufficient entropy, it is possible to omit τ in the above construction, still for the sake of clarity we stick to the formulation provided in Definition 4.2.1 and we use independent randomness for hashing sk. Also, for brevity we will use h to refer both to the program of the hash function and the key, i.e., z is omitted.

In what follows we prove that Construction 5.2.1 is strongly non-malleable (cf. Definition 2.3.3) against the class of split-state functions $\mathcal{F}_{\mathsf{ss}}$ (cf. Definition 2.2.3), assuming

that for any $f = (f_0, f_1) \in \mathcal{F}_{\mathsf{ss}}$, f_0, f_1, tamper with (τ, sk) and (e, v), respectively, i.e., we assume the strings $r||sk$, $e||v$, are of length $\nu/2$, where ν is the length the codeword.[5]

Before formally analyzing the security of the construction presented above, we first discuss the ideas on why our construction is secure. Consider a split-state tampering function (f_0, f_1), where f_0 is applied to (τ, sk) and f_1 is applied to (e, v). To prove non-malleability, one of the primitives that we rely on, is leakage-resilient semantically secure encryption. In our proof, the simulator is provided with the ciphertext, e, and the hash, v, where the latter is treated as leakage over the secret key sk, and clearly, the simulator can compute $(\tilde{e}, \tilde{v}) \leftarrow f_1(e, v)$. However, in case $\tilde{v} \neq v$, the simulator needs to be able to produce the decoding of the codeword, and this is where 1-more weak extractability will be used as the tool for obtaining a valid preimage, $(\hat{\tau}, \hat{sk})$, for $\tilde{v}$. It might be very tempting to conclude the simulation by outputting the decrypted message $\mathsf{D}_{\hat{sk}}(\tilde{e})$ (where $\tilde{e}$ is the modified ciphertext). However, this may not be consistent with the real-world experiment, as the values produced by the extractor $(\hat{\tau}, \hat{sk})$ might not be consistent with the output of f_0. To check consistency, the simulator would want to check the output of f_0, yet such a simulation would be impossible to prove since it depends on sk, and the semantic security of the encryption does not hold in the presence of it. To go around this, we use a similar technique to Liu and Lysyanskaya [LL12], who observed that, the equality test between $f_0(\hat{\tau}, \hat{sk})$ and $f_0(\tau, sk)$ can be performed via leakage of over sk, i.e., by leaking the output of a universal hash function (cf. Definition 2.1.10) with $\log^2 k$ bits of output, over sk. Putting this to our setting, by requiring the encryption scheme $(\mathsf{KGen}, \mathsf{E}, \mathsf{D})$ to be one-time semantically secure, symmetric-key authenticated encryption, that is secure under $2k + \log^2 k$ bits of leakage, is sufficient to facilitate the simulation. We also note, that the case where $\tilde{v}$ is not modified by f_1, can be easily taken care of by the security of the authenticated encryption and the collision resistance property of h: if $v = \tilde{v}$, then with overwhelming probability $sk = \tilde{sk}$, and any attempt to modify the ciphertext results in an invalid ciphertext, with overwhelming probability.

Below we prove strong non-malleability for Construction 5.2.1.

Theorem 5.2.2. *Let k be the security parameter, $\mathcal{H}_k$ be a 1-more* wECRH *that outputs $\beta(k)$ bits, $\beta(k) = \mathrm{poly}(k)$, and let $(\mathsf{KGen}, \mathsf{E}, \mathsf{D})$ be an authenticated, semantically secure, symmetric encryption scheme, that is leakage resilient against $\lambda(k) := \omega(\log k) + \beta(k)$, bits of leakage. Then, Construction 5.2.1 is strongly non-malleable against $\mathcal{F}_{\mathsf{ss}}$.*

[5] This can always be achieved using padding.

Proof. Following the definition of strong non-malleability (Definition 2.3.3), we need to prove that for any $f = (f_0, f_1) \in \mathcal{F}_{\mathsf{ss}}$ and any pair of messages s_0, s_1, $(\Sigma, \mathsf{Tamper}_{s_0}^{f,\Sigma}) \approx_c (\Sigma, \mathsf{Tamper}_{s_1}^{f,\Sigma})$, where $\Sigma \leftarrow \mathsf{Init}(1^k)$. We introduce a series of hybrid experiments (see Figure 5.1), and non-malleability can be derived directly from the indistinguishability between adjacent hybrids. We first explain the hybrids and define the notation used in those experiments.

- Given a tampering function $f = (f_0, f_1)$ and message s, the first experiment, $\mathsf{Exp}_0^{f,\Sigma,s}$, is exactly the original tampering game, $\mathsf{Tamper}_s^{f,\Sigma}$, of Definition 2.3.3, and $\Sigma \leftarrow \mathsf{Init}(1^k)$.

- In $\mathsf{Exp}_1^{f,\Sigma,s}$, we slightly modify the previous hybrid by checking whether the function f_1 has modified the hash value v. Intuitively, by the collision resistance property of the hash function family $\mathcal{H}_k$, if f_1 does not modify v, then the attack produces a valid codeword, $\tilde{c}$, only if the parts of $\tilde{c}$ that constitute the preimage of $\tilde{v}$, are kept intact, i.e., $(\tau, sk) = (\tilde{\tau}, \tilde{sk})$, otherwise there is a collision. In addition, assuming $sk = \tilde{sk}$, we have that, if $\tilde{e} \neq e$, then the output of the decoder should be $\perp$, otherwise we break the authenticity under leakage (v is considered as leakage over sk) property of the encryption scheme. On the other hand, if $v \neq \tilde{v}$, the output of the current experiment is produced as in $\mathsf{Exp}_0^{f,\Sigma,s}$.

- In $\mathsf{Exp}_2^{f,\Sigma,s}$, we modify the previous experiment for the case in which v is modified. For this case, instead of using the real decoding procedure, we use the extractor of the hash function family, to extract a preimage $(\hat{\tau}, \hat{sk})$, for $\tilde{v}$, and then we compute the output, $\tilde{s}$, with respect to that preimage. However, we cannot output $\tilde{s}$ directly as we still need to check consistency with the output of f, i.e., we need to check whether $(\hat{\tau}, \hat{sk})$ is equal to $(\tilde{\tau}, \tilde{sk})$. The indistinguishability between the current hybrid and the previous one, follows by the 1-more weak extractability property of the hash function, which, informally, guarantees that if $\tilde{c}$ is a valid codeword, then there exists an extractor $\mathcal{E}_{\mathcal{A}_v}^{\mathcal{H}}$ (denoted as $\mathcal{E}$ in Figure 5.1) that produces a valid preimage for $\tilde{v}$, with overwhelming probability. If the extracted preimage is consistent with the one output by f, the current hybrid outputs a non-bottom value, equal to the one output by the decoding procedure of $\mathsf{Exp}_1^{f,\Sigma,s}$. On the other hand, if $(\hat{\tau}, \hat{sk}) \neq (\tilde{\tau}, \tilde{sk})$, the collision resistance property of $\mathcal{H}_k$ (cf. Lemma 4.2.2) guarantees that $(\tilde{\tau}, \tilde{sk})$ is not a valid preimage for $\tilde{v}$, with overwhelming probability, and the current experiment

properly outputs $\perp$. Finally, it is straightforward to see, that if $\tilde{v}$ is invalid, both experiments output $\perp$.

In order to define the extractor $\mathcal{E}^{\mathcal{H}}_{\mathcal{A}_v}$ (or $\mathcal{E}$ for brevity), introduced in $\mathsf{Exp}_2^{f,\Sigma,s}$, we first need to define $\mathcal{A}_v$, z_v, with respect to h, v, e, and $f = (f_0, f_1)$. Formally, we define the following:

1. (**Define $\mathcal{A}_v$**): $\mathcal{A}_v(h, v, z_v) := ([f_1(z_v, v)]_2, st)$, where
$$st := (f_1(z_v, v), z_v, v).$$
2. (**Choose auxiliary info for $\mathcal{A}_v$**): define $z_v := e$.
3. (**Existence of the extractor, $\mathcal{E}^{\mathcal{H}}_{\mathcal{A}_v}$, and auxiliary input, z_v**): Given $\mathcal{A}_v$ and z_v, by the 1-more weak extractability property of $\mathcal{H}_k$, there exists an extractor $\mathcal{E}^{\mathcal{H}}_{\mathcal{A}_v}$, with hardwired auxiliary input, $z_{\mathcal{E}}$, that computes $(\hat{\tau}, \hat{sk}) \leftarrow \mathcal{E}^{\mathcal{H}}_{\mathcal{A}_v}(h, v)$. The extractor $\mathcal{E}^{\mathcal{H}}_{\mathcal{A}_v}$ is used in $\mathsf{Exp}_2^{f,\Sigma,s}$ and all subsequent experiments (for brevity we denote it as $\mathcal{E}$).

We remind, that, for any vector x, $[x]_i$, denotes the i-th coordinate of x.

- In $\mathsf{Exp}_3^{f,\Sigma,s}$, we modify the consistency check procedure, so that we access the right part of the codeword, only through leakage. Instead of checking consistency using directly the output of f_0, we do the check using a uniformly random hash function, $\bar{h}$, from a universal family (cf. Definition 2.1.10), applied to the output of f_0; we also leak an additional bit that indicates whether f_0 has modified its input. Here, the hash v is computed through leakage over the secret key, sk. The experiment differs from the previous one only when there is a collision against $\bar{h}$, which happens with negligible probability, as $\bar{h}$ is a universal hash function.

  In what follows, we formalize the procedure described above. Let $\bar{h} \leftarrow \bar{\mathcal{H}}_{\lambda-1}$ be a uniformly random element from a universal hash function family, that outputs $\lambda - 1$ bits. We define the function $g_{\bar{h},h}(\cdot)$ as follows:
$$g_{\bar{h},h}(x, y) = \begin{cases} (0, \bar{h}(f_0(x, y)), h(x, y)), & \text{if } f_0(x, y) = (x, y), \\ (1, \bar{h}(f_0(x, y)), h(x, y)), & \text{if } f_0(x, y) \neq (x, y). \end{cases}$$
  We view $g_{\bar{h},h_z}$ as a leakage function that outputs $\lambda = \omega(\log k) + \beta(k)$ bits in total. The experiment will then use the leaked value to check consistency, instead of using the whole string output by f_0. Concretely, we introduce the random variable b,

which depends on the output of the leakage function, and we modify $\mathsf{Exp}_2^{f,\Sigma,s}$, so that the condition "If $(b = 1)$", introduced in $\mathsf{Exp}_3^{f,\Sigma,s}$, is exactly the same as the condition "If $(\tau, sk, e) = (\tilde{\tau}, \tilde{sk}, \tilde{e})$", in experiment $\mathsf{Exp}_2^{f,\Sigma,s}$. This modification does not induce any statistical difference. In the next modification, we check equality between $(\hat{\tau}, \hat{sk})$, $(\tilde{\tau}, \tilde{sk})$, by checking if $\bar{h}(\hat{\tau}, \hat{sk}) = \bar{h}(f_0(r, sk))$. Clearly, this part induces a statistical difference only if there is a collision against $\bar{h}$, which happens with negligible probability, since $\bar{h}$ is a universal hash function, chosen by the current experiment, independently.

- Finally, we are going to show that $\mathsf{Exp}_3^{f,\Sigma,s}$ is indistinguishable from $\mathsf{Exp}_3^{f,\Sigma,0}$, for any message s, where 0 denotes the zero-message. This follows by the semantic security of the leakage resilient encryption scheme (Definition 2.1.6).

A concrete presentation of the hybrids, is given in Figure 5.1.

In the following claims we prove indistinguishability between the hybrids.

Claim 5.2.3. *Assuming $\mathcal{H}_k$ is collision resistant and $(\mathsf{KGen}, \mathsf{E}, \mathsf{D})$ is an authenticated leakage-resilient scheme against $\beta(k)$ bits of leakage, for any $f = (f_0, f_1) \in \mathcal{F}_{\mathsf{ss}}$ and any message s, $\mathsf{Exp}_0^{f,\Sigma,s} \approx_c \mathsf{Exp}_1^{f,\Sigma,s}$, over the randomness of Init, Enc, where Σ follows $\mathsf{Init}(1^k)$.*

Proof. We observe, that the only difference between the two experiments, is that $\mathsf{Exp}_1^{f,\Sigma,s}$ introduces the following branches of conditions: (1) $(v = \tilde{v}) \wedge (\tau, sk, e) = (\tilde{\tau}, \tilde{sk}, \tilde{e})$, (2) $(v = \tilde{v}) \wedge (\tau, sk, e) \neq (\tilde{\tau}, \tilde{sk}, \tilde{e})$, and (3) $v \neq \tilde{v}$. It follows directly that for the conditions (1) and (3), the two experiments are identical. Denote as B the event in which (2) happens and the output of $\mathsf{Exp}_0^{f,\Sigma,s}$ is not $\perp$. From the above we have that $\mathsf{Exp}_0^{f,\Sigma,s} = \mathsf{Exp}_1^{f,\Sigma,s}$ conditioned on $\neg B$. By a standard analysis, we know that the statistical distance between the two experiments is bounded by $\Pr[B]$.

Let E be the event in which $(\tau, sk) = (\tilde{\tau}, \tilde{sk})$. Then we have $\Pr[B] = \Pr[B \wedge E] + \Pr[B \wedge \neg E]$. We will prove that $\Pr[B \wedge E], \Pr[B \wedge \neg E] \leq \mathsf{negl}(k)$. Towards contradiction, suppose there exist function $f \in \mathcal{F}_{\mathsf{ss}}$ and message s, such that $\Pr[B \wedge \neg E] > \epsilon$, for $\epsilon = 1/\mathrm{poly}(k)$. Then, there exists a PPT adversary, $\mathcal{A}$, that breaks the collision resistance property of $\mathcal{H}_k$: the adversary $\mathcal{A}$ simulates the experiment $\mathsf{Exp}_1^{f,\Sigma,s}$ and outputs $(\tau, sk), (\tilde{\tau}, \tilde{sk})$. The function f is computable in polynomial time, so the adversary is also polynomial-time. The adversary wins if the event $B \wedge \neg E$ happens, where by assumption we have $\Pr[B \wedge \neg E] > \epsilon$. Hence, the attacker breaks collision resistance with non-negligible probability. Similarly,

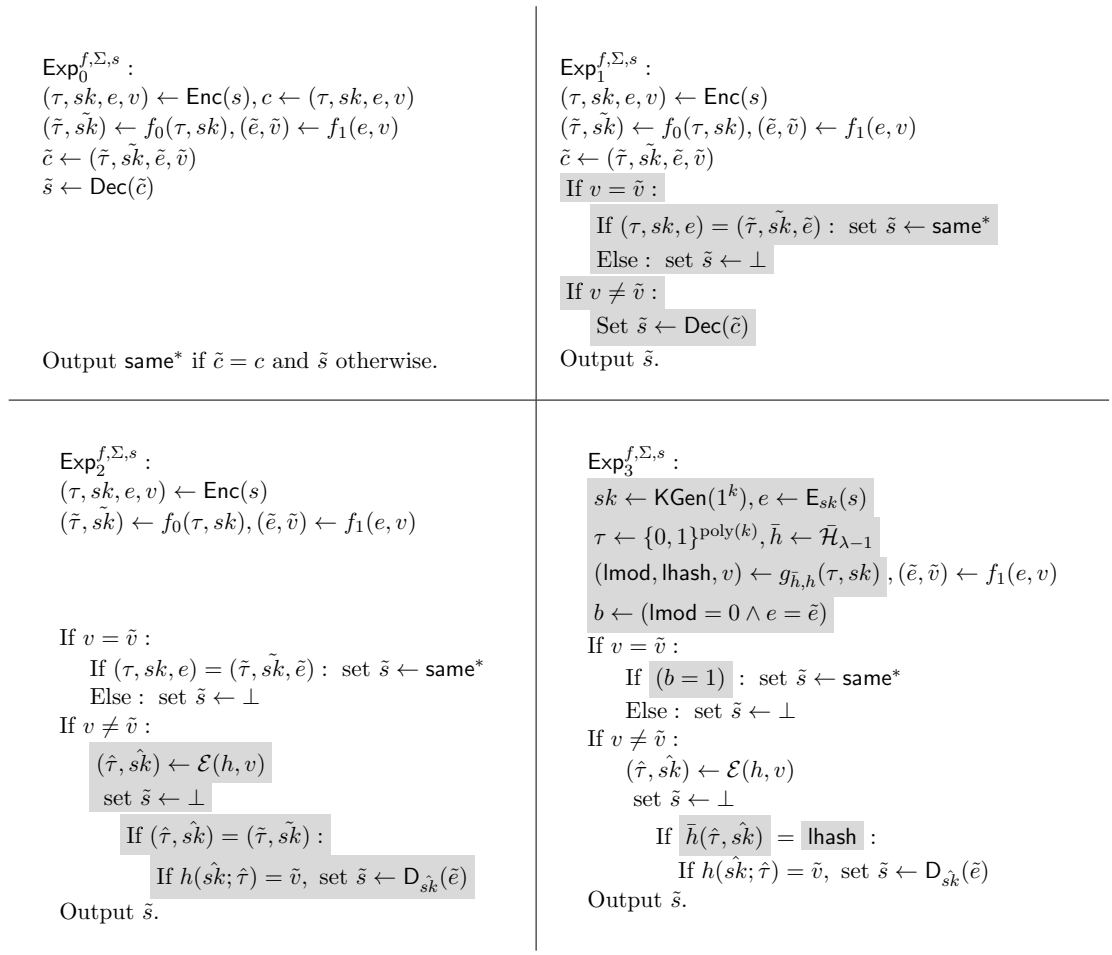

Figure 5.1: Hybrid experiments for the proof of Theorem 5.2.2. Their programs are based on the encoding scheme, $(\mathsf{Init}, \mathsf{Enc}, \mathsf{Dec})$, the encryption scheme, $(\mathsf{KGen}, \mathsf{E}, \mathsf{D})$, and the extractor that is specified in the proof, $\mathcal{E}$. The gray part signifies the portion of the code of an experiment that differs from the previous one.

assuming there exist function $f \in \mathcal{F}_{\mathsf{ss}}$ and message s, such that $\Pr[B \wedge E] > \epsilon$, for some non-negligible ϵ, we have an attacker against the authenticity, under leakage, property of the encryption scheme: the attacker samples $h \leftarrow \mathcal{H}_k$, $\tau \leftarrow \{0,1\}^{\mathrm{poly}(k)}$, and issues a leakage query $g_h(x) := h(x; \tau)$, against the secret key of the encryption scheme. Then, it receives $v = h(sk; \tau)$ and $e \leftarrow \mathsf{E}_{sk}(s)$, executes $(\tilde{e}, \tilde{v}) \leftarrow f_1(e, v)$, and outputs $\tilde{e}$. Assuming $\Pr[B \wedge E] > \epsilon$, we have that $\tilde{e} \neq e$, and $\tilde{e}$ is a valid ciphertext with respect to the secret key sk. Thus, the authenticity under leakage property of the encryption scheme breaks with non-negligible probability ϵ. □

Claim 5.2.4. *Assuming $\mathcal{H}_k$ is a 1-more* wECRH*, for any $f = (f_0, f_1) \in \mathcal{F}_{\mathsf{ss}}$ and any message s, we have that $\mathsf{Exp}_1^{f,\Sigma,s} \approx_c \mathsf{Exp}_2^{f,\Sigma,s}$, over the randomness of* Init, Enc, $\mathcal{E}$, *where*

Σ *follows* $\mathsf{Init}(1^k)$.

Proof. $\mathsf{Exp}_2^{f,\Sigma,s}$ differs from $\mathsf{Exp}_1^{f,\Sigma,s}$, in the following way: instead of using the real decoding procedure, it produces output using the extractor of the 1-more weakly extractable hash function family, $\mathcal{H}_k$. Below we show that the two experiments are computationally indistinguishable.

We first notice that if $(\hat{\tau}, \hat{sk}) = (\tilde{\tau}, \tilde{sk})$, i.e., if the extracted value matches the corresponding value output by f, then, the two experiments are identical. So, it remains to analyze the case where the values are not the same, i.e., the case in which $(\hat{\tau}, \hat{sk}) \neq (\tilde{\tau}, \tilde{sk})$. We denote such an event with E. Then, we partition E into three cases: (1) $E \wedge \big(h(\tilde{sk}; \tilde{\tau}) \neq \tilde{v}\big)$; (2) $E \wedge \big(h(\tilde{sk}; \tilde{\tau}) = h(\hat{sk}; \hat{\tau}) = \tilde{v}\big)$; and (3) $E \wedge \big(h(\tilde{sk}; \tilde{\tau}) = \tilde{v} \wedge h(\hat{sk}; \hat{\tau}) \neq \tilde{v}\big)$. We denote those events by E_1, E_2, E_3, respectively, and we analyze them as follows:

- First, we observe that, whenever E_1 takes place, the two experiments are identical, as both output $\perp$. Thus, the statistical distance between those two experiments can be upper bounded by $\Pr[E_2] + \Pr[E_3]$.

- Next, we observe, that E_2 happens exactly when there is a collision against $\mathcal{H}_k$, i.e., $(\hat{r}, \hat{\mathsf{sk}}) \neq (\tilde{r}, \tilde{\mathsf{sk}})$, and their hash values collide. By Lemma 4.2.2, we have $\Pr[E_2] < \mathsf{negl}(k)$.

- Finally, we argue that $\Pr[E_3] < \mathsf{negl}(k)$, based on the 1-more extractability property of $\mathcal{H}_k$. In order to exploit that property, we need to relate $\mathsf{Exp}_2^{f,\Sigma,s}$, with the experiment of Definition 4.2.1, $\mathsf{Exp}^{s',h}_{\mathcal{A}_v,\mathcal{A}_s,\mathcal{E}^{\mathcal{H}}_{\mathcal{A}_v}}(1, z_v, z_{\mathcal{E}})$, for some message s', algorithms $\mathcal{A}_v$, $\mathcal{A}_s$, $\mathcal{E}^{\mathcal{H}}_{\mathcal{A}_v}$, and strings z_v, $z_{\mathcal{E}}$. Recall, that, $\mathcal{A}_v$, z_v, $\mathcal{E}^{\mathcal{H}}_{\mathcal{A}_v}$ and $z_{\mathcal{E}}$, have already been defined with respect to h, v, sk, e, and $f = (f_0, f_1)$, in the beginning of the proof. For the remaining we have:

  1. (**Define** $\mathcal{A}_s$): on input τ, s, st, $\mathcal{A}_s(h, \tau, s, st)$, samples $(\tilde{\tau}, \tilde{s}) \leftarrow f_0(\tau, s)$ and outputs $(\tilde{\tau}, \tilde{s})$.
  2. (**Define message** s'): set $s := sk$.

  By the 1-more weak extractability property of $\mathcal{H}_k$, we have

  $$\Pr_{h \leftarrow \mathcal{H}_k}\left[\mathsf{Exp}^{s',h}_{\mathcal{A}_v,\mathcal{A}_s,\mathcal{E}^{\mathcal{H}}_{\mathcal{A}_v}}(1, z_v, z_{\mathcal{E}}) = 1\right] \leq \mathsf{negl}(k),$$

  and notice, that whenever E_3 happens, we also have $\mathsf{Exp}^{s',h}_{\mathcal{A}_v,\mathcal{A}_s,\mathcal{E}^{\mathcal{H}}_{\mathcal{A}_v}}(1, z_v, z_{\mathcal{E}}) = 1$, since $\mathcal{A}_v$ produces a valid, new hash, $\tilde{v}$, $\mathcal{A}_s$, produces a valid preimage for $\tilde{v}$, still the extractor fails. Thus, $\Pr[E_3] < \mathsf{negl}(k)$.

Therefore, we have $\Pr[E_2] + \Pr[E_3] < \mathsf{negl}(k)$, and the two experiments are computationally indistinguishable. □

Claim 5.2.5. *Assuming $\bar{\mathcal{H}}_{\lambda-1}$ is a universal hash function family that outputs $\lambda - 1$ bits, where $\lambda = \omega(\log k)$, we have that $\mathsf{Exp}_2^{f,\Sigma,s} \approx_c \mathsf{Exp}_3^{f,\Sigma,s}$, over the randomness of $\mathsf{Init}, \mathsf{Enc}, \mathcal{E}$, where $\Sigma \leftarrow \mathsf{Init}(1^k)$.*

Proof. In $\mathsf{Exp}_3^{f,\Sigma,s}$ we unfold the encoding procedure and we treat v as leakage over sk (those modifications do induce any statistical difference). The main difference between $\mathsf{Exp}_2^{f,\Sigma,s}$ and $\mathsf{Exp}_3^{f,\Sigma,s}$ is in the way we check the "if" statements of the program that indicate whether the preimage has been modified. We note that the first condition, "If $(b = 1)$", is exactly the same as the condition "If $(\tau, sk, e) = (\tilde{\tau}, \tilde{sk}, \tilde{e})$", since the first bit output by the leakage function indicates weather f_0 has modified (τ, sk). Therefore, this modification does not induce any statistical difference. Then, we analyze the next condition and we observe that there is a difference only when $\bar{h}(\hat{\tau}, \hat{sk}) = \mathsf{lhash}$, still $(\hat{\tau}, \hat{sk}) \neq (\tilde{\tau}, \tilde{sk})$. This happens when the following event, denoted as B, takes place: $\bar{h}(\hat{\tau}, \hat{sk}) = \bar{h}(\tilde{\tau}, \tilde{sk}) \wedge (\hat{\tau}, \hat{sk}) \neq (\tilde{\tau}, \tilde{sk})$. Clearly, the statistical difference between the two experiments is bounded by $\Pr[B]$. Now, since the universal hash function $\bar{h}$ is chosen independently from its inputs, the collision probability is bounded by $2^{\lambda-1} = \mathsf{negl}(k)$. The event B, is exactly the collision event, therefore we have $\Pr[B] \leq \mathsf{negl}(k)$. The proof of the claim is complete. □

Claim 5.2.6. *Assuming $(\mathsf{KGen}, \mathsf{E}, \mathsf{D})$ is semantically secure against λ bits of leakage, we have that for any $f \in \mathcal{F}_{\mathsf{ss}}$ and any message s, $\mathsf{Exp}_4^{f,\Sigma,s} \approx_c \mathsf{Exp}_4^{f,\Sigma,\mathsf{0}}$, over the randomness of Init, Enc, $\mathcal{E}$, where $\Sigma \leftarrow \mathsf{Init}(1^k)$ and $\mathsf{0}$ denotes the zero message.*

Proof. Towards contradiction, assume there exist $f \in \mathcal{F}_{\mathsf{ss}}$, message s, and PPT distinguisher D such that $|\Pr[D(\Sigma, \mathsf{Exp}_4^{f,\Sigma,s}) = 1] - \Pr[D(\Sigma, \mathsf{Exp}_4^{f,\Sigma,\mathsf{0}})] = 1| > \epsilon$, for $\epsilon = 1/\mathrm{poly}(k)$. We are going to define an attacker $\mathcal{A}$ that breaks the semantic security against one-time leakage of $(\mathsf{KGen}, \mathsf{E}, \mathsf{D})$.

$\mathcal{A}$ has hardwired the leakage function $g'_{\tau,\bar{h},h}(sk) := g_{\bar{h},h}(\tau, sk)$ where $\tau \leftarrow \{0,1\}^{\mathrm{poly}(k)}$, $\bar{h} \leftarrow \bar{\mathcal{H}}_{\lambda-1}$, $h \leftarrow \mathcal{H}_k$, and two messages, $s_0 := s$, $s_1 := \mathsf{0}$. Then, on input

$$\Big(e \leftarrow \mathsf{E}_{sk}(s_b), (\mathsf{lmod}, \mathsf{lhash}, v) = g'_{\tau,\bar{h},h}(sk)\Big)$$

it sets $q = \mathsf{Program}(h, \bar{h}, e, v, \mathsf{lmod}, \mathsf{lhash})$, $\Sigma = h$, and outputs $D(\Sigma, q)$, where $\mathsf{Program}$ is defined as follows

$\mathsf{Program}(h, \bar{h}, e, v, \mathsf{lmod}, \mathsf{lhash})$:
$(\tilde{e}, \tilde{v}) \leftarrow f_1(e, v)$
$b \leftarrow (\mathsf{lmod} = 0 \wedge \tilde{e} = e)$
If $v = \tilde{v}$:
  If $(b = 1)$: set $\tilde{s} \leftarrow \mathsf{same}^*$
  Else : set $\tilde{s} \leftarrow \perp$
If $v \neq \tilde{v}$:
  $(\hat{\tau}, \hat{sk}) \leftarrow \mathcal{E}(h, v)$
  set $\tilde{s} \leftarrow \perp$
    If $\bar{h}(\hat{\tau}, \hat{sk}) = \mathsf{lhash}$:
      If $h(\hat{sk}; \hat{s}) = \tilde{v}$, set $\tilde{s} \leftarrow \mathsf{D}_{\hat{sk}}(\tilde{e})$
Output $\tilde{s}$.

It is straightforward to see that $\mathcal{A}$ simulates $\mathsf{Exp}_3^{f,\Sigma,s_b}$, so the advantage of $\mathcal{A}$ in breaking the semantic security of the leakage-resilient encryption scheme matches the advantage of D in distinguishing between $\mathsf{Exp}_3^{f,\Sigma,s_0}$ and $\mathsf{Exp}_3^{f,\Sigma,s_1}$, which by assumption is non-negligible. This leads to a contradiction and the proof of the claim is complete. □

From the above claims we have that for any function f and any message s, $\mathsf{Tamper}_s^{\Sigma,f} \approx_c \mathsf{Exp}_3^{f,\Sigma,0}$, which implies that for any f and any pair of messages s_0, s_1, $\mathsf{Tamper}_{s_0}^{\Sigma,f} \approx_c \mathsf{Tamper}_{s_1}^{\Sigma,f}$, and the proof is complete. □

Length of the CRS. For our KEA based construction, the length of the CRS is roughly $32k$ bits, as we need to hash a $6k$-bit key of an authenticated encryption scheme and then encrypt the message using that key; this would require the parameters for the 16-KEA to be on the CRS, yielding a CRS with $32k$ bits.

5.3 Instantiating authenticated encryption

In the following we provide an instantiation for a one-time leakage-resilient, authenticated, semantically secure symmetric encryption (Definition 2.1.6), against λ bits of leakage. The idea is to combine a leakage-resilient pseudorandom generator [Pie09] with a message authentication code that outputs k bits.

Construction 5.3.1 (Authenticated encryption)**.** *Let* PRG *be a pseudo-random generator,* $\mathsf{PRG} : \{0,1\}^{2\lambda} \to \{0,1\}^{|s|+k}$*, and let* $(\mathsf{Gen}, \mathsf{Mac}, \mathsf{Vrfy})$ *be a message authentication code that outputs tags of length* k *(cf. [KL14]). We define a symmetric encryption scheme* $(\mathsf{KGen}, \mathsf{E}, \mathsf{D})$*, as follows:*

- $\mathsf{KGen}(1^k)$*: sample* $sk \leftarrow \{0,1\}^{2\lambda}$*.*
- $\mathsf{E}_{sk}(\cdot)$*: On input message* s*, compute* $(r_0||r_1) = \mathsf{PRG}(sk)$*, where* $|r_0| = |s|$ *and* $|r_1| = k$*,* $e = r_0 + s$*,* $t = \mathsf{Mac}_{r_1}(e)$*, and outputs* (e, t)*.*
- $\mathsf{D}_{sk}(\cdot)$*: On input* (e, t)*, compute* $(r_0||r_1) = \mathsf{PRG}(sk)$*, and if* $\mathsf{Vrfy}_{r_1}(e, t) = 1$*, output* $s = r_0 - e$*, otherwise output* $\perp$*.*

The PRG of [Pie09] considers $|sk| = 2\lambda/\alpha$, and sustains $\alpha\lambda$ bits of leakage (cf. [SPY+10]), where $\alpha \in [0, 1]$ depends on how strong the underlying assumption is. In the above construction we use the strongest assumption, i.e., $\alpha = 1$, which yields $|sk| = 2\lambda$, assuming weak pseudorandom functions against exponential adversaries. The ciphertext length is $|s| + k$, and by setting $\lambda = 2k + \log^2 k$, which is sufficient for our needs, we receive $|sk| + |e| + |t| = 5k + 2\log^2 k + |s|$. By plugging the above instantiation to our split-state non-malleable code, the total codeword length with respect to the KEA-based ℓ-more wECRH (cf. Section 4.3.4) is $|s| + 9k + 2\log^2(k)$, since the hash and the randomness for computing it, are of size $2k$, each, while with respect to the random oracle-based wECRH (cf. Section 4.4), the total codeword length is $|s| + 6k + 2\log^2(k)$.

In what follows, we provide an instantiation that uses regular PRG, but first we present a useful lemma.

Lemma 5.3.2. *Any* ϵ*-secure one-time message authentication code* $\Pi = (\mathsf{Gen}, \mathsf{Mac}, \mathsf{Vrfy})$ *is* $2^\lambda\epsilon$*-secure against* λ *bits of leakage.*

Proof. (proof sketch) Towards contradiction, assume an attacker $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3)$, issuing a single leakage query g with λ bits of output, against the secret key sk of Π, and breaking its security with probability greater than $2^\lambda\epsilon$. We build an attacker $\mathcal{A}'$ that acts as follows: it samples $g \leftarrow \mathcal{A}_1(1^k)$, makes a guess g on $g(sk)$, and executes the rest of the $\mathsf{LRMAC} - \mathsf{forge}$ experiment with $(\mathcal{A}_2, \mathcal{A}_3)$. Clearly, the probability of wining is equal to the probability of making a correct guess on $g(sk)$, say p_1, times the probability that $\mathcal{A}$ breaks Π in the presence of leakage, say p_2, which by assumption is greater then $2^\lambda\epsilon$. Assuming $H_\infty(g(sk)) \leq \lambda$, the winning probability of $\mathcal{A}'$ is $p_1 \cdot p_2 > \epsilon$, which is a contradiction. □

Construction 5.3.3 (One-time MAC)**.** *Let $\mathcal{H}_{\mathsf{pi}}$ be a pair-wise independent hash function family, $\mathcal{H}_{\mathsf{pi}} = \{h : \mathcal{K} \times \mathcal{M} \to \mathcal{T}\}$. A one-time message authentication code* $(\mathsf{Gen}, \mathsf{Mac}, \mathsf{Vrfy})$ *is defined as follows:*

- Gen*: sample $z \leftarrow \mathcal{K}$, and output h_z.*
- $\mathsf{Mac}(z, \cdot)$*: on input message s, output $t := h_z(s)$.*
- $\mathsf{Vrfy}(z, \cdot)$*: on input s, t, if $h_z(s) = t$, output 1, otherwise output 0.*

It is not hard to see that by instantiating the above construction with $h_{a,b}(s) = a \cdot s + b \mod p$, where p is a k-bit prime, (a, b) is a $2k$-bit key and $\mathcal{M} = \mathcal{T} = \mathbb{Z}_p$, we receive an $1/2^k$-secure message authentication code (this is standard one-time information theoretic MAC). By combining that code with a semantically secure, leakage resilient encryption scheme, we construct an authenticated, semantically secure encryption scheme against λ bits of leakage.

Construction 5.3.4 (Authenticated one-time LR-encryption against λ bits of leakage)**.** *Let $\bar{\mathcal{H}}$ be a hash function family, that outputs k bits, let* PRG *be a pseudo-random generator,* $\mathsf{PRG} : \{0,1\}^k \to \{0,1\}^{|s|}$*, where $|s|$ denotes the length of the message. We define a symmetric encryption scheme* $(\mathsf{KGen}, \mathsf{E}, \mathsf{D})$*, as follows:*

- $\mathsf{KGen}(1^k)$*: sample $r \leftarrow \{0,1\}^{(k+\log^2 k+\lambda)}$, and two random integers a, b, over $\{0,1\}^{k+\lambda}$, and output $sk := (r, a, b)$.*
- $\mathsf{E}_{sk}(\cdot)$*: On input message s, the encryption algorithm computes $\bar{h} \leftarrow \bar{\mathcal{H}}$, $e = \mathsf{PRG}(\bar{h}(r)) + s$, $t = h_{a,b}(\bar{h}||e)$ and outputs $(\bar{h}, e, t)$, where $h_{a,b}(s) := as + b \mod p$ and p is a $k + \log^2 k + \lambda + |s|$-bit prime.*
- $\mathsf{D}_{sk}(\cdot)$*: On input $(\bar{h}, e, t)$, if $t = h_{a,b}(\bar{h}||e)$ output $s = \mathsf{PRG}(\bar{h}(r)) - e$, otherwise output $\perp$.*

Theorem 5.3.5. *Assuming $\bar{\mathcal{H}}$ is a universal hash function family, $\mathcal{H}$ is pairwise independent ($h_{a,b} \in \mathcal{H}$), and one-way functions, Construction 5.3.4 is a one-time leakage-resilient, semantically secure, authenticated encryption scheme against λ bits of leakage.*

Proof. (proof sketch)
Clearly, the above scheme satisfies correctness. Regarding semantic security, by construction we have $H_\infty(r|g(sk)) \geq k + \log^2 k$, for any g that outputs λ bits. Thus, by the

LeftOver Hash Lemma (Lemma 2.1.11), $\bar{h}(r)$ is statistically close to uniform over $\{0,1\}^k$, and $\mathsf{PRG}(h(r))+s$, is computationally indistinguishable from a uniform element in $\{0,1\}^{|s|}$. Since the tag, t, is computed over $(\bar{h}, e)$, it does not reveal any information about the message s, and semantic security follows.

Now, since $h_{a,b}$ belongs to a pairwise independent hash function family (see above), any attacker without leakage access on (a,b), makes a forgery against the above scheme with probability at most $1/2^{(k+\lambda)}$. Thus, by Lemma 5.3.2 unforgeability against λ bits of leakage breaks with probability at most $2^{\lambda} \cdot 1/2^{(k+\lambda)} = \mathsf{negl}(k)$, and the unforgeability property of the scheme breaks with negligible probability in k, even given λ bits of leakage. □

In the above construction, the length of the secret key is $3k + 3\lambda + \log^2 k$ bits while the length of the ciphertext is $2k + 2\log^2 k + 2\lambda + 2|s|$ bits, giving a total of $l(\lambda, s) := 5k + 5\lambda + 3\log^2 k + 2|s|$ bits.

Clearly, the above scheme is not sufficient for getting a rate 1 non-malleable code, thus we combine the above scheme with the following authenticated encryption scheme for which we do not require leakage resilience.

Construction 5.3.6 (Authenticated encryption)**.** *Let* PRG *be a pseudo-random generator,* $\mathsf{PRG} : \{0,1\}^k \to \{0,1\}^{|s|+k}$*, where* $|s|$ *denotes the length of the message, and let* $(\mathsf{Gen}, \mathsf{Mac}, \mathsf{Vrfy})$ *be a CBC message authentication code that outputs tags of length* k *(cf. [KL14]). We define a symmetric encryption scheme* $(\mathsf{KGen}', \mathsf{E}', \mathsf{D}')$*, as follows:*

- $\mathsf{KGen}'(1^k)$*: sample* $r \leftarrow \{0,1\}^k$*, and output* $sk := r$*.*
- $\mathsf{E}'_{sk}(\cdot)$*: On input message* s*, the encryption algorithm computes* $(r_0||r_1) = \mathsf{PRG}(sk)$*, where* $|r_0| = |s|$ *and* $|r_1| = k$*,* $e = r_0 + s$*,* $t = \mathsf{Mac}_{r_1}(e)$*, and outputs* (e,t)*.*
- $\mathsf{D}'_{sk}(\cdot)$*: On input* (e,t)*, compute* $(r_0||r_1) = \mathsf{PRG}(sk)$*, and if* $\mathsf{Vrfy}_{r_1}(e,t) = 1$*, output* $s = e - r_0$*, otherwise output* $\perp$*.*

It is not hard to see that the above construction is secure: r_0 is indistinguishable from random, thus e is indistinguishable from random over the message space. Moreover, the unforgeability property of the message authentication code guarantees the authenticity of the encryption scheme. In the above construction the length of secret key and ciphertext is $2k + |s|$.

The final construction is a combination between constructions 5.3.6 and 5.3.4, for $\lambda = 2k + \log^2 k$ bits of leakage. In order to encrypt a message s, we execute $sk' \leftarrow \mathsf{KGen}'(1^k)$

and $sk \leftarrow \mathsf{KGen}(1^k)$ and we output $(e_1 = \mathsf{E}_{sk}(sk'), e_2 = \mathsf{E}'_{sk'}(s))$, i.e., we encrypt the secret key of an authenticated encryption scheme using leakage-resilient authenticated encryption, and then we encrypt the message using the former scheme. The decryption procedure is straightforward: if $\mathsf{D}_{sk}(e_1) = sk'' \neq \perp$, output $\mathsf{D}'_{sk''}(e_2)$, otherwise, output $\perp$. Correctness and semantic security follow directly by the correctness and semantic security of the underlying schemes. Now, if the attacker modifies e_1, then with high probability $sk'' = \perp$ by the authenticity property of Construction 5.3.4. Assuming, the attacker does not modify e_1, if e_2 is modified then, $\mathsf{D}'_{sk''}(e_2) = \perp$, with overwhelming probability, by the authenticity property of Construction 5.3.6. The final construction has ciphertext and key length $l(2k + \log^2 k, k) + k + |s| = 18k + 8\log^2 k + |s|$.

Chapter 6

Continuous non-malleable codes

6.1 Introduction

The notion of *continuous non-malleable codes* (CNMCs) was introduced by Faust et al. [FMNV14], as an extension to the original notion [DPW10], which considered a one-time adversary. Informally, CNMCs provide simulation-based security, in a setting where the adversary tampers *repeatedly* with the *same* codeword (a notion which is known as *non-persistent tampering*), until the first time he produces an invalid one, in which case the codeword is erased and the adversary looses access to it. The main advantage of CNMCs over the original notion, is that they provide *continuous security* while avoiding *memory erasures*, which, as we have already discussed in Chapter 1, is a feasible but highly problematic process.

6.2 Contributions

In the present Chapter, we leverage the power of leakage-resilient ℓ-more wECRH (cf. Chapter 4) in the continuous setting, and we construct efficient, continuously non-malleable leakage-resilient codes, against split-state adversaries [FMNV14]. Our result is summarized in the following, informally stated theorem.

Theorem 6.2.1 (Informal)**.** *Assuming leakage resilient,* 1*-more* wECRH*, there exists an explicit, leakage-resilient, continuously non-malleable code, against split-state functions.*

By instantiating the above theorem with the leakage-resilient ℓ-more wECRH of the Informal Theorem 4.1.4, we receive continuous non-malleable codes in the CRS model against non-adaptive leakage, and for at least poly-logarithmic (in the security parameter)

number of rounds (cf. Corollary 6.4.13 and the discussion that follows). In addition, from the Informal Theorem 3.1.2, we obtain leakage-resilient continuous non-malleable codes, for polynomially many rounds, assuming random oracles. Our starting point is the construction of [FMNV14], which combines leakage-resilient storage with non-interactive zero-knowledge (NIZK). In our construction we combine leakage-resilient storage with leakage resilient, 1-more wECRH, and as a result we improve the efficiency of [FMNV14] while avoiding the need for trapdoor CRS.[1] In addition, the simulator of [FMNV14] requires leakage proportional to $O(k \log(q) + \lambda)$, while we only require $O(k + \lambda)$, where k is the security parameter, q is the number of rounds that the attacker tampers with the codeword, and λ is the leakage requested by the tampering adversary. As a result, the size of the code becomes independent of q.

6.3 Related work

In [FMNV14], the authors introduce the notion of continuous non-malleable codes and construct computationally secure encoding schemes, for split-state adversaries. In addition to that, they provide an impossibility result for CNMC in the information-theoretic setting, against the same class. Slightly later, Jafargholi and Wichs [JW15], propose an unconditionally secure CNMC, for any class of functions $\mathcal{F}$, such that for any $f \in \mathcal{F}$, (i) the output of f has high entropy, and (ii) f has a limited number of fixed points. The work of Aggarwal et al. [AKO17], builds information-theoretic CNMCs for split-state functions, against *persistent tampering*, which considers adversaries that in round i, receive access to the tampered codeword of round $i-1$, as opposed to the stronger model of *non-persistent tampering*, in which the adversary always tampers with the original codeword. In [OPVV18], the authors construct continuously secure CNMCs in the computational setting, using as a main ingredient a one-time, information-theoretic NMC, while in [DKO$^+$18], Damgård et al., construct information-theoretic CNMC s, against permutations and overwrites. In [FN17], Faonio et al., construct CNMCs for split-state adversaries, using NIZKs. Finally, in [CMTV15, CDTV16], the authors construct information-theoretically secure, continuously non-malleable codes, for bit-wise independent tampering adversaries.

[1] The construction of [FMNV14] requires four NIZK proofs, while we only require two hashes of size at most $2k$.

6.4 Continuous NMC from ℓ-more wECRH

In the current section, we construct *leakage-resilient continuous non-malleable codes* from any *leakage-resilient, ℓ-more weakly extractable, hash function family*, and then, in Section 6.4.1, we provide instantiations. For the needs of the current section, we present a definition of leakage-resilient ℓ-more wECRH, which is stronger than the notion presented in Definition 4.3.15.

Definition 6.4.1 (ℓ-more weakly extractable, leakage-resilient hash function families)**.** *Let $\ell, \lambda \in \mathbb{N}$. An efficiently samplable hash function ensemble $\mathcal{H} = \{\mathcal{H}_k\}_{k\in\mathbb{N}}$, is ℓ-more weakly extractable against λ bits of leakage, if for any* PPT *algorithm $\mathcal{A}_v$ and any $z_v \in \{0,1\}^{\mathrm{poly}(k)}$, there exist a* PPT *extractor $\mathcal{E}^{\mathcal{H}}_{\mathcal{A}_v}$ and $z_{\mathcal{E}} \in \{0,1\}^{\mathrm{poly}(k)}$, such that for all* PPT *algorithms $\mathcal{A}_s$, any large $k \in \mathbb{N}$ and any vector of messages $\mathbf{s} = (s_1, \ldots, s_\ell)$, we have $\Pr_{h\leftarrow\mathcal{H}_k}\left[\mathsf{Exp}^{\mathbf{s},h}_{\mathcal{A}_v,\mathcal{A}_s,\mathcal{E}_{\mathcal{A}_v}}(\ell,\lambda,z_v,z_{\mathcal{E}})=1\right] \leq \mathsf{negl}(k)$, where,*

$\mathsf{Exp}^{\mathbf{s},h}_{\mathcal{A}_v,\mathcal{A}_s,\mathcal{E}_{\mathcal{A}_v}}(\ell,\lambda,z_v,z_{\mathcal{E}})$:
$\tau_i \leftarrow \{0,1\}^{\mathrm{poly}(k)}, v_i = h(s_i;\tau_i), i \in [\ell]$ *(* **hash computation** *)*
$\mathbf{t} = (\tau_1,\ldots,\tau_\ell), \mathbf{v} = (v_1,\ldots,v_\ell)$
$(\tilde{v}, st) \leftarrow \mathcal{A}_v^{\mathcal{O}^\lambda(\mathbf{t},\cdot)}(h,\mathbf{v},z_v)$ *(* **hash tampering** *)*
$(\hat{\tau},\hat{s}) \leftarrow \mathcal{E}_{\mathcal{A}_v}(h,\mathbf{v},z_{\mathcal{E}})$ *(* **pre-image extraction** *)*
$(\tilde{\tau},\tilde{s}) \leftarrow \mathcal{A}_s(h,\mathbf{t},\mathbf{s},st)$ *(* **pre-image tampering** *)*
If $h(\tilde{s};\tilde{\tau}) = \tilde{v} \wedge \forall i : \tilde{v} \neq v_i \wedge h(\hat{s};\hat{\tau}) \neq \tilde{v}$, *return 1*
otherwise, return 0

Below, we define the notion of *leakage-resilient storage* due to [DDV10, DF11, FMNV14], which is one of the main building blocks of the proposed scheme.

Definition 6.4.2 (Leakage-resilient storage [FMNV14])**.** *Let* $(\mathsf{LRS}_{\mathsf{enc}},\mathsf{LRS}_{\mathsf{dec}})$ *be a coding scheme. For any algorithm $\mathcal{A}$, message m, $\theta \in \{0,1\}$, and $k \in \mathbb{N}$ we define*

$$\mathsf{Leak}^{\theta}_{\mathcal{A},m}(k) := \left\{(s_0,s_1) \leftarrow \mathsf{LRS}_{\mathsf{enc}}(1^k,m); out \leftarrow \mathcal{A}^{\mathcal{O}^\lambda(s_0,\cdot),\mathcal{O}^\lambda(s_1,\cdot)}; \textit{Output: } (s_\theta, out)\right\}.$$

Then, $(\mathsf{LRS}_{\mathsf{enc}},\mathsf{LRS}_{\mathsf{dec}})$ *is a λ-leakage-resilient storage (λ-*LRS*), if for any algorithm $\mathcal{A}$, messages m_0, $m_1 \in \{0,1\}^{\mathrm{poly}(k)}$, $\theta \in \{0,1\}$, and all, sufficiently large $k \in \mathbb{N}$,*

$$\left\{\mathsf{Leak}^{\theta}_{\mathcal{A},m_0}(k)\right\}_{k\in\mathbb{N}} \approx \left\{\mathsf{Leak}^{\theta}_{\mathcal{A},m_1}(k)\right\}_{k\in\mathbb{N}}.$$

Here, we follow the definition of [FMNV14], which is stronger than previous definitions in the sense that the attacker is allowed to see one of the two shares, after the completion of the leakage experiment. As the authors suggest in [FMNV14], the scheme of [DF11] satisfies this stronger notion.

Our construction of non-malleable codes is inspired by [FMNV14], thus we first revisit their construction. To encode a message m, the encoder of [FMNV14], computes $(s_0, s_1) \leftarrow \mathsf{LRS}_{\mathsf{enc}}(m)$ and outputs $((s_0, v_1, \pi_1, \pi_0), (s_1, v_0, \pi_0, \pi_1))$, where $\mathsf{LRS}_{\mathsf{enc}}$ is the encoder of a leakage-resilient storage (LRS) scheme, $v_i = h(s_i)$ and h is a member of a collision resistant hash function family, and π_i is a robust non-interactive zero knowledge proof, proving knowledge of the witness (pre-image) s_i of v_i, with label v_{1-i}.[2]

Our construction, which is defined below, improves the efficiency of [FMNV14] by combining LRS with leakage-resilient 1-more wECRH.

Construction 6.4.3 (Continuous NMC from wECRH and LRS.)**.** *Let $\mathcal{H}_k$, $\bar{\mathcal{H}}_k$, be hash function families and $(\mathsf{LRS}_{\mathsf{enc}}, \mathsf{LRS}_{\mathsf{dec}})$ be a leakage-resilient storage scheme. We define an encoding scheme $(\mathsf{Init}, \mathsf{Enc}, \mathsf{Dec})$ as follows:*

- $\mathsf{Init}(1^k)$*: Sample $h \leftarrow \mathcal{H}_k$, $\bar{h} \leftarrow \bar{\mathcal{H}}_k$, and set $\Sigma := (h, \bar{h})$.*
- $\mathsf{Enc}(\Sigma, \cdot)$*: Let m be the input to the encoder. The encoder samples $(s_0, s_1) \leftarrow \mathsf{LRS}_{\mathsf{enc}}(1^k, m)$, $\tau_0, \tau_1 \leftarrow \{0,1\}^{\mathrm{poly}(k)}$, computes $\bar{v}_0 \leftarrow \bar{h}(\tau_0 || s_0)$, $\bar{v}_1 \leftarrow \bar{h}(\tau_1 || s_1)$, and outputs $((\tau_0, s_0, v_1), (\tau_1, s_1, v_0))$, where $v_0 \leftarrow h(s_0 || \bar{v}_1; \tau_0)$, $v_1 \leftarrow h(s_1 || \bar{v}_0; \tau_1)$.*
- $\mathsf{Dec}(\Sigma, \cdot)$*: On input $((\tau_0, s_0, v_1), (\tau_1, s_1, v_0))$, for $i \in \{0,1\}$, if $h\left(s_i || \bar{h}(\tau_{1-i} || s_{1-i}); \tau_i\right) = v_i$, output $\mathsf{LRS}_{\mathsf{dec}}(1^k, (s_0, s_1))$, otherwise, output $\perp$.*

In what follows we will assume that $\mathcal{H}$ is a 1-more wECRH and this is essential for proving security. Observe that, if $\mathcal{H}$ is 0-more extractable, then it can be malleable (as it is proven in Chapter 4 and [KLT16]) and security cannot be proved, as generic LRS schemes do not provide non-malleability, thus the attacker could create related codewords. The 1-more extractability property resolves this issue: even if the attacker is given access to a valid hash value v_i, it cannot produce a valid hash value unless it knows a valid pre-image.

In what follows, we briefly discuss the main ideas behind our proof, while highlighting the differences from [FMNV14]. The security of our scheme relies on three primitives, namely, on leakage-resilient 1-more wECRH, on the collision resistance property of $\bar{\mathcal{H}}$ and

[2]The labels are used to bind together the two parts of the memory.

the security of LRS. Our adversary is denoted by $\mathcal{A}'$ and is depicted in Figure 6.1, while its main subroutine, TComp, is depicted in Figure 6.2. $\mathcal{A}'$ simulates the tampering experiment against the codeword, while given split-state leakage over $(s_0, s_1) \leftarrow \mathsf{LRS}_{\mathsf{enc}}(m)$, where m denotes the message. In step 1, $\mathcal{A}'$ samples the elements required for simulating the codewords inside the leakage oracles, i.e., it samples hash functions h, $\bar{h}$, and randomness τ_i, for h. Then, it makes a guess, j^*, on the index of the round in which the attacker produces an invalid codeword. Such a round is called a "self-destruct" round.[3] Then, in step 2, the adversary leaks the actual hashes over s_0, s_1, and in this way it simulates perfectly the codewords inside the leakage oracles without using a trapdoor CRS, i.e., for $i \in \{0,1\}$, $c_i = (\tau_i, s_i, v_{1-i})$ is perfectly simulated inside $\mathcal{O}^{\lambda}(s_i, \cdot)$. This approach is in contrast to [FMNV14], in which the authors use robust NIZKs, and simulate the codeword inside the oracles using simulated proofs, that require a trapdoor CRS.

In step 3, $\mathcal{A}'$ executes $\mathcal{A}$ inside the leakage oracles and verifies if j^* is a correct guess for the self-destruct round. As we prove, this holds, if for all rounds before j^* the executions inside the two oracles are identical, while they differ in round j^*. The challenge here, is to execute $\mathcal{A}$ inside the oracles, as each $\mathcal{O}^{\lambda}(s_i, \cdot)$ gives access to $c_i = (\tau_i, s_i, v_{1-i})$, but provides no information about s_{1-i} (recall that τ_{1-i}, v_i can be simulated), thus it is unclear how to provide the adversary with $\tilde{c}_{1-i}$. We discuss how to resolve this issue, by first considering a non-leakage tampering adversary, $\mathcal{A}$. The main idea behind step 3 of $\mathcal{A}'$, is as follows: $\mathcal{A}'$ executes $\mathcal{A}$ inside $\mathcal{O}^{\lambda}(s_i, \cdot)$, and for each tampering query (f_0, f_1) of $\mathcal{A}$, $\mathcal{A}'$ computes $\tilde{c}_i \leftarrow f_i(c_i)$, and uses the 1-more wECRH property of $\mathcal{H}$ to extract $\tilde{c}_{1-i}$. When considering adversaries that issue leakage queries, $\mathcal{A}'$ replies to those queries by executing repeatedly TComp (cf. Figure 6.2) against the two oracles (cf. step 3-(a) in Figure 6.1). In steps 3-(b),(c), $\mathcal{A}'$ verifies if j^* is a self-destruct round, by leaking the hashes of the replies sent to $\mathcal{A}$ inside the oracles. We note that, our strategy is similar to [MSD16], but different than [FMNV14] in which the adversary executes binary search to compute the exact value of the index, requiring leakage proportional to $O(k \log(q) + \lambda)$, while we only require $O(k + \lambda)$. In Step 4, $\mathcal{A}'$ learns s_0 and simulates the tampered execution in the same way it does in Step 3. Finally, in contrast to [FMNV14], in which extractability is easily implied by the robust NIZKs, proving extractability in the continuous setting using ℓ-more wECRH, which is a one-time primitive, is non-trivial.

For starters, we prove that the above construction satisfies the uniqueness property

[3] We can always assume that such a round exists, since for any $\mathcal{A}$ that is not producing an invalid codeword, we can construct another adversary that does so and has the same advantage with $\mathcal{A}$, cf. [FMNV14].

(cf. Definition 2.3.4), which is required for achieving non-malleability in the continuous setting.

Lemma 6.4.4. *Assuming $\mathcal{H}_k$ is a collision resistant hash function family, the split-state code of Construction 6.4.3 satisfies the uniqueness property.*

Proof. Let $(h, \bar{h}) \in \mathcal{H}_k \times \bar{\mathcal{H}}_k$, and let $\mathcal{H}_k$ be a collision resistant hash function family. Towards contradiction, assume there exists a PPT attacker $\mathcal{A}$ that, given $(h, \bar{h})$, it produces two distinct, valid codewords, (c_0, c_1), (c_0, c_1'), with probability greater than $\epsilon = 1/\text{poly}(k)$, i.e., $\mathcal{A}$ produces $c_0 = (\tau_0, s_0, v_1)$, $c_1 = (\tau_1, s_1, v_0)$, $c_1' = (\tau_1', s_1', v_0')$, where $c_1 \neq c_1'$,[4] with probability ϵ. We construct an adversary $\mathcal{A}'$, that given $h \leftarrow \mathcal{H}_k$, it breaks the collision resistance property of $\mathcal{H}_k$ with non-negligible probability: $\mathcal{A}'$ samples $\bar{h} \leftarrow \bar{\mathcal{H}}_k$ and

$$(\tau_0, s_0, v_1), (\tau_1, s_1, v_0), (\tau_1', s_1', v_0') \leftarrow \mathcal{A}(h, \bar{h}),$$

and outputs (τ_1, s_1), (τ_1', s_1'). Let $\bar{v}_0 \leftarrow \bar{h}(\tau_0 || s_0)$. By the validity of the codewords and Construction 6.4.3, we have that $h(s_1 || \bar{v}_0; \tau_1) = v_1 = h(s_1' || \bar{v}_0; \tau_1')$. Conditioned on $c_1 \neq c_1'$ we have that $(\tau_1, s_1) \neq (\tau_1', s_1')$: if $(\tau_1, s_1) = (\tau_1', s_1')$, we also have that $v_0 = v_0'$ and the codewords are equal. Thus, conditioned on $c_1 \neq c_1'$ we have that $(\tau_1, s_1) \neq (\tau_1', s_1')$ and $\mathcal{A}'$ breaks the collision resistance property of $\mathcal{H}_k$ with non-negligible probability, ϵ. □

Below we prove non-malleability of Construction 6.4.3 in the continuous setting with respect to any 1-more wECRH.

Theorem 6.4.5. *Let $k, \lambda, \lambda' \in \mathbb{N}$, and let b be polynomial in k. Assuming $\mathcal{H}_k$ is a leakage-resilient 1-more* wECRH *function family against λ bits of leakage, that outputs $b(k)$ bits, $\bar{\mathcal{H}}_k$, $\hat{\mathcal{H}}_k$, are collision resistant hash function families that output k bits, and $(\mathsf{LRS}_{\mathsf{enc}}, \mathsf{LRS}_{\mathsf{dec}})$ is a λ'-*LRS*, for $\lambda' \geq 2\lambda + 2b(k) + 4k + 1$. Then, the encoding scheme $(\mathsf{Init}, \mathsf{Enc}, \mathsf{Dec})$ of Construction 6.4.3 is a (q, λ)-*CNMLR *code (cf. Definition 2.3.7), for $q = \text{poly}(k)$.*

Proof. Towards contradiction, assume there exists a pair of messages m_0, m_1, PPT adversary $\mathcal{A}$ and PPT distinguisher D, such that for infinitely many $k \in \mathbb{N}$,

$$\left| \Pr\left[\mathsf{D}\left(\mathsf{Tamper}^{\mathsf{cnmlr}}_{\mathcal{A},m_0}(k)\right) = 1\right] - \Pr\left[\mathsf{D}\left(\mathsf{Tamper}^{\mathsf{cnmlr}}_{\mathcal{A},m_1}(k)\right) = 1\right] \right| > \epsilon,$$

for $\epsilon = 1/\text{poly}(k)$. Here, $\mathsf{Tamper}^{\mathsf{cnmlr}}_{\mathcal{A},m_i}(k)$ is the experiment of Definition 2.3.7 with respect to $\mathcal{A}$, m_i. We will use m_0, m_1, $\mathcal{A}$, D, to construct m_0', m_1', $\mathcal{A}'$, D', for which

$$\left| \Pr\left[\mathsf{D}'\left(\mathsf{Leak}^0_{\mathcal{A}',m_0'}(k)\right) = 1\right] - \Pr\left[\mathsf{D}'\left(\mathsf{Leak}^0_{\mathcal{A}',m_1'}(k)\right) = 1\right] \right| > \epsilon',$$

[4] The case where $c_0 \neq c_0'$ is symmetric.

for $\epsilon' = 1/\text{poly}(k)$ and infinitely many k, where $\mathsf{Leak}^0_{\mathcal{A}',m'_i}(k)$ is the experiment of Definition 6.4.2, with respect to $\mathcal{A}'$, m'_i. The idea is that $\mathcal{A}'$ will play in $\mathsf{Leak}^0_{\mathcal{A}',m'_b}(k)$, for $b \in \{0,1\}$, interacting with $\mathcal{O}^\lambda(s_0,\cdot)$, $\mathcal{O}^\lambda(s_1,\cdot)$, where (s_0,s_1) follows $\mathsf{LRS}_{\mathsf{enc}}(m_b)$, and it will simulate $\mathsf{Tamper}^{\mathsf{cnmlr}}_{\mathcal{A},m_b}(k)$, through its access to the aforementioned oracles. $\mathcal{A}'$ executes $\mathcal{A}$ using always the same randomness. Assume $\mathcal{A}$ issues q leakage/tampering queries and that there is always a round in which self-destruct occurs, i.e., a round in which the output of the decoder in the real experiment will be $\perp$. This round is denoted by j_{d}. We define $m'_0 := m_0$, $m'_1 := m_1$ and $\mathcal{A}'$ is defined in Figure 6.1.

The definition of the distinguisher D' against $\mathsf{Leak}^0_{\mathcal{A}',m_b}(k)$ follows.

Algorithm D': D' receives the output of $\mathcal{A}'$, (out,d), and if $d=1$ it outputs $b' \leftarrow \mathsf{D}(out)$, otherwise it outputs $b' \leftarrow \{0,1\}$.

Claim 6.4.6. *For any message m and all sufficiently large k, $\mathcal{A}'$ simulates perfectly $\mathsf{Enc}(\Sigma,m)$ inside the leakage oracles by leaking $2k+2b(k)$ bits during the execution of $\mathsf{Leak}^0_{\mathcal{A}',s_b}(k)$, where $\Sigma \leftarrow \mathsf{Init}(1^k)$.*

Proof. By Construction 6.4.3 and the definition of $\mathcal{A}'$, it is not hard to see that c_0 (resp. c_1) is perfectly simulated inside the oracle $\mathcal{O}^\lambda(s_0,\cdot)$ (resp. $\mathcal{O}^\lambda(s_1,\cdot)$). $\mathcal{A}'$ initially samples $h \leftarrow \mathcal{H}_k$, $\bar{h} \leftarrow \bar{\mathcal{H}}_k$ (the CRS), and τ_0, $\tau_1 \leftarrow \{0,1\}^{\text{poly}(k)}$. Then, by querying the leakage oracles with $(\mathcal{L}_0,\mathcal{L}_1)$, where $\mathcal{L}_i(s_i) := \bar{h}(\tau_i||s_i)$, it receives $\bar{v}_0$, $\bar{v}_1$ and finally, by leaking $(\mathcal{L}'_0,\mathcal{L}'_1)$, where $\mathcal{L}'_i(s_i) := h(s_i||\bar{v}_{1-i};\tau_i)$, it receives v_0, v_1. All the remaining leakage queries against $\mathcal{O}^\lambda(s_0,\cdot)$ (resp. $\mathcal{O}^\lambda(s_1,\cdot)$) depend on τ_0, v_1 (resp. τ_1, v_0), and the execution inside the oracles takes place over (c_0,c_1). □

Claim 6.4.7. *Let $\mathbf{lk}_{\mathsf{Real}}$ and $\mathbf{t}_{\mathsf{Real}}$ be the vectors of the replies to the first j^*-1 leakage and tampering, respectively, queries made by $\mathcal{A}$ in $\mathsf{Tamper}^{\mathsf{cnmlr}}_{\mathcal{A},m_b}(k)$. Then, conditioned on $j_{\mathsf{d}} = j^*$, for any message m and all sufficiently large k, $\mathbf{lk}[1:j^*-1] \approx_c \mathbf{lk}_{\mathsf{Real}}$, $\mathbf{t}_{\mathsf{Real}} \approx_c \mathbf{t}_i[1:j^*-1]$, for $i \in \{0,1\}$, over the randomness of $\mathsf{Tamper}^{\mathsf{cnmlr}}_{\mathcal{A},m_b}(k)$, $\mathsf{Leak}^0_{\mathcal{A}',m_b}(k)$.*

Proof. Using strong induction, we prove that conditioned on $j_{\mathsf{d}} = j^*$, $\mathbf{lk}[1:j^*-1] \approx_c \mathbf{lk}_{\mathsf{Real}}$, $\mathbf{t}_{\mathsf{Real}} \approx_c \mathbf{t}_i[1:j^*-1]$, for $i \in \{0,1\}$, assuming that $j_{\mathsf{d}} > 1$, otherwise the claim holds trivially.

Algorithm $\mathcal{A}'$

1. **(Setup)**: Sample $h \leftarrow \mathcal{H}_k$, $\bar{h} \leftarrow \bar{\mathcal{H}}_k$, $\hat{h} \leftarrow \hat{\mathcal{H}}_k$, τ_0, $\tau_1 \leftarrow \{0,1\}^{\mathrm{poly}(k)}$, $j^* \leftarrow [q]$.
2. **(Hash leaking)**:
   a) For $i \in \{0,1\}$, define $\mathcal{L}_i(s_i) := \bar{h}(\tau_i||s_i)$ and issue the leakage query $(\mathcal{L}_0, \mathcal{L}_1)$ against $\mathcal{O}^\lambda(s_0,\cdot)$, $\mathcal{O}^\lambda(s_1,\cdot)$. Let $\bar{v}_0$, $\bar{v}_1$, be the corresponding leaked values.
   b) For $i \in \{0,1\}$, define $\mathcal{L}'_i(s_i) := h(s_i||\bar{v}_{1-i};\tau_i)$ and issue the leakage query $(\mathcal{L}'_0, \mathcal{L}'_1)$ against $\mathcal{O}^\lambda(s_0,\cdot)$, $\mathcal{O}^\lambda(s_1,\cdot)$. Let v_0, v_1, be the leaked values.
3. **(Verifying j^*)**: Let $\mathbf{lk}$ be a $q \times 2$ zero matrix and for $j \in [q]$, $i \in \{0,1\}$ define the leakage function $\mathcal{L}_i^j(s_i, \mathbf{lk})$ that computes $\mathsf{TComp}_q(i, \tau_i, s_i, v_{1-i}, \mathbf{lk}, j)$ (cf. Figure 6.2) and outputs its first coordinate, i.e., $\mathbf{lk}$.
   a) For $j = 1, \ldots, q$: (i) $\mathbf{lk}' \leftarrow \mathcal{L}_0^j(s_0, \mathbf{lk})$, $\mathbf{lk}'' \leftarrow \mathcal{L}_1^j(s_1, \mathbf{lk}')$, (ii) set $\mathbf{lk} \leftarrow \mathbf{lk}''$.
   b) Let $\bar{\mathsf{L}}_0(\cdot)$ be the leakage function that on input s_0, it computes $(\sim, \mathbf{t}_0) \leftarrow \mathsf{TComp}_q(0, \tau_0, s_0, v_1, \mathbf{lk}, q)$, $lk \leftarrow \hat{h}(\mathbf{t}_0[1 : j^* - 1])$, $lk' \leftarrow \hat{h}(\mathbf{t}_0[j^*])$, and outputs (lk, lk'). Send $\bar{\mathsf{L}}_0(\cdot)$ to $\mathcal{O}^\lambda(s_0,\cdot)$.
   c) Define $\bar{\mathsf{L}}_1(s_1)$ that,
      i. Computes $(\sim, \mathbf{t}_1) \leftarrow \mathsf{TComp}_q(1, \tau_1, s_1, v_0, \mathbf{lk}, q)$.
      ii. If $lk = \hat{h}(\mathbf{t}_1[1 : j^* - 1])$ and $lk' \neq \hat{h}(\mathbf{t}_1[j^*])$, output 1, otherwise output 0.

      $\bar{\mathsf{L}}_1(\cdot)$ is executed against $\mathcal{O}^\lambda(s_1,\cdot)$, and let d be the bit output by the leakage query.
   d) Receive s_0 (the leakage queries have ended).
4. **(Simulating tampering and leakage queries)**: Set $c_0 := (\tau_0, s_0, v_1)$.
   Execute $\mathcal{A}$ and for $j = 1, \ldots, q$:
   Receive (g_0^j, g_1^j) from $\mathcal{A}$, send $\mathbf{lk}[j]$ to it, receive $f^j = (f_0^j, f_1^j)$ from $\mathcal{A}$ and:
   - if $j \geq j^*$ send $\perp$ to $\mathcal{A}$,
   - otherwise, compute $\tilde{c}_0 := (\tilde{\tau}_0, \tilde{s}_0, \tilde{v}_1) = f_0^j(\tau_0, s_0, v_1)$ and
     a) If $\tilde{c}_0 = c_0$, send same^* to $\mathcal{A}$.
     b) If $\tilde{c}_0 \neq c_0$:
        If $\tilde{v}_1 = v_1$, send $\perp$ to $\mathcal{A}$, otherwise, $(\hat{\tau}_1, \hat{s}_1, \hat{v}_0) \leftarrow \mathcal{E}_{0,j}(h, v_1, z_{\mathcal{E}_{0,j}})$.
        – If $\bar{h}(\tilde{\tau}_0||\tilde{s}_0) = \hat{v}_0$ and $h(\hat{s}_1||\hat{v}_0;\hat{\tau}_1) = \tilde{v}_1$, set $v_0 := h\left(\tilde{s}_0||\bar{h}(\hat{\tau}_1||\hat{s}_1);\tilde{\tau}_0\right)$, and $\tilde{c}_1 := (\hat{r}_1, \hat{s}_1, v_0)$. Send $(\tilde{c}_0, \tilde{c}_1)$ to $\mathcal{A}$.
        – Otherwise, send $\perp$ to $\mathcal{A}$.
5. **(Output)**: Let *out* be the output of $\mathcal{A}$ after the completion of the previous step. $\mathcal{A}'$ outputs (out, d).

Figure 6.1: The algorithm $\mathcal{A}'$ playing in $\mathsf{Leak}^0_{\mathcal{A}',m_b}(k)$.

Algorithm TComp_q

Input: $i, \tau_i, s_i, v_{1-i}, \mathbf{lk}, r$.
Set $c_i := (\tau_i, s_i, v_{1-i})$ and let $\mathbf{t}$ be the zero vector with q coordinates.

For $j = 1, \ldots, r$:

- Receive (g_0^j, g_1^j) from $\mathcal{A}$ and
  1. If $j < r$, send $\mathbf{lk}[j]$ to $\mathcal{A}$ and receive (f_0^j, f_1^j) from it.
  2. If $j = r$, set $\mathbf{lk}[j, i+1] = g_i^j(s_i)$ and **break**.
- Compute $\tilde{c}_i := (\tilde{\tau}_i, \tilde{s}_i, \tilde{v}_{1-i}) = f_i^j(\tau_i, s_i, v_{1-i})$.
  1. If $\tilde{c}_i = c_i$, set $\mathbf{t}[j] = \mathsf{same}^*$ and send same^* to $\mathcal{A}$.
  2. If $\tilde{c}_i \neq c_i$, then if $\tilde{v}_{1-i} = v_{1-i}$, set $\mathbf{t}[j] = \perp_i$ and send $\perp$ to $\mathcal{A}$, otherwise, sample $(\hat{\tau}_{1-i}, \hat{s}_{1-i}, \hat{v}_i) \leftarrow \mathcal{E}_{i,j}(h, v_{1-i}, z_{\mathcal{E}_{i,j}})$.
     - If $\bar{h}(\tilde{\tau}_i || \tilde{s}_i) = \hat{v}_i$ and $h(\hat{s}_{1-i} || \hat{v}_i; \hat{\tau}_{1-i}) = \tilde{v}_{1-i}$, set $\mathbf{t}[j] = (\tilde{c}_0, \tilde{c}_1)$, where $\tilde{c}_i$ is defined above, $\tilde{c}_{1-i} = (\hat{r}_{1-i}, \hat{s}_{1-i}, v_i)$ and $v_i = h\left(\tilde{s}_i || \bar{h}(\hat{\tau}_{1-i} || \hat{s}_{1-i}); \tilde{\tau}_i\right)$. Send $\mathbf{t}[j]$ to $\mathcal{A}$.
     - Otherwise, set $\mathbf{t}[j] = \perp_i$ and send $\perp$ to $\mathcal{A}$.

Output $(\mathbf{lk}, \mathbf{t})$.

Figure 6.2: The algorithm TComp executed by $\mathcal{A}'$.

Base, $j = 1$: By executing $\mathsf{TComp}_q(i, \tau_i, s_i, v_{1-i}, \mathbf{lk}, 1)$, for $i \in \{0, 1\}$ (Step 3-a), it is clear that $\mathcal{A}'$ computes the replies to (g_0^1, g_1^1) as in the real execution. Thus, $\mathbf{lk}[1] = \mathbf{lk}_{\mathsf{Real}}[1]$. Regarding, the replies to the tampering queries produced by the execution of $\mathsf{TComp}_q(i, \tau_i, s_i, v_{1-i}, \mathbf{lk}, 1)$ inside $\mathcal{O}^\lambda(s_i, \cdot)$, we consider the following cases.

- $\exists i : \tilde{c}_i = c_i$: Assume that for some $i \in \{0, 1\}$, $\tilde{c}_i = c_i$. Then by assumption we have that j is not a round in which self-destruct occurs, and in order for the tampered codeword to be valid, it must be the case that $\tilde{c}_{1-i} = c_{1-i}$ with overwhelming probability, otherwise we can use (f_0^1, f_1^1) to break the uniqueness property of the encoding scheme (cf. Lemma 6.4.4), by simulating the first round of execution. Thus, for $i \in \{0, 1\}$, $\mathbf{t}_i[j] = \mathsf{same}^*$, which matches the reply $\mathbf{t}_{\mathsf{Real}}[1]$ of the tampering oracle in the real execution, since by Claim 6.4.6, c_i is perfectly simulated inside $\mathcal{O}^\lambda(s_i, \cdot)$.

- $\forall i : \tilde{c}_i \neq c_i \wedge \exists j : \tilde{v}_j = v_j$: Let E_i be the event in which $\tilde{c}_i \neq c_i \wedge \tilde{v}_{1-i} = v_{1-i}$, and E the event of not having a round with self-destruct. We prove that $\Pr[E_i \wedge E] \leq \mathsf{negl}(k)$, for $i \in \{0, 1\}$. Towards contradiction, assume that for some i in $\{0, 1\}$, $\Pr[E_i \wedge E] > \epsilon_i'$,

for $\epsilon'_i = 1/\text{poly}(k)$. Then, we have a valid codeword for which $(\tau_i, s_i) \neq (\tilde{\tau}_i, \tilde{s}_i)$ and $\tilde{v}_{1-i} = h(\tilde{s}_{1-i}||\bar{h}(\tilde{\tau}_i||\tilde{s}_i); \tilde{\tau}_{1-i}) = h(s_{1-i}||\bar{h}(\tau_i, s_i); \tau_{1-i}) = v_{1-i}$, and we can use f^j to break the collision resistance property of h with non-negligible probability ϵ_i, by simulating $\mathsf{Tamper}^{\mathsf{cnmlr}}_{\mathcal{A},m_b}(k)$. Thus, such an event never happens with non-negligible probability during the execution of $\mathsf{Tamper}^{\mathsf{cnmlr}}_{\mathcal{A},m_b}(k)$ and $\mathsf{Leak}^0_{\mathcal{A}',m}(k)$.

- $\underline{\forall i : \tilde{v}_i \neq v_i}$: In order to prove consistency between $\mathsf{Tamper}^{\mathsf{cnmlr}}_{\mathcal{A},m_b}(k)$ and $\mathsf{Leak}^0_{\mathcal{A}',m}(k)$, we need to define two extractors, $\mathcal{E}_{i,1}$, for $i \in \{0,1\}$, and the corresponding auxiliary inputs, and prove that the extracted values are consistent with $\mathsf{Tamper}^{\mathsf{cnmlr}}_{\mathcal{A},m_b}(k)$. For this reason we relate (see below) the execution of TComp inside the oracles, with the ℓ-more experiment of Definition 6.4.1.

In the execution inside $\mathcal{O}^\lambda(s_i, \cdot)$, the adversary is given direct access to $c_i = (\tau_i, s_i, v_{1-i})$ and leakage access over c_{1-i}, and tampers with the hash v_{1-i} and some auxiliary values (τ_i, s_i). We relate this adversary to $\mathcal{A}_v$ of Definition 6.4.1 by defining a program $\mathcal{A}_{v,i}$ with auxiliary input $z_{v,i}$, as follows:

1. **Program** $\mathcal{A}^{\mathcal{O}^\lambda(\tau_{1-i},\cdot)}_{v,i}(h, v_{1-i}, z_{v,i})$:
   - Sample $(g^1_0, g^1_1) \leftarrow \mathcal{A}(h)$ and parse $z_{v,i}$ as (τ_i, s_i, s_{1-i}).
   - Query $\mathcal{O}^\lambda(\tau_{1-i}, \cdot)$ with $g_{s_{1-i}}(x) := \bar{h}(x||s_{1-i})$ and let $\bar{v}_{1-i}$ be the answer.
   - Set $v_i = h(s_i||\bar{v}_{1-i}; \tau_i)$.
   - Define $g_{v_i,s_{1-i}}(\tau_{1-i}) := g^1_{1-i}(\tau_{1-i}, s_{1-i}, v_i)$, send $g_{v_i,s_{1-i}}$ to $\mathcal{O}^\lambda(\tau_{1-i}, \cdot)$ and let w_{1-i} be the answer.
   - Compute $w_i \leftarrow g^1_i(\tau_i, s_i, v_{1-i})$, send (w_0, w_1) to $\mathcal{A}$ and receive (f^1_0, f^1_1).
   - **Output**: $([f^1_i(\tau_i, s_i, v_{1-i})]_3, st)$, where $st := (f^j_i(\tau_i, s_i, v_{1-i}), z_{v,i}, v_{1-i})$.
2. (**Auxiliary input for** $\mathcal{A}_{v,i}$): set $z_{v,i} = (\tau_i, s_i, s_{1-i})$.
3. (**Existence of the extractor,** $\mathcal{E}_{i,1}$**, and auxiliary input,** $z_{\mathcal{E}_{i,1}}$): Given $\mathcal{A}_{v,i}$ and $z_{v,i}$, by the 1-more extractability property of $\mathcal{H}_k$ under leakage, there exists an extractor $\mathcal{E}_{i,1}$ for $\mathcal{A}_{v,i}$, with auxiliary input, $z_{\mathcal{E}_{i,1}}$, that computes $(\hat{\tau}_{1-i}, \hat{s}_{1-i}, \hat{v}_i) \leftarrow \mathcal{E}_{i,1}(h, v_{1-i}, z_{\mathcal{E}_{i,1}})$.

Clearly, $\mathcal{A}_{v,i}$ is an admissible attacker against $\mathcal{H}_k$, that produces a tampered hash value $\tilde{v}_{1-i}$, as $\mathcal{A}$ does in the first round of the execution inside $\mathcal{O}^\lambda(s_i, \cdot)$. Now we relate the execution inside $\mathcal{O}^\lambda(s_{1-i}, \cdot)$ with a program $\mathcal{A}_{s,1-i}$ that outputs a tampered pre-image, and we define the message s of the experiment of Definition 6.4.1.

1. **Program** $\mathcal{A}_{s,1-i}(h,\tau,s,st)$:
   - Parse s as $s_{1-i}||\bar{v}_i$, and set $\tau_{1-i} := \tau$.
   - Compute $v_i := h(s_i||\bar{h}(\tau_{i-1}||s_{i-1});\tau_i)$,[5] $(\tilde{\tau}_{1-i},\tilde{s}_{1-i},\tilde{v}_i) = f^1_{1-i}(\tau_{1-i},s_{1-i},v_i)$.
   - **Output**: $(\tilde{\tau}_{1-i},\tilde{s}_{1-i}||\bar{h}(\tilde{\tau}_i,\tilde{s}_i))$.
2. (**Define message** s): set $s := s_{1-i}||\bar{h}(\tau_i||s_i)$.

By the 1-more extractability property of $\mathcal{H}_k$ under leakage, we have

$$\Pr_{h\leftarrow\mathcal{H}_k}\left[\mathsf{Exp}^{s,h}_{\mathcal{A}_{v,i},\mathcal{A}_{s,1-i},\mathcal{E}_{i,1}}(1,\lambda,z_{v,i},z_{\mathcal{E}_{i,1}})=1\right]\leq \mathsf{negl}(k).$$

Let B be the event in which the extractor fails to produce a valid pre-image. Then, if B happens, and since we are not in a self-destruct round (an event denoted as E), we have that $\mathcal{A}_{v,i}$ produces a valid hash and $\mathcal{A}_{s,1-i}$, produces a valid pre-image, still the extractor fails, i.e., we have $\mathsf{Exp}^{s,h}_{\mathcal{A}_{v,i},\mathcal{A}_{s,1-i},\mathcal{E}_{i,1}}(1,\lambda,z_{v,i},z_{\mathcal{E}_{i,1}}) = 1$. Thus, by the above relation we receive $\Pr[B\wedge E]\leq\mathsf{negl}(k)$, and the extractor outputs a valid pre-image for $\tilde{v}_{1-i}$, i.e., $h(\hat{s}_{1-i}||\hat{v}_i;\hat{\tau}_{1-i}) = \tilde{v}_{1-i}$. Since the attacker creates a valid codeword, we also have $h(\tilde{s}_{1-i}||\bar{h}(\tilde{\tau}_i||\tilde{s}_i);\tilde{\tau}_{1-i}) = \tilde{v}_{1-i}$. We prove that the extracted values are consistent with the ones produced by the attacker in $\mathsf{Tamper}^{\mathsf{cnmlr}}_{\mathcal{A},m_b}(k)$, i.e., we prove that $(\hat{\tau}_{1-i},\hat{s}_{1-i}||\hat{v}_i) = (\tilde{\tau}_{1-i},\tilde{s}_{1-i}||\bar{h}(\tilde{\tau}_i||\tilde{s}_i))$, with overwhelming probability.

Let B' be the event in which $(\hat{\tau}_{1-i},\hat{s}_{1-i}||\hat{v}_i)\neq(\tilde{\tau}_{1-i},\tilde{s}_{1-i}||\bar{h}(\tilde{\tau}_i||\tilde{s}_i))$. Then, assuming there exist m, $\mathcal{A}$, $\mathcal{A}'$, for which $\Pr[\neg B\wedge E\wedge B']>\epsilon''$, for $\epsilon''=1/\mathrm{poly}(k)$, we build an attacker $\mathcal{A}''$ that breaks the collision resistance property of $\mathcal{H}_k$, with non-negligible probability: given m, $\mathcal{A}''$ simulates $\mathsf{Leak}^0_{\mathcal{A}',s}(k)$ while having full access to $(s_0,s_1)\leftarrow\mathsf{LRS}_{\mathsf{enc}}(m)$, and outputs $(\hat{\tau}_{1-i},\hat{s}_{1-i}||\hat{v}_i)$, $(\tilde{\tau}_{1-i},\tilde{s}_{1-i}||\bar{h}(\tilde{\tau}_i||\tilde{s}_i))$. We conclude that $\mathbf{t}_i[1]\approx_c\mathbf{t}_{\mathsf{Real}}[1]$. The case of $\mathbf{t}_{1-i}[1]$ is symmetric.

Inductive step:

Inductive hypothesis: *For* $i\in\{0,1\}$, $\mathbf{lk}[1:n]\approx_c\mathbf{lk}_{\mathsf{Real}}[1:n]$, $\mathbf{t}_i[1:n]\approx_c\mathbf{t}_{\mathsf{Real}}[1:n]$. *In particular, for* $j\in[n]$, $i\in\{0,1\}$, *there exist* $\mathcal{E}_{i,j}$ *with auxiliary inputs* $z_{\mathcal{E}_{i,j}}$, *that output valid pre-images in rounds* $1,\ldots,n$.

Proving that $\mathbf{lk}[n+1]\approx_c\mathbf{lk}_{\mathsf{Real}}[n+1]$ is straightforward, as the tampering and leakage queries are simulated correctly in the previous rounds: by leaking the first coordinate

[5] $\mathcal{A}_s$ knows (τ_i,s_i), $(\tilde{\tau}_i,\tilde{s}_i)$, since they are stored in st.

of $\mathsf{TComp}_q(i, \tau_i, s_i, v_{1-i}, \mathbf{lk}, n+1)$, for $i \in \{0,1\}$, it is clear that $\mathcal{A}'$ computes the replies to (g_0^{n+1}, g_1^{n+1}) as in the real execution, as previous tampering and leakage queries are simulated correctly. Thus, $\mathbf{lk}[n+1] \approx_c \mathbf{lk}_{\mathsf{Real}}[n+1]$. Regarding, the replies to the tampering queries the proof for the two cases, "$\exists i : \tilde{c}_i = c_i$" and "$\forall i : \tilde{c}_i \neq c_i \wedge \exists j : \tilde{v}_j = v_j$" are identical to the base case. For the last case "$\forall i : \tilde{v}_i \neq v_i$", the proof slightly changes, as the extractors $\mathcal{E}_{i,n+1}$ for the round $n+1$ depend on all previous extractors, and $\mathcal{A}$. As in the base case, we define $\mathcal{A}_{v,i}$, $z_{v,i}$ and $\mathcal{A}_{s,1-i}$.

1. **Program** $\mathcal{A}_{v,i}^{\mathcal{O}^\lambda(\tau_{1-i},\cdot)}(h, v_{1-i}, z_{v,i})$:
   Parse $z_{v,i}$ as (τ_i, s_i, s_{1-i}), query $\mathcal{O}^\lambda(\tau_{1-i}, \cdot)$ with $g_{s_{1-i}}(x) := \bar{h}(x||s_{1-i})$ and let $\bar{v}_{1-i}$ be the answer. Compute $v_i \leftarrow h(s_i||\bar{v}_{1-i}; \tau_i)$.
   For $j = 1, \ldots, n+1$:

   - Sample $(g_0^j, g_1^j) \leftarrow \mathcal{A}(h)$.
   - Define $g_{v_i,s_{1-i}}(\tau_{1-i}) := g_{1-i}^j(\tau_{1-i}, s_{1-i}, v_i)$, send $g_{v_i,s_{1-i}}$ to $\mathcal{O}^\lambda(\tau_{1-i}, \cdot)$ and let w_{1-i} be the answer.
   - Compute $w_i \leftarrow g_i^j(\tau_i, s_i, v_{1-i})$.
   - Send (w_0, w_1) to $\mathcal{A}$ and receive (f_0^j, f_1^j).
   - Compute $(\tilde{\tau}_i, \tilde{s}_i, \tilde{v}_{1-i}) \leftarrow f_i^j(\tau_i, s_i, v_{1-i})$ and set $\tilde{c}_i := (\tilde{\tau}_i, \tilde{s}_i, \tilde{v}_{1-i})$.
   - If $j \leq n$:
     - Sample $(\hat{\tau}_{1-i}, \hat{s}_{1-i}, \hat{v}_i) \leftarrow \mathcal{E}_{i,j}(h, v_{1-i}, z_{\mathcal{E}_{i,j}})$.
     - Compute $v_i \leftarrow h\left(\tilde{s}_i||\bar{h}(\hat{\tau}_{1-i}||\hat{s}_{1-i}); \tilde{\tau}_i\right)$ and set $\tilde{c}_{1-i} := (\hat{r}_{1-i}, \hat{s}_{1-i}, v_i)$.
     - Send $(\tilde{c}_0, \tilde{c}_1)$ to $\mathcal{A}$.

   Output: $(\tilde{v}_{1-i}, st)$, where $st = (\tilde{\tau}_i, \tilde{s}_i, \tilde{v}_{1-i}, \tau_i, s_i, v_{1-i})$.

2. (**Auxiliary input for** $\mathcal{A}_{v,i}$): set $z_{v,i} = (\tau_i, s_i, s_{1-i})$.

3. (**Existence of the extractor,** $\mathcal{E}_{i,n+1}$**, and auxiliary input,** $z_{\mathcal{E}_{i,n+1}}$): Given $\mathcal{A}_{v,i}$ and $z_{v,i}$, by the 1-more extractability property of $\mathcal{H}_k$ under leakage, there exists an extractor $\mathcal{E}_{i,n+1}$ for $\mathcal{A}_{v,i}$, with auxiliary input, $z_{\mathcal{E}_{i,n+1}}$, that computes $(\hat{\tau}_{1-i}, \hat{s}_{1-i}, \hat{v}_i) \leftarrow \mathcal{E}_{i,n+1}(h, v_{1-i}, z_{\mathcal{E}_{i,n+1}})$.

The definition of $\mathcal{A}_{s,1-i}$, s, follows.

1. **Program** $\mathcal{A}_{s,1-i}(h, \tau, s, st)$:

- Parse s as $s_{1-i}||\bar{v}_i$, and set $\tau_{1-i} := \tau$.
- Compute $v_i \leftarrow h(s_i||\bar{h}(\tau_{i-1}||s_{i-1});\tau_i)$.[6]
- Compute $(\tilde{\tau}_{1-i}, \tilde{s}_{1-i}, \tilde{v}_i) \leftarrow f_{1-i}^{n+1}(\tau_{1-i}, s_{1-i}, v_i)$.
- **Output**: $(\tilde{\tau}_{1-i}, \tilde{s}_{1-i}||\bar{h}(\tilde{\tau}_i, \tilde{s}_i))$.

2. (**Define message** s): set $s := s_{1-i}||\bar{h}(\tau_i||s_i)$.

By the 1-more extractability property of $\mathcal{H}_k$ under leakage, $\mathcal{E}_{i,n+1}$ outputs a valid preimage with overwhelming probability and using the same arguments that we used for the base case, we prove that the extracted value is consistent with $\mathbf{t}_{\mathsf{Real}}[n+1]$. This concludes the proof of the claim and implies the correctness of the values computed by $\mathcal{A}'$ in step 3, $\mathbf{lk}$, $\mathbf{t}_0$, $\mathbf{t}_1$, up to round $j^* - 1$, conditioned on $j_{\mathsf{d}} = j^*$. □

Claim 6.4.8. *Conditioned on $j_{\mathsf{d}} = j^*$, for any message m and all sufficiently large k, $\mathbf{t}_0[j_{\mathsf{d}}] \neq \mathbf{t}_1[j_{\mathsf{d}}]$, with overwhelming probability over the randomness of $\mathsf{Leak}^0_{\mathcal{A}',m}(k)$.*

Proof. By Claim 6.4.7, we have that the output of the decoder in $\mathsf{Tamper}^{\mathsf{cnmlr}}_{\mathcal{A},m_b}(k)$, is simulated perfectly for the first $j_{\mathsf{d}} - 1$ rounds, by the execution of TComp inside the leakage oracles. Thus, the tampering query made by the attacker in round j_{d} inside the oracles is consistent with the j_{d}-th tampering query made by the attacker in the execution of $\mathsf{Tamper}^{\mathsf{cnmlr}}_{\mathcal{A},m_b}(k)$. By assumption, j_{d} is self-destruct round, thus there exists $i \in \{0,1\}$, for which $h(\tilde{s}_i||\bar{h}(\tilde{\tau}_{1-i}||\tilde{s}_{1-i});\tilde{\tau}_i) \neq \tilde{v}_i$. We denote such an event by E_i and analyze the execution of TComp under the event $E_0 \vee E_1$. We consider the following cases.

- $\underline{\forall i\ \tilde{v}_i = v_i}$: Since $E_0 \vee E_1$ has occurred we know that for some $i \in \{0,1\}$, $(\tilde{\tau}_i, \tilde{s}_i) \neq (\tau_i, s_i)$, and by the definition of TComp_q, $\mathbf{t}_i[j_{\mathsf{d}}] = \perp_i \neq \mathbf{t}_{1-i}[j_{\mathsf{d}}]$, independently of the value in $\mathbf{t}_{1-i}[j_{\mathsf{d}}]$.
- $\underline{\exists i : \tilde{v}_i \neq v_i \wedge \tilde{v}_{1-i} = v_{1-i}}$: If $(\tau_i, s_i) = (\tilde{\tau}_i, \tilde{s}_i)$, we have that $\mathbf{t}_i[j_{\mathsf{d}}] = \mathsf{same}^* \neq \mathbf{t}_{1-i}[j_{\mathsf{d}}]$, since $\tilde{c}_{1-i} \neq c_{1-i}$, else if $(\tau_i, s_i) \neq (\tilde{\tau}_i, \tilde{s}_i)$, we have that $\mathbf{t}_i[j_{\mathsf{d}}] = \perp_i \neq \mathbf{t}_{1-i}[j_{\mathsf{d}}]$, independently of the value in $\mathbf{t}_{1-i}[j_{\mathsf{d}}]$.
- $\underline{\forall i\ \tilde{v}_i \neq v_i}$: We prove the needed for the non-trivial case in which for all $i \in \{0,1\}$, $\mathbf{t}_i[j_{\mathsf{d}}] \neq \perp_i$. Assuming the extractors executed inside the oracles output valid preimages, we have $\mathbf{t}_0[j_{\mathsf{d}}] = ((\tilde{\tau}_0, \tilde{s}_0, \tilde{v}_1), (\hat{\tau}_1, \hat{s}_1, v_0))$, $\mathbf{t}_1[j_{\mathsf{d}}] = ((\hat{\tau}_0, \hat{s}_0, v_1), (\tilde{\tau}_1, \tilde{s}_1, \tilde{v}_0))$. Conditioned on E_i, we have $h(\tilde{s}_i||\bar{h}(\tilde{\tau}_{1-i}, \tilde{s}_{1-i});\tilde{\tau}_i) \neq \tilde{v}_i = h(\hat{s}_i||\bar{h}(\tilde{\tau}_{1-i}, \tilde{s}_{1-i});\hat{\tau}_i)$, which implies that $(\hat{\tau}_i, \hat{s}_i) \neq (\tilde{\tau}_i, \tilde{s}_i)$. Thus $\mathbf{t}_0[j_{\mathsf{d}}] \neq \mathbf{t}_1[j_{\mathsf{d}}]$, under $E_0 \vee E_1$. □

[6] $\mathcal{A}_s$ knows (τ_i, s_i), $(\tilde{\tau}_i, \tilde{s}_i)$, since they are stored in st.

Claim 6.4.9. *For any message m and all sufficiently large k, $j^* = j_{\mathsf{d}}$ iff $d = 1$, with overwhelming probability over the randomness of $\mathsf{Leak}^0_{\mathcal{A}',m}(k)$.*

Proof. By Claims 6.4.7, 6.4.8, we have that for all $j \in [j_{\mathsf{d}} - 1]$, $\mathbf{t}_0[j] = \mathbf{t}_1[j]$ and $\mathbf{t}_0[j_{\mathsf{d}}] \neq \mathbf{t}_1[j_{\mathsf{d}}]$, and clearly, assuming $j^* = j_{\mathsf{d}}$, we have that $\hat{h}(\mathbf{t}_0[1 : j^* - 1]) = \hat{h}(\mathbf{t}_1[1 : j^* - 1])$ and $\hat{h}(\mathbf{t}_0[j^*]) \neq \hat{h}(\mathbf{t}_1[j^*])$, with overwhelming probability, otherwise we can break the collision resistance property of $\hat{h}$ by simulating $\mathsf{Leak}^0_{\mathcal{A}',m}(k)$. Thus $d = 1$. Symmetrically, assuming $d = 1$ we know that $\mathbf{t}_0[j^*] \neq \mathbf{t}_1[j^*]$, and using the collision resistance property of $\hat{h}$, with overwhelming probability, $\mathbf{t}_0[1 : j^* - 1] = \mathbf{t}_1[1 : j^* - 1]$. Thus, $d = 1$ iff $\mathcal{A}'$ makes a correct guess on j_{d}. □

Claim 6.4.10. *Conditioned on $j^* = j_{\mathsf{d}}$, out $\approx_c \mathsf{Tamper}^{\mathsf{cnmlr}}_{\mathcal{A},m_b}(k)$, over the randomness of $\mathsf{Leak}^0_{\mathcal{A}',m}(k)$.*

The main arguments that prove the current claim, have already been proved in Claim 6.4.7, as the execution in of $\mathcal{A}'$ in Step 4, is similar to the TComp.

Claim 6.4.11. *For any pair of messages m_0, m_1,* PPT *adversary $\mathcal{A}$, all sufficiently large k, and all* PPT *distinguishers D, assuming that*

$$\left|\Pr\left[\mathsf{D}\left(\mathsf{Tamper}^{\mathsf{cnmlr}}_{\mathcal{A},m_0}(k)\right) = 1\right] - \Pr\left[\mathsf{D}\left(\mathsf{Tamper}^{\mathsf{cnmlr}}_{\mathcal{A},m_1}(k)\right) = 1\right]\right| > \epsilon,$$

for $\epsilon = 1/\mathrm{poly}(k)$, we have

$$\left|\Pr\left[\mathsf{D}'\left(\mathsf{Leak}^0_{\mathcal{A}',m_0}(k)\right) = 1\right] - \Pr\left[\mathsf{D}'\left(\mathsf{Leak}^0_{\mathcal{A}',m_1}(k)\right) = 1\right]\right| > \epsilon/q - \mathsf{negl}(k),$$

where D', $\mathcal{A}'$, have already been defined above with respect to D, $\mathcal{A}$, respectively.

Proof. Let $L_i := \mathsf{Leak}^0_{\mathcal{A}',m_i}(k)$ and $T_i := \mathsf{Tamper}^{\mathsf{cnmlr}}_{\mathcal{A},m_i}(k)$. For $i \in \{0, 1\}$ we have,

$$\begin{aligned}\Pr\left[\mathsf{D}'(L_i) = 1 | j^* = j_{\mathsf{d}}\right] &\geq \Pr\left[\mathsf{D}'(L_i) = 1 | j^* = j_{\mathsf{d}}, d = 1\right] \cdot \Pr[d = 1 | j^* = j_{\mathsf{d}}] \\ &\geq \Pr\left[\mathsf{D}(T_i) = 1\right] - \mathsf{negl}(k).\end{aligned}$$

The last inequality follows from Claims 6.4.9, 6.4.10 and the definition of D'. Symmetrically,

$$\begin{aligned}\Pr\left[\mathsf{D}'(L_i) = 1 | j^* \neq j_{\mathsf{d}}\right] &\geq \Pr\left[\mathsf{D}'(L_i) = 1 | j^* \neq j_{\mathsf{d}}, d \neq 1\right] \cdot \Pr[d \neq 1 | j^* \neq j_{\mathsf{d}}] \\ &\geq \frac{1}{2} - \mathsf{negl}(k). \text{ (Claim 6.4.9, } \mathsf{D}')\end{aligned}$$

and from the above relations we receive

$$\begin{aligned}\Pr\left[\mathsf{D}'(L_i)=1\right] &= \Pr\left[\mathsf{D}'(L_i)=1|j^*=j_{\mathsf{d}}\right]\cdot\Pr[j^*=j_{\mathsf{d}}]\\ &+ \Pr\left[\mathsf{D}'(L_i)=1|j^*\neq j_{\mathsf{d}}\right]\cdot\Pr[j^*\neq j_{\mathsf{d}}]\\ &\geq \Pr\left[\mathsf{D}(T_i)=1\right]\cdot\frac{1}{q}+\frac{q-1}{2q}-\mathsf{negl}(k). \end{aligned} \tag{6.1}$$

Similarly, we obtain an upper bound on $\Pr\left[\mathsf{D}'(L_i)=1\right]$,

$$\Pr\left[\mathsf{D}'(L_i)=1\right]\leq\Pr\left[\mathsf{D}(T_i)=1\right]\cdot\frac{1}{q}+\frac{q-1}{2q}+\mathsf{negl}(k). \tag{6.2}$$

From 6.1, 6.2 we receive

$$\left|\Pr\left[\mathsf{D}'(L_i)=1\right]-\left(\frac{\Pr\left[\mathsf{D}(T_i)=1\right]}{q}+\frac{q-1}{2q}\right)\right|\leq\mathsf{negl}(k). \tag{6.3}$$

Let $\delta := |\Pr[\mathsf{D}'(L_0)=1]-\Pr[\mathsf{D}'(L_1)=1]|$. We compute,

$$\begin{aligned}\delta &= \left|\frac{\Pr\left[\mathsf{D}(T_0)=1\right]}{q}-\frac{\Pr\left[\mathsf{D}(T_1)=1\right]}{q}+\Pr[\mathsf{D}'(L_0)=1]-\Pr[\mathsf{D}'(L_1)=1]\right.\\ &- \left.\left(\frac{\Pr\left[\mathsf{D}(T_0)=1\right]}{q}+\frac{q-1}{2q}\right)+\left(\frac{\Pr\left[\mathsf{D}(T_1)=1\right]}{q}+\frac{q-1}{2q}\right)\right|\\ &\geq \epsilon/q-\left|\frac{\Pr\left[\mathsf{D}(T_0)=1\right]}{q}+\frac{q-1}{2q}-\Pr[\mathsf{D}'(L_0)=1]\right.\\ &+ \left.\Pr[\mathsf{D}'(L_1)=1]-\frac{\Pr\left[\mathsf{D}(T_1)=1\right]}{q}-\frac{q-1}{2q}\right|\geq\epsilon/q-\mathsf{negl}(k). \end{aligned} \tag{6.4}$$

The above inequalities follow from 6.3, the triangle inequality, and the assumption that D distinguishes between $\mathsf{Tamper}^{\mathsf{cnmlr}}_{\mathcal{A},m_0}(k)$ and $\mathsf{Tamper}^{\mathsf{cnmlr}}_{\mathcal{A},m_1}(k)$, with non-negligible probability ϵ. Assuming such an attacker $\mathcal{A}$, we proved that D' distinguishes between $\mathsf{Leak}^0_{\mathcal{A}',m_0}(k)$ and $\mathsf{Leak}^0_{\mathcal{A}',m_1}(k)$, with non-negligible probability $\epsilon/q-\mathsf{negl}(k)$, and the proof is complete. □

The above claim completes the proof of the theorem. □

6.4.1 Instantiations

It is not hard to see that, the ℓ-more wECRH of Theorem 4.4.1 satisfies definition 6.4.1, thus by plugging that construction to Theorem 6.4.5, we obtain a leakage-resilient continuous NMC for polynomially many rounds, as the extractor of Theorem 4.4.1 has running time linear in the running time of the adversary (cf. Section 4.4). In particular, we receive the following corollary.

Corollary 6.4.12. *Assuming random oracles and collision resistant hash function families, construction 6.4.3 is a* q-CNMLR *code against* λ *bits of leakage, for any* $q=\mathrm{poly}(k)$*, assuming* $\lambda'\geq 2\lambda+6k+1$.

It should be noted that, the proof of Theorem 6.4.5 is with respect to any ℓ-more wECRH. When considering the random oracle model, and similarly to [FHMV17], the random oracle can be simulated inside the leakage oracles using a pseudo-random function.

By plugging the leakage-resilient ℓ-more wECRH of Section 4.3.5,[7] against λ bits of leakage, to Theorem 6.4.5, we receive a continuous NMC for a number of rounds which is at least poly-logarithmic in the security parameter, and tolerating λ bits of non-adaptive leakage. In particular, we receive the following corollary.

Corollary 6.4.13. *For $k, t, \lambda' \in \mathbb{N}$, assuming* DLOG*, t-*KEA*, and collision resistant hash function families, Construction 6.4.3 is a q-*CNMC *against λ bits of non-adaptive leakage, for $\lambda' \geq 2\lambda + 8k + 1$ and any q that is at least poly-logarithmic in k. Here, λ' is the leakage parameter of the underlying* LRS *scheme.*

Our t-KEA based construction tolerates at least poly-logarithmic number of rounds due to the fact that, in the current proof, the extractor for round i depends on the extractor for round $i-1$ and extraction is non-black box, thus by considering poly-logarithmic number of rounds, we avoid the super-polynomial blow-up in the size of the final extractor. If the extractor's overhead is linear, then the construction can tolerate polynomially many rounds of tampering. In addition, our KEA based construction guarantees extractability only if the hash is indistinguishable from uniform, which is a property that cannot be achieved when the adversary is given access to the CRS and split-state leakage over both parts of the codeword. However, the weaker form of leakage resilience that we prove in Section 4.3.5, yields continuous non-malleable codes against non-adaptive leakage, in which the adversary can request leakage over the randomness only at the very beginning of the experiment.

[7] Here, leakage resilience is with respect to Definition 4.3.15, which is non-adaptive with respect to the CRS.

Chapter 7

Non-malleable commitments

7.1 Introduction

A commitment scheme is a cryptographic primitive that enables an entity to commit to a chosen value, while ensuring that: (i) the committed value remains private until the committer decides to reveal it (*hiding property*) (ii) the committer cannot decommit to value different than the one he committed to (*binding property*) [KL14]. Commitment schemes can be classified to *interactive* or *non-interactive*, and as far as security is concerned, they may provide *perfect*, *statistical* or *computational* security, with respect to the *hiding* and *binding* properties. For instance, a commitment scheme is said to be *statistically hiding*, if for any two messages s, s', the distribution of commitments over s, is statistically close to the distribution of commitments over s'.[1] Analogously, a commitment scheme is said to be *statistically binding*, if the probability that a computationally unbounded adversary manages to decommit to a different value, is negligible.

The notion of non-malleable commitments was introduced in the seminal work of Dolev, Dwork and Naor [DDN91], as a countermeasure against *man-in-the-middle* adversaries. In the man-in-the-middle setting, we consider two parties that wish to execute a protocol in the presence of an adversary, that fully controls the communication channel between the parties. The adversary is allowed to modify, block, or introduce messages, and also schedule the order of delivery, while the parties might not be aware of the adversarial presence. Protocols that remain secure against man-in-the-middle adversaries are said to be non-malleable [DDN91].

The capabilities of man-in-the-middle adversaries are strong, thus the task of designing

[1] The process of committing is randomized.

and analyzing non-malleable protocols is highly non-trivial. In the seminal paper [DDN91], Dolev, Dwork and Naor propose security definitions for the notions of non-malleable commitments and non-malleable zero-knowledge, and assuming the existence of one-way functions, they construct secure protocols that require $\log(k)$ rounds of interaction, where k denotes the security parameter. Informally, a commitment scheme is non-malleable, if any man-in-the-middle adversary that is given a commitment over a message v, is not able to create a valid commitment of a message $\tilde{v}$, which is related to v. This notion has been modeled in two ways:

- *Non-malleability with respect to commitment* [DDN91]: According to this notion, the adversary succeeds in breaking security, if he manages to commit to a related value, even without being able to produce a valid decommitment. This notion is meaningful only for *statistically-binding* commitments.

- *Non-malleability with respect to opening* [DIO98]: In this setting, the adversary breaks security if he manages to both commit and decommit to a related value. This notion is meaningful, both for the case of *statistically-binding* and *statistically-hiding* commitments.

In the present chapter, we construct *succinct,*[2] *non-interactive non-malleable commitments with respect to opening*, from ℓ-more wECRHs. Our result is summarized in the following informal theorem.

Theorem 7.1.1 (Informal)**.** *Assuming ℓ-more* wECRH*, there exists an explicit succinct non-interactive, non-malleable commitment scheme with respect to opening.*

Our primitive achieves a stronger definition of non-malleability, that allows the adversary's auxiliary input to depend on the message (this is not allowed in [DIO98]), and in contrast to [PR05], our simulator is weaker, in the sense that it does not need access to the original message in order to simulate the decommitment phase. Our KEA based instantiation produces commitments of size $2k$, while for the random oracle based construction the commitment size is k.

7.2 Related work

The notion of non-malleable commitments (NMCOM) was introduced in the seminal work of Dolev, Dwork and Naor [DDN98], as a countermeasure against man-in-the-middle

[2]The length of the commitment is independent of the message length.

(MIM) adversaries, and assuming one-way functions, the authors built NMCOM requiring at least logarithmic number of rounds of interaction and zero-knowledge proofs. In [DIO98, DKOS01, DG03], the authors construct non-interactive NMCOM, by either assuming that the adversary's auxiliary input does not depend on the message, or that the process of sampling a message that is consistent with the adversarial auxiliary input, is efficient. The work of [PR05], allows the adversarial auxiliary input to depend on the message, however the simulator requires access to the original message in order to simulate a valid decommitment. In [GPR16, LP11, CVZ10, LPS17, COSV17, COSV16] the authors construct interactive (concurrent) non-malleable commitments using various assumptions, while in [Pas13] Pass proves that non-interactive NMCOM *cannot be proved using a black-box reduction to standard assumptions.*

7.3 Non-malleable commitments from ℓ-more wECRH

We start by presenting the notion of non-interactive commitments in the CRS model.

Definition 7.3.1 (Non-interactive commitment in the CRS model)**.**
Let $(\mathsf{Init}, \mathsf{Commit}, \mathsf{Open})$ *be a commitment scheme and let* $\Sigma \leftarrow \mathsf{Init}(1^k)$*. Then, it satisfies the following properties:*

- **(Computational binding)***: It is computationally infeasible to find* $s \neq s'$*, and* τ*,* τ'*, such that* $\mathsf{Commit}(\Sigma, s; \tau) = \mathsf{Commit}(\Sigma, s'; \tau')$.
- **(Statistical hiding)***: For any two messages* s, s'*,* $\mathsf{Commit}(\Sigma, s) \approx \mathsf{Commit}(\Sigma, s')$.

For brevity, the CRS is omitted when calling Commit and Open.

We define the notion of non-malleable, non-interactive commitments, in the standalone setting, following the definition of Pass and Rosen [PR05], with some simplifications for the non-interactive settings. First, present the *man-in-the-middle execution* with respect to the *real game* and the *ideal experiment*, as follows.

Man-in-the-middle experiment (real game). Here we consider a two-stage man-in-the-middle adversary, $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, where $\mathcal{A}_1$ participates in the commitment stage and $\mathcal{A}_2$ participates in the opening stage. More specifically, given a binary relation $\mathcal{R}(\cdot, \cdot)$, a man-in-the-middle adversary, $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, a sender, Sender, a receiver, $\mathsf{Receiver}$, a message s, and auxiliary input z, we define the real experiment $\mathsf{Real}_{\mathcal{A}}(\mathcal{R}, s, z)$ as follows:

(1) Sender sends a commitment $c \leftarrow \mathsf{Commit}(s;\tau)$ to $\mathcal{A}$; (2) $\mathcal{A}_1(c,z)$ sends a commitment $\tilde{c}$ to Receiver; (3) Sender sends the opening of c to $\mathcal{A}$; (4) $\mathcal{A}_2(z,s,\tau)$ sends the opening of $\tilde{c}$ to Receiver. The experiment outputs 1 if and only if $\mathcal{A}$ produces a valid $\tilde{s}$ for the commitment $\tilde{c}$, and $\mathcal{R}(s,\tilde{s}) = 1$. The message s is chosen prior to the experiment and $\mathcal{A}$ is allowed to receive auxiliary input, z, that might depend on s.

The Ideal experiment. Given a binary relation $\mathcal{R}(\cdot,\cdot)$, an ideal adversary, $\mathcal{S}$, a sender, Sender and receiver Receiver, message s, and auxiliary input z, the Ideal experiment, $\mathsf{Ideal}_{\mathcal{S}}(\mathcal{R},s,z)$, is defined as follows: $\mathcal{S}$ only interacts with Receiver by (1) sending a commitment to $\tilde{c}$ it and (2) sending the corresponding decommitment. The experiment outputs 1 if and only if $\mathcal{S}$ produces a valid decommitment, $\tilde{s}$, for $\tilde{c}$ and $\mathcal{R}(s,\tilde{s}) = 1$. The message s is chosen prior to the experiment and $\mathcal{S}$ receives the auxiliary input z, as $\mathcal{A}$ does in the real experiment.

Having defined the real and ideal executions, we define the notion of non-malleable (non-interactive) commitments.

Definition 7.3.2 (Non-malleable non-interactive commitment)**.** *A non-interactive commitment scheme is said to be non-malleable (with respect to opening) if for every* PPT *man-in-the-middle adversary $\mathcal{A}$, there exists a* PPT *adversary $\mathcal{S}$ and a negligible function* $\mathsf{negl}(\cdot)$*, such that for every non-reflexive polynomial-time computable relation $\mathcal{R} \subseteq \{0,1\}^n \times \{0,1\}^n$, every $s \in \{0,1\}^n$ and every $z \in \{0,1\}^*$, we have that*

$$\Pr\left[\mathsf{Real}_{\mathcal{A}}(\mathcal{R},s,z) = 1\right] < \Pr\left[\mathsf{Ideal}_{\mathcal{S}}(\mathcal{R},s,z)\right] + \mathsf{negl}(n).$$

It should be noted that, the definition presented above is stronger from the ones presented in [DIO98, PR05], since (1) we allow the attacker's auxiliary input to depend on the message s, and (2), our simulator does not need the original message in order to simulate the decommitment phase.

In the CRS model, both the real and ideal experiments will generate the CRS by running Init and all parties will receive access to it.[3]

Below we define our construction.

Construction 7.3.3 (Non-malleable non-interactive commitment)**.** *Let $\mathcal{H}_k$ be a hash function family. We define a non-interactive commitment scheme* $(\mathsf{Init},\mathsf{Commit},\mathsf{Open})$*, as follows:*

[3]There is a weaker model called trapdoor CRS, in which the simulator $\mathcal{S}$ generates an indistinguishable CRS with a trapdoor. The construction that is proposed in the present thesis uses the honestly generated CRS, i.e., it does not require trapdoor information.

- $\mathsf{Init}(1^k)$*: Sample* $h \leftarrow \mathcal{H}_k$ *and set* $\Sigma := h$*.*
- $\mathsf{Commit}(\Sigma, \cdot)$*: on input string* $s \in \{0,1\}^{\mathrm{poly}(k)}$*, the algorithm selects random string* $\tau \in \{0,1\}^{\mathrm{poly}(k)}$ *and outputs* $h(s;\tau)$*.*
- $\mathsf{Open}(\Sigma, \cdot)$*: on input a commitment* c*, the algorithm outputs* s, τ*. The receiver accepts if* $h(s;\tau) = c$*.*

In the following statement we formalize the properties that $\mathcal{H}_k$ should meet, in order for the above scheme to be non-malleable.

Theorem 7.3.4. *Let* $\mathcal{H}_k$ *collision resistant hash function family, such that for* $h \leftarrow \mathcal{H}_k$ *and any message* s*,* $h(s)$ *is statistically close to uniform. Then the commitment scheme of Construction 7.3.3 is statistically hiding and computationally binding. Furthermore, if the hash function family* $\mathcal{H}_k$*, is a* 1*-more* wECRH*, then the commitment scheme is non-malleable with respect to opening (cf. Definition 7.3.2).*

Intuitively, if the hash function produces outputs that are indistinghuishable from uniform, then the commitment scheme achieves the hiding property. In addition, if it is collision resistant, then the scheme is also binding. Finally, if the hash function is a 1-more wECRH, then for any man-in-the-middle attacker that produces a commitment (hash value) $\tilde{c}$, given c, where $\tilde{c} \neq c$, there exists an extractor that extracts $(\hat{s}, \hat{\tau})$, such that $h(\hat{s};\hat{\tau}) = \tilde{c}$. Since c reveals no information about s, the extracted value, $\hat{s}$, is unrelated to s. Below we provide a formal proof of security, based on those ideas.

Proof. (of Theorem 7.3.4) The first part of the proof, i.e., proving the statistical hiding property of the scheme, is straightforward, as by assumption, the distribution of $h(s;\tau)$ is statistically close to uniform. In addition, since the hash function family is collision resistant, no PPT adversary can find two distinct valid openings for the same commitment, i.e., computing efficiently $(s,\tau) \neq (s',\tau')$, for which $h(s;\tau) = h(s';\tau')$, happens only with negligible probability, assuming the collision resistance property of $\mathcal{H}_k$. The binding property of the scheme follows.

Next we are going to prove non-malleability. Given any man-in-the-middle adversary $\mathcal{A}$, we define an ideal adversary $\mathcal{S}$ as follows: $\mathcal{S}$, (1) samples $\tau \leftarrow \{0,1\}^{\mathrm{poly}(k)}$ and sends $c := h(0;\tau)$ to $\mathcal{A}$, (2) then executes $\tilde{c} \leftarrow \mathcal{A}_1(h, c, z)$ and forwards $\tilde{c}$ to the external receiver $\mathsf{Receiver}$, and (3) for the opening, if $\tilde{c} = c$, then $\mathcal{S}$ just sends $(0,\tau)$. Otherwise, $\mathcal{S}$ runs the extractor $\mathcal{E}$ (defined below) to extract $(\tilde{s},\tilde{\tau})$ and forwards the extracted value to $\mathsf{Receiver}$.

Below, we relate the above execution, with the execution of the ℓ-more ECRH experiment, of Definition 4.2.1.
We first define $\mathcal{A}_v$:

> $\mathcal{A}_v$ on input the description of a hash function h, a hash value $c = h(s;\tau)$, and the auxiliary input z, outputs $\tilde{c} \leftarrow \mathcal{A}_1(h, c, z)$. By the properties of the 1-more wECRH (cf. Definition 4.3.15), there exist auxiliary input z' and ex tractor $\mathcal{E}$ that on input z' and c outputs $(\hat{s}, \hat{\tau})$.

We will prove that for any $\mathcal{A}$,

$$\Pr\left[\mathsf{Real}_{\mathcal{A}}(\mathcal{R}, s, z) = 1\right] < \Pr\left[\mathsf{Ideal}_{\mathcal{S}}(\mathcal{R}, s, z)\right] + \mathsf{negl}(k),$$

using a hybrid argument presented below.

H_1: this hybrid is the same as the Real execution, for the first two steps. In the third step, the sender Sender does not provide an opening for the commitment. Instead, if the man-in-the middle adversary $\mathcal{A}$ forwards the commitment sent by Sender, i.e., $\tilde{c} = c$ the experiment just outputs $\mathcal{R}(s, s)$. Otherwise, the experiment runs the extractor $\mathcal{E}$ (defined above) to extract a value $(\hat{s}, \hat{\tau})$, and sends $(\hat{s}, \hat{\tau})$ to Receiver. The experiment finally outputs $\mathcal{R}(s, \hat{s})$.

H_2: this hybrid is the same as H_1 except that Sender commits to zero in the first step.

Claim 7.3.5. *Assuming that $\mathcal{H}_k$ is a 1-more* wECRH*, then for any non-reflexible polynomially computable relation $\mathcal{R}$ and any s, z we have*

$$\Pr[\mathsf{Real}_{\mathcal{A}}(\mathcal{R}, s, z) = 1] < \Pr[H_1 = 1] + \mathsf{negl}(k).$$

Proof. Let $\mathcal{A}_v$ and $\mathcal{E}$ be the adversary and extractor as defined above, and let $(\hat{s}, \hat{\tau})$ be the extracted value in H_1. We further define $\mathcal{A}_s$ as follows: on input (h, τ, s, z), $\mathcal{A}_s$ outputs $(\tilde{s}, \tilde{\tau}) \leftarrow \mathcal{A}_2(h, \tau, s, z)$. We define the following events:

- E_1: $h(\tilde{s};\tilde{\tau}) = h(\hat{s};\hat{\tau}) \ \wedge \ (\tilde{s};\tilde{\tau}) \neq (\hat{s};\hat{\tau})$.
- E_2: $h(\tilde{s};\tilde{\tau}) = \tilde{c} \ \wedge \ \tilde{c} \neq c \ \wedge \ h(\hat{s};\hat{\tau}) \neq c$.

Since $\mathcal{H}_k$ is collision resistant, $\Pr[E_1] = \mathsf{negl}(k)$. Otherwise, one can find a collision with non-negligible probability by simulating H_1 with $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ and $\mathcal{E}$. By the 1-more extractability property of $\mathcal{H}_k$, we have that $\Pr[E_2] = \mathsf{negl}(k)$.

Finally, we observe that Real outputs 1 and $H_1 = 0$ only when the extractor $\mathcal{E}$ fails to extract a valid decommitment, or it extracts a value such that $(\hat{s}, \hat{\tau}) \neq (\tilde{s}, \tilde{\tau})$. This is captured by the events E_1, E_2. Therefore, conditioned on $\neg(E_1 \vee E_2)$, $\mathsf{Real}_{\mathcal{A}}(\mathcal{R}, s, z) = 1$ implies that $H_1 = 1$. We can then conclude:

$$\Pr[\mathsf{Real}_{\mathcal{A}}(\mathcal{R}, s, z) = 1] < \Pr[H_1 = 1] + \Pr[\neg(E_1 \vee E_2)] = \Pr[H_1 = 1] + \mathsf{negl}(k).$$

This completes the proof of the claim. □

Claim 7.3.6. *Assuming the hiding property of the commitment scheme, we have*

$$|\Pr[H_1 = 1] - \Pr[H_2 = 1]| < \mathsf{negl}(k).$$

Proof. We observe that both H_1 and H_2 do not depend on the opening of the commitment. Therefore, if one can distinguish H_1 from H_2, then it can distinguish between $\mathsf{Commit}(0)$ from $\mathsf{Commit}(s)$. □

It is clear that $\Pr[\mathsf{Ideal}_{\mathcal{S}}(\mathcal{R}, s, z) = 1] = \Pr[H_2 = 1]$, as $\mathcal{S}$ simulates perfectly the experiment H_2. By the above claims, we receive

$$\begin{aligned}\Pr\left[\mathsf{Real}_{\mathcal{A}}(\mathcal{R}, s, z) = 1\right] &< \Pr[H_1 = 1] + \mathsf{negl}(k) \\ &\leq \Pr[H_2 = 1] + \mathsf{negl}(k) \\ &= \Pr\left[\mathsf{Ideal}_{\mathcal{S}}(\mathcal{R}, s, z)\right] + \mathsf{negl}(k).\end{aligned}$$

This concludes the proof of the theorem. □

Instantiations. Construction 7.3.3 can be instantiated using the 1-more (ℓ-more) wECRH of Construction 4.3.2, since it produces hashes that are indistinguishable from uniform (see Claim 4.3.17 in the proof of Theorem 4.3.16). It can also be instantiated in the random oracle model, as the uniformity property of the random oracle based ℓ-more wECRH (cf. Theorem 4.4.1) is straightforward.

Chapter 8

Secure circuit outsourcing

8.1 Introduction

The fabrication process of integrated circuits (ICs) adopted by the semiconductor industry is fundamentally global, involving several parties that may not be trusted. As a result, integrated circuits (ICs) are susceptible to the so-called hardware Trojans, which are hardware components maliciously implanted to the original circuitry, having as a purpose to alter its functionality, while remaining undetected. Hardware Trojans are aiming at disabling or compromising the security defences of a system, or covertly leaking information related to the systems' private state [LKG+09, CNB09, BRPB14]. Their implantation may occur during the design phase, by a malicious designer, or during the manufacturing phase, by a malicious fabrication facility. Once the Trojan is implanted, it may be active the entire time, or it may be triggered by some special event, e.g., when the user supplies the circuit with some special input, or after a specific number of circuit invocations.

Reliable detection of compromised circuits via testing and reverse engineering techniques, seems to be an impossible task [BR15], as in practice, all non-destructive testing techniques can easily be circumvented by properly obfuscating the implanted Trojans. The U.S. military recognized this threat and started two programs, namely, Trust and IRIS, with the intent of developing techniques and metrics for certifying ICs that are designated for weapon systems. The main concern behind this decision, is that non-certified advanced weapon systems can be potential Trojan carriers, thus while they may appear to function property when tested, they could be deactivated during combat, after being triggered by some specific event. Furthermore, even if they remain partially functional, there is no way to verify that they are not programmed to leak sensitive information [Sha07].

Inspired by the above considerations, in this chapter we put forward a formal security model for the problem of utilizing off-shore fabrication facilities, for IC manufacturing.

8.1.1 Contributions

The present chapter proposes a formal framework for assessing security of circuits, whose production has been partially outsourced to a set of untrusted, and possibly adversarial, manufacturers. Our security definition ensures that, any black-box execution of the produced circuit in the wild, leaks no information over its private state, even if the adversary has modified the outsourced components, arbitrarily. The only requirement is that during the execution, the circuits communicate with the user/adversary, through its input and output gates. In the following paragraphs, we briefly summarize the contributions of the present chapter.

Secure circuit fabrication. Let Γ be any circuit that needs to be produced. We propose a compiler that, given the description of Γ, returns the description of a circuit $\widehat{\Gamma}$, together with some auxiliary information, specifying (i) how $\widehat{\Gamma}$ can be divided into sub-components whose production can be outsourced to a set of possibly malicious facilities, and (ii) how the circuit designer should combine the outsourced components with the ones built in-house, so as to assemble $\widehat{\Gamma}$. After assembling $\widehat{\Gamma}$, the circuit's private state is initialized with some private value M_1, and the circuit is ready to be used in the wild.

In order for the above approach to make sense, certain requirements need to be satisfied. First of all, our compiler needs to be functionality preserving, meaning that the compiled circuit, $\widehat{\Gamma}$, should compute the same functionality with the original circuit, Γ, for all possible initial memories M_1, and for all possible inputs.[1] Secondly, our compiler should be secure under plausible assumptions over the set manufacturers that construct the outsourced components, ensuring that no information over the private state is leaked, when $\widehat{\Gamma}$ is used in the wild.

Our security definition is *simulation-based*, and is inspired by similar definitions in the setting of *tamper-proof circuit compilers* [IPSW06, FPV11, DK12, DK14, KT13]. In a nutshell, our security definition requires that, whatever the adversary can learn by interacting with the fabricated circuit, $\widehat{\Gamma}$, it can be simulated given only black-box access to the original circuit, Γ. This implies that, even if the outsourced components have been

[1] For randomized functionalities, we require the outputs of the circuits to be statistically close.

maliciously modified, e.g., by implanting a hardware Trojan, using $\widehat{\Gamma}$ is as secure as using the original circuit Γ, and thus, no information over the private state is leaked.

We also consider a weaker version of the above definition, in which the simulator is allowed to receive a short advice (*leakage*) over the circuit's private state, M_1. This models the setting in which the adversary might be able to learn a short amount of information over the secret memory, and provides security whenever the original circuit is resilient in the presence of leakage. An appealing advantage of the weaker definition is that, it might allow for more efficient circuit compilers.

A compiler based on MPC. In Section 8.3, we show how to construct secure outsourcing compilers, for arbitrary circuits, in the setting where $m \geq 2$ outsourcing manufacturers are available, and a certain unknown subset of them is malicious. Our construction utilizes a general client-server secure multiparty computation (MPC) protocol, i.e., a protocol that, for any functionality, enables a set of clients to privately communicate their inputs to a set of servers that will perform a computation and return the output to a single designated recipient. We stress that many MPC protocols follow this paradigm (e.g., [DI05]), while others, as we comment later, can be easily adapted to it.

Given such a protocol, the compiler operates in the following way. For a given circuit Γ it produces the MPC protocol implementing it, isolates the client and recipient computation for manufacturing in-house, and outsources each of the other components (representing a server in the MPC protocol) to the untrusted manufacturers. The key points of this compiler construction are as follows: (i) The client and recipient computation are typically quite lightweight; the client, in many protocols, simply performs an encryption or a secret-sharing operation, and the recipient a secret-reconstruction protocol; in either case, the computation is independent of the circuit that is outsourced, and (ii) there are MPC protocols that can tolerate up to $m - 1$ malicious servers, something we can leverage to argue that, if at least one of the outsourcing manufacturers is honest, the compiled circuit would be safe for use.

Additional properties of the underlying MPC protocol can also be very valuable by our compiler: for instance, if the underlying MPC protocol supports guaranteed output delivery, we can use this guarantee to argue that the final circuit will be resilient to a certain faulty outsourced sub-component. Moreover, if the underlying protocol satisfies the identifiable abort property, cf. [IOZ14], we can enable our compiled circuit to switch-off an outsourced sub-component that is discovered to be faulty (or malicious), thus reducing energy consumption.

8.1.2 Related work

The work of Seifert and Bayer [SB15] proposes a security model for the fabrication of Trojan-resilient circuits, requiring that the final circuit always produces the same output as the original one. In this setting, the authors present a secure construction, only for very limited classes of Trojans, that are allowed to "corrupt", only a small fraction of the gates in each layer of the IC and a small fraction of the wires connecting different layers.

Recently, Wahby *et al.* [WHG+16] introduced a new approach to the problem of defeating hardware Trojans in fabless circuit manufacturing. Their model reflects the fact that the IC specification and design are trusted, but the fabrication facility is not. Rather than testing or reverse engineering the IC hardware, which only provides limited security, they consider a class of solutions where the IC's operations are continuously verified. Such an approach makes sense as long as the verification circuitry: (i) is not costly to construct, and (ii) it is efficient. These properties are achieved by leveraging a *verifiable computation* (VC) scheme for the function implemented by the original circuit. Verifiable computation (see, e.g., [GGP10]) is a recent paradigm in which resource-constrained clients can delegate the computation of some function, $\mathcal{F}$, on possibly private input X, to an untrusted and computationally powerful server, without the server being able to lie about the outcome of the computation, and with the property that verifying the server's answer is much more efficient than computing the function from scratch. In a nutshell, the goal of [WHG+16] is to ensure that the output of the produced circuit is either invalid, or equal to the output of the original circuit. The main drawback in this setting is that, invalid outputs might be arbitrarily correlated, and thus leak, part of the circuit's private state.

In [DFS16], the authors show how to protect against hardware Trojans using testing-based mechanisms. Their work is based on two existing techniques for Trojan detection, called "input scrambling" and "split manufacturing" [IEGT13], for which the authors provide formal models. In this setting, they present a generic compiler that transforms any circuit into a functionally equivalent one, that satisfies the following: Assuming the attacker invokes the circuit $q \in \mathbb{N}$ times, and that the device is being tested t times, for $t > q$ uniformly random over on a specific range which is not known to the adversary, the compiled circuit is secure with probability at least $1 - (q/t)^{\ell/2}$, were ℓ is the number of sub-circuits whose production is outsourced. The assumption of [DFS16] is an a-priori known bound on the number q of interactions between the adversary and the device; in fact, without such a bound, their construction would require a super-polynomial number of tests. Unfortunately, in many important applications, it is not realistic to assume an upper

bound on the value q, and thus it is an important open problem to design a methodology that provides security for an arbitrary polynomial number of interactions between the adversary and the device.

The approach of applying secure distributed computing to defeat hardware Trojans has also been recently explored by [MCS+17]. However, this work is more focused on the implementation aspects of this idea, and moreover it assumes that the possibly malicious circuit components run applications that are developed and signed by a trusted software developer.

Prevention of hardware Trojans in ICs might take place during the design, manufacturing, and post-manufacturing stage [Pot10, LJM11]. However, since it is not always possible to prevent Trojan *insertion*, Trojan *detection* has also been vastly explored [BR15]. Common methodologies used to perform Trojan detection vary from invasive ones, that destroy the IC to examine its inner parts, to non-invasive ones, in which the circuit is executed and its output is compared to the output of a trusted copy, or against some already known output values. Trojan detection is typically an expensive and unreliable process for circuit protection, explicit *countermeasures* have also been proposed. For instance, the so-called "data guards" are designed to prevent a Trojan from being activated and/or to access sensitive data [WS11], while in [MWPB09, WS11] the authors propose the duplication of logic elements and the division of the sensitive data to independent parts of the circuit.

Our security definition shares similarities with analogous definitions in the context of protecting circuit implementations against tampering attacks, which received considerable attention in the past few years [IPSW06, FPV11, DK12, DK14, KT13]. The main difference between this setting and the one considered in this chapter, is that tamper-proof circuit compilers are typically used to protect against fault injection and tampering attacks, during run-time. Such attacks are usually carried out in an adaptive manner, depending on the outcome of previous attempts. Outsourcing compilers, instead, only protect against non-adaptive tampering, taking place during the circuit fabrication process. Importantly, the latter restriction allows to obtain security against arbitrary modifications, whereas in circuit tampering, one needs to consider a restricted class of adversaries, e.g., wire tampering [IPSW06] or gate tampering [KT13].

8.2 Definitions

In the current section, we put forward a formal model for assessing security of a (cryptographic) circuit, whose production is outsourced to one or more untrusted facilities. We

start by recalling the notion of a *Boolean circuit*.

8.2.1 Boolean Circuits

A Boolean circuit, is represented by a directed graph $\Gamma = (V, E)$, where the set of nodes, V, represents the set of logical gates, and the set of edges, E, represents wires that establish connections between gates. For the case of *deterministic* circuits, the gates can be of type `AND`, `XOR` and `copy`, where `AND` (resp. `XOR`) have fan-in two and fan-out one, and output the outcome of the AND (resp. XOR) operation over the input bits; a copy gate, which is denoted as `copy`, simply forwards the input bit into two output wires. The depth of a circuit is defined as the longest path from an input to an output gate and the size of a circuit is defined as its total number of gates. For simplicity, we will use Γ, to denote both a circuit and its description.

A circuit is clocked, if it executes in clock cycles (or rounds). The input and output values of a circuit 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. This randomness is produced by gates that we call *randomness gates* and we denote them as \$. In each clock cycle, a gate \$ outputs a fresh random bit.

Circuits may also possess memory gates, that maintain the circuit's state. Each memory gate has a single incoming edge (wire) and any number of outgoing edges. At any clock cycle, each memory gate sends its current state to the outgoing edges and updates it according to the value of the input edges. 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_1 denoting the initial state. When a circuit is executed with state M_i and input X_i, it produces output Y_i and the memory gates will reach the new state M_{i+1}. This process is denoted by $(Y_i, M_{i+1}) \leftarrow \Gamma[M_i](X_i)$.

Next, we introduce the notion of an *outsourcing circuit compiler* (or simply compiler). In a nutshell, a circuit compiler is an efficient algorithm, Φ, that takes as input the description of a circuit Γ, $\langle\Gamma\rangle$, and outputs the description of a compiled circuit $\widehat{\Gamma}$. Additionally, Φ returns a list of sub-components $\widehat{\Gamma}_i$, of $\widehat{\Gamma}$, whose production can be outsourced to one or more external manufacturers, together with the relevant information on how to connect those sub-components with the remaining ones (that need to be built in-house) in order to re-assemble the compiled circuit $\widehat{\Gamma}$.

Definition 8.2.1 (Outsourcing circuit compiler)**.** *Let Γ be an arbitrary circuit. A (ρ, m)-outsourcing compiler Φ is a* PPT *algorithm $(\widehat{\Gamma}, z) \leftarrow \Phi(\Gamma)$, such that the following holds:*

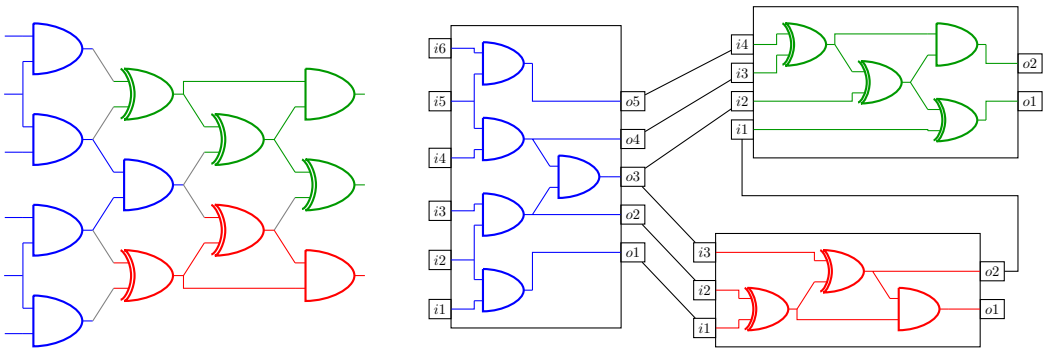

Figure 8.1: On the left side we present the description of a (compiled) circuit. On the right side the same circuit is represented as three different components. The mapping function $\mathcal{M}$ establishes the connections between the blue component and the green and red components.

- $z := ((\widehat{\Gamma}_1, \ldots, \widehat{\Gamma}_n), \mathcal{M}, (I_1, \ldots, I_m))$, *with* $n \in \mathbb{N}$ *and* $I_j \subseteq [n]$, *for* $j \in [m]$, *mutually disjoint subsets.*
- $(\widehat{\Gamma}_1, \ldots, \widehat{\Gamma}_n)$ *forms a partition of* $\widehat{\Gamma}$ *with* n *components, where* $\widehat{\Gamma}_i := (V_i, E_i)$.
- $\mathcal{M} : V \times V \to \{0, 1\}$ *is a function such that* $\mathcal{M}(v, v') = 1$ *iff* $v, v' \in V_i, V_j$ *for some* $i \neq j$, *for* $i, j \in [n]$ *and* $(v, v') \in E$, *i.e.,* $\mathcal{M}$ *defines the connectivity between the sub-components* $(\widehat{\Gamma}_1, \ldots, \widehat{\Gamma}_n)$.

We call $\rho := \frac{\sum_{i \in [n] \setminus I_1 \cup \ldots \cup I_m} |\widehat{\Gamma}_i|}{|\Gamma|}$ *the outsourcing ratio of the compiler.*

Informally, the outsourcing ratio ρ, is the ratio of the size of the circuit that is built in-house, over the total size of the original circuit, Γ. Also, the sub-components $(\widehat{\Gamma}_i)_{i \in [n]}$ "cover" the entire compiled circuit $\widehat{\Gamma}$, without any overlap, and the mapping function $\mathcal{M}$ specifies the connectivity between the sub-components $(\widehat{\Gamma}_1, \ldots, \widehat{\Gamma}_n)$, enabling the reconstruction of $\widehat{\Gamma}$. For $j \in [m]$, the set of indices $I_j \subseteq [n]$, defines the set of sub-components that will be outsourced to the manufacturer with index j, i.e., the total number of manufacturers is m. See Fig. 8.1, for a pictorial representation of a simple toy example.

Correctness of an outsourcing compiler requires that the compiled circuit realizes the same functionality with the original one.

Definition 8.2.2 (Correctness)**.** *We say that an outsourcing compiler* Φ, *is functionality preserving, if for all circuits* Γ, *for all values of the initial memory* M_1, *and for any set of public inputs* $X_1, \ldots, X_q$, *the sequence of outputs* $Y_1, \ldots, Y_q$ *produced by running the original circuit* Γ *with initial state* M_1, *is identical to the sequence of outputs produced by*

running the compiled circuit $\widehat{\Gamma}$, with initial state M_1 (with all but negligible probability over the randomness of the compiler and the randomness of the original and compiled circuit).

For randomized functionalities, we require the output distributions of the original and the compiled, circuits, to be statistically close.

8.2.2 Simulation-based security

In what follows we define security using the *real/ideal* paradigm. Our approach is similar in spirit to previous works in tamper resilient cryptography (see, e.g., [IPSW06, FPV11, KT13]), which aims at protecting cryptographic circuits, against active physical attacks. In a nutshell, in the real/idea paradigm, security is defined by comparing two experiments. In the *real experiment*, the circuit designer compiles the circuit and outsources the production of some of the resulting components, to a set of m untrusted manufacturers. A subset of size at most t of those manufacturers can be malicious, and be controlled by a monolithic adversary $\mathcal{A}$, and the circuit designer cannot distinguish between honest and malicious manufacturers. During production, $\mathcal{A}$ is allowed to modify the outsourced circuit components, arbitrarily, e.g., by adding, removing or modifying gates and/or wires. Later, the designer assembles the circuit by combining all the components (the outsourced ones and the ones built in-house). Finally, $\mathcal{A}$ accesses the assembled circuit in a *black-box* way, that is, it is allowed to execute the circuit and observe the output of it on inputs of its choice, with some initial and unknown memory value M_1. The purpose of the adversary in the real execution, is to leak information over the circuit's private memory, by exploiting the fact that some circuit components have been maliciously modified during the production process.

In the *ideal experiment*, a simulator $\mathcal{S}$, is given black-box access to the original circuit, with initial memory M_1. The goal of the simulator is to produce an output distribution, which is indistinguishable from the one in the real experiment, *without accessing* M_1. In its most general form, our definition allows the simulator to obtain a short leakage over M_1, and this captures a real world scenario in which the adversary could possibly learn a short amount of information over the circuit's private memory.

Below, we formally define the real/ideal experiment.

Real experiment. The distribution $\mathbf{Real}_{\mathcal{A},\Phi,\mathcal{C},\Gamma,M_1}(k)$ is parameterized by the adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$, the set of corrupted manufacturers $\mathcal{C}$, the compiler Φ, and the original circuit Γ, with initial memory M_1.

1. $(\widehat{\Gamma}, z) \leftarrow \Phi(\Gamma)$: In the first step, the description of the original circuit Γ is given as input to the compiler Φ; the compiler outputs the description of the compiled circuit $\widehat{\Gamma}$ plus the auxiliary information $z := ((\widehat{\Gamma}_1, \ldots, \widehat{\Gamma}_n), \mathcal{M}, (I_1, \ldots, I_m))$ which specifies (1) how the compiled circuit is split into sub-components, (2) how the different sub-components are connected (via the mapping function $\mathcal{M}$), and (3) the subset of sub-components whose production is outsourced to each manufacturer (in the index sets I_j, for $j \in [m]$).

2. $(\{\widehat{\Gamma}'_i\}_{i \in I}, st) \leftarrow \mathcal{A}_0(1^k, \{\widehat{\Gamma}_i\}_{i \in I}, \Gamma, \widehat{\Gamma})$: The adversary is given as input the description of the components from the index set $I = \cup_{j \in \mathcal{C}} I_j$, the description of the original circuit Γ, the description of the compiled circuit $\widehat{\Gamma}$, and returns the modified components along with some auxiliary state information, st.

3. $\widehat{\Gamma}' := (\widehat{V}', \widehat{E}')$: The compiled circuit $\widehat{\Gamma}'$ is rebuilt by replacing the components $(\widehat{\Gamma}_i)_{i \in I}$ with the modified components $(\widehat{\Gamma}'_i)_{i \in I}$, and by connecting the different components as specified by the mapping $\mathcal{M}$.

4. $\mathcal{A}_1^{\widehat{\Gamma}'[M_1](\cdot)}(1^k, st)$: The adversary $\mathcal{A}_1$, with auxiliary information st, is given oracle access to the circuit $\widehat{\Gamma}'$, with private memory M_1.

Below, we define the *ideal experiment*.

Ideal experiment. The distribution $\mathbf{Ideal}_{\mathcal{S},\mathcal{A},\Phi,\mathcal{C},\Gamma,M_1,\ell}(k)$ is parameterized by the simulator $\mathcal{S}$, the adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$, the compiler Φ, the set of corrupt manufacturers $\mathcal{C}$, the original circuit Γ with initial memory M_1, and some value $\ell \in \mathbb{N}$.

1. $f \leftarrow \mathcal{S}(1^k, \Gamma, \Phi, \mathcal{A}, \mathcal{C}, \ell)$: Given as input a description of the original circuit, of the compiler and the adversary, the subset of corrupt manufacturers, and the parameter $\ell \in \mathbb{N}$, the simulator specifies an arbitrary polynomial-time computable function $f : \{0,1\}^* \rightarrow \{0,1\}^\ell$.

2. $\mathcal{S}^{\mathcal{A},\Gamma[M_1](\cdot)}(1^k, L)$: The simulator takes as input leakage $L = f(M_1)$, and is given oracle access to the adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ and the original circuit Γ, with private memory M_1. We remark that the simulator is restricted to be fully black-box. In particular, $\mathcal{S}$ only accesses the modified sub-components returned by $\mathcal{A}_0$ in a black-box way, i.e., without knowing their description.

Having the above definitions, we formally define security for outsourcing circuit compilers.

Definition 8.2.3 (Security)**.** *We say that a* (ρ, m)*-outsourcing circuit compiler* Φ *is* (ℓ, t)*-secure, if the following conditions are met.*

(i) Non-triviality: $\rho < 1$*, for sufficiently large values of* $k \in \mathbb{N}$*.*

(ii) Simulatability: *For all* $\mathcal{C} \subseteq [m]$ *of size at most* t*, for all circuits* Γ*, and for all* PPT *adversaries* $\mathcal{A}$*, there exists a simulator* $\mathcal{S}$ *with running time* $\mathrm{poly}(|\mathcal{A}|, |\Gamma|)$*, such that for all initial values of the memory* $M_1 \in \{0,1\}^*$*,*

$$\{\mathbf{Real}_{\mathcal{A},\Phi,\mathcal{C},\Gamma,M_1}(k)\}_{k\in\mathbb{N}} \approx_c \{\mathbf{Ideal}_{\mathcal{S},\mathcal{A},\Phi,\mathcal{C},\Gamma,M_1,\ell}(k)\}_{k\in\mathbb{N}}\,.$$

In the above definition, the adversary is allowed to modify each $\widehat{\Gamma}_i$ arbitrarily, i.e., there is no restriction on the edges and nodes of $\widehat{\Gamma}'_i$, as long as the input and output gates enable connectivity with the remaining components. Also, observe that, the above definition is only interesting for small values of ℓ (as, e.g., it becomes trivial in case $\ell = |M_1|$). Finally, the non-triviality condition requires that the ratio of the size of the sub-components built in-house, over the size of the original circuit, should be less than one, as otherwise a manufacturer could simply produce the entire circuit by itself, without using any off-shore facility. Clearly, smaller values of ρ, enable outsourcing of larger fractions of the original circuit.

8.3 A circuit outsourcing compiler based on MPC

In the current section, we construct a compiler based on *multi-party computation* (MPC). Before presenting our compiler, we first revisit the core ideas of MPC and then we give a generic definition for MPC protocols in the *client-server* model, along the lines of [Bea97].

MPC in the Client-Server Model. We consider $p \in \mathbb{N}$ parties, where each party P_i, for $1 \leq i \leq p$, possesses an input X_i and they all wish to jointly compute the vector $(Y_1, \ldots, Y_p) := \mathcal{F}(X_1, \ldots, X_p)$, where Y_i is the output for party P_i. In the client-server model, the parties are divided into two categories: the *clients*, that provide inputs and wish to receive the output of the computation, and the *servers*, that perform the computation. A *t-private* MPC protocol guarantees that any adversary who controls up to t servers, cannot leak any information related to the private inputs of the clients, besides the information that can be inferred by the output of the computation, and regardless of the number of corrupted clients.

In our construction, the circuits that correspond to the code executed by the servers, will be outsourced to a number of possibly malicious manufacturers, that may apply arbitrary modifications against the circuit components. Thus, we require MPC protocols that are secure against *active* (*malicious*) *attackers*. The general idea behind our compiler is the following. Let $\mathcal{F}$ be any functionality, let Γ be a circuit implementing $\mathcal{F}$, and let $\Pi_{\mathcal{F}}$ be a t-private MPC protocol, realizing $\mathcal{F}$. Then, assuming that the number of malicious manufacturers is at most $t < m$, the circuit $\widehat{\Gamma}$ will implement the code of $\Pi_{\mathcal{F}}$, and each $\widehat{\Gamma}_i$ will implement the code of the i-th server.

Below, we define the protocol framework that we are going to use for the rest of this section. The idea is to describe any MPC protocol using its *next message function*, denoted as Next.

Definition 8.3.1 (Protocols using the). *Let C, S, be sets of probabilistic interactive Turing machines, with cardinalities p, m, respectively. An r-round protocol Π for p clients and m servers is a tuple $(C, S, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Next})$, where $\mathsf{Next} = (\mathsf{Next}_1, \ldots, \mathsf{Next}_m)$, is described as follows.*

- Setup: *Each client computes $(X_i^1, \ldots, X_i^m) \leftarrow \mathsf{Enc}(X_i)$, and sends X_i^j to the server indexed by j. For $j \in [m]$, let $\mathbf{in}^j := (X_1^j, \ldots, X_p^j)$ and $st_j := 0$.*[2]
- Computation:
  - *In each round, for $j \in [m]$ execute $(o_1^j, \ldots, o_m^j, st'_j) \leftarrow \mathsf{Next}_j(\mathbf{in}^j, st_j)$, send o_k^j, $k \neq j$, to the server with index k. Set $\mathbf{in}^j \leftarrow (o_j^1, \ldots, o_j^m)$ and $st_j \leftarrow st'_j$.*
  - *In the final round, for $j \in [m]$ execute $o^j \leftarrow \mathsf{Next}_j(\mathbf{in}^j, st_j)$ and send o_j to Dec.*
- Output: *Execute $(Y_1, \ldots, Y_p) \leftarrow \mathsf{Dec}(o_1, \ldots, o_m)$, and send Y_j to the client with index j.*

For any function $\mathcal{F}$, the protocol computing $\mathcal{F}$ will be denoted by $\Pi_{\mathcal{F}}$.

Informally, in the first step of the protocol execution, the clients encode their inputs, as it is prescribed by Enc, and then the main computation begins. The code executed by the servers at each round is defined by the function Next, the next message function. Hence, in the i-th round, server S_j computes Next_j over the outputs and the state information, st, produced by the other servers in round $i-1$. One can also consider deterministic next

[2]Here, we assume that the network is fully connected, still the properties of the communication channel depend on the instantiation.

message functions, assuming the randomness is given as input in each round. Below, we formally define *correctness* and *privacy* for MPC protocols.

Definition 8.3.2 (Correctness)**.** *Let $\mathcal{F}$ be a p-party functionality. We say that Π realizes $\mathcal{F}$ with perfect (resp., statistical) correctness if for any input $(X_1, \ldots, X_p)$, the probability that the output delivered to the i-th client, for $i \in [p]$, after the protocol execution, is different from Y_i, is 0 (resp., negligible in the security parameter), where $(Y_1, \ldots, Y_p) := \mathcal{F}(X_1, \ldots, X_p)$.*

The definition of privacy follows.

Definition 8.3.3 ((t,m)-privacy)**.** *Let k be the security parameter, p be the number of clients and m be the number of servers, and let $\mathcal{A}$ be an adversary that may corrupt any set of parties $I_c \subseteq [p]$, and servers $I_s \subset [m]$, where $|I_s| \leq t$. We say that the protocol Π realizes $\mathcal{F}$ with (t,m)-privacy, if there exists a* PPT *algorithm $\mathcal{S}$ such that for all sufficiently large $k \in \mathbb{N}$,*

$$\mathsf{View}_{I_s,I_c}(k, X_1, \ldots, X_p) \approx_c \mathcal{S}(1^k, I_c, I_s, (X_i, Y_i)_{i \in I_c}),$$

where $\mathsf{View}_{I_s,I_c}(k, X_1, \ldots, X_p)$ denotes the joint view of the servers and clients in I_s and I_c, respectively, within an execution of the protocol upon inputs $X_1, \ldots, X_p$, and $(Y_1, \ldots, Y_p) = \mathcal{F}(X_1, \ldots, X_p)$.

The main idea behind the above definition is that the view of the adversary during the protocol execution depends only on its own input and output.

Below we present our compiler.

The compiler $\Phi_{\Pi_\mathcal{F}}$. Let Γ be a circuit implementing the function $\mathcal{F}(M_1, \cdot)$, where for any X and $i \in \mathbb{N}$, we have $(Y, M_{i+1}) = \mathcal{F}(M_i, X)$. Let $\Pi_\mathcal{F} = (C, S, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Next})$ be a protocol realizing $\mathcal{F}$, over a set of m servers and a single client. The compiler produces $(\widehat{\Gamma}, \mathsf{aux}) \leftarrow \Phi_{\Pi_\mathcal{F}}(\Gamma)$, where

- $\widehat{\Gamma}$ is the circuit that implements $\Pi_\mathcal{F}$ (depicted in Figure 8.2 for the case $m = 2$ and $p = 1$), having as a sub-circuit $\widehat{\Gamma}_{\mathsf{Memory}}$, which is a circuit consisting only of memory gates, as needed by the original circuit Γ. During initialization, $\widehat{\Gamma}_{\mathsf{Memory}}$ stores the initial private memory value, M_1.
- $z = ((\widehat{\Gamma}_1, \ldots, \widehat{\Gamma}_{m+2}), \mathcal{M}, (I_1, \ldots, I_m))$, where

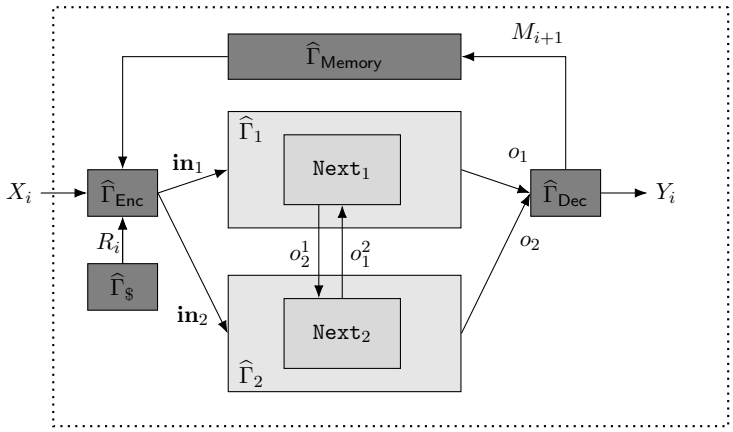

Figure 8.2: The MPC compiler for the case of $m = 2$ outsourcing facilities. The components $\widehat{\Gamma}_1$ and $\widehat{\Gamma}_2$ can be outsourced, while the connectivity between them and the remaining components are built in-house. $\widehat{\Gamma}_1$ and $\widehat{\Gamma}_2$ exchange the outputs of the next message functions Next_1, Next_2, and they internally update their states. The dotted line depicts the circuit boundaries.

- $\widehat{\Gamma}_{m+1} = \widehat{\Gamma}_{\mathsf{Enc}}$ and $\widehat{\Gamma}_{m+2} = \widehat{\Gamma}_{\mathsf{Dec}}$, i.e., the circuits $\widehat{\Gamma}_{m+1}$ and $\widehat{\Gamma}_{m+2}$ implement the encoder, Enc, and the decoder, Dec, of $\Pi_{\mathcal{F}}$, respectively.
- For $i \in [m]$, $\widehat{\Gamma}_i$ is the circuit that implements the code of the i-th server, for the entire execution of $\Pi_{\mathcal{F}}$. Those circuits can be implemented in a straightforward way using the next message function Next_i (cf. the sub-components $\widehat{\Gamma}_1$ and $\widehat{\Gamma}_2$ in Figure 8.2).
- The mapping function $\mathcal{M}$ describes the physical connections between the circuits described above, and I_j, for $j \in [m]$, specifies the components that will be outsourced to the manufacturer with index j. In our case $I_j = \{j\}$.
- In case the original circuit is randomized, in addition to the components described above, Φ also outputs a circuit $\widehat{\Gamma}_{\$}$ producing randomness R_i, that is needed in each invocation of the circuit.

Our construction must be non-trivial (cf. Definition 8.2.3), thus the underlying protocol Π must satisfy the following outsourceability property.

Definition 8.3.4 (Outsourceability of procotols)**.** *A protocol* $\Pi = (C, S, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Next})$ *that realizes the function* $\mathcal{F}$ *can be outsourced if it satisfies the following condition: The circuit computing the encoding and decoding procedures* $(\mathsf{Enc}, \mathsf{Dec})$ *must be smaller than the circuit computing the function* $\mathcal{F}$*.*

Below we prove security for the proposed compiler.

Theorem 8.3.5. *Let $\mathcal{F}$ be any function, and let $\Pi_{\mathcal{F}}$ be a (t,m)-private MPC protocol for $\mathcal{F}$, satisfying the correctness and outsourceability properties. Then, the compiler $\Phi_{\Pi_{\mathcal{F}}}$ is a correct, $(0,t)$-secure, (ρ,m)-outsourcing circuit compiler, for $\rho<1$.*

Proof. The correctness and outsourceability of $\Phi_{\Pi_{\mathcal{F}}}$ follow directly by the corresponding properties of $\Pi_{\mathcal{F}}$.

Let $\mathcal{F}$ be any functionality and let Γ be the circuit implementing $\mathcal{F}$. Assuming that $\Pi_{\mathcal{F}}$ is a (t,m)-private MPC protocol for $\mathcal{F}$, we will prove that $\Phi_{\Pi_{\mathcal{F}}}$ is a $(0,t)$-secure, circuit compiler. Concretely (cf. Definition 8.2.3), we need to prove that for all $\mathcal{C}\subseteq[m]$ of size at most t, all circuits Γ, all PPT adversaries $\mathcal{A}$, and for all initial values of the memory $M_1\in\{0,1\}^*$, there exists a simulator $\mathcal{S}$ with running time poly$(|\mathcal{A}|,|\Gamma|)$ such that

$$\{\mathbf{Real}_{\mathcal{A},\Phi,\mathcal{C},\Gamma,M_1}(k)\}_{k\in\mathbb{N}}\approx_c\{\mathbf{Ideal}_{\mathcal{S},\mathcal{A},\Phi,\mathcal{C},\Gamma,M_1,\ell}(k)\}_{k\in\mathbb{N}}, \tag{8.1}$$

for all sufficiently large values of k. Let $\mathcal{A}$ be an attacker $\Phi_{\Pi_{\mathcal{F}}}$. The idea behind the proof is to relate the interaction between $\mathcal{A}$ and the circuits produced by $\Phi_{\Pi_{\mathcal{F}}}$, with the interaction between an attacker $\mathcal{A}'$ corrupting up to t servers, while executing $\Pi_{\mathcal{F}}$. Then, we will use the simulator $\mathcal{S}'$ that is given by the (t,m)-privacy of $\Pi_{\mathcal{F}}$ to construct a simulator $\mathcal{S}$, satisfying equation 8.1. In what follows, and for the sake of simplicity, we prove the needed assuming $\mathcal{A}$ is a single round attacker, and then we discuss how the proof easily extends to the setting of multiple executions.

By the compiler definition, the protocol $\Pi_{\mathcal{F}}$ that $\Phi_{\Pi_{\mathcal{F}}}$ is based on, consists of two clients, C_1, C_2, where C_1 is the corrupted client that provides the public input to the circuit, X, and C_2 supplies the circuit with private input, M, and m servers. Let Γ be the circuit implementing $\mathcal{F}$. Given the adversary $\mathcal{A}$ for $\Phi_{\Pi_{\mathcal{F}}}$ we define the adversary $\mathcal{A}'=(\mathcal{A}'_0,\mathcal{A}'_1)$ against $\Pi_{\mathcal{F}}$ as follows:

- **(server corruption)**: $\mathcal{A}'_0$ runs $(\widehat{\Gamma},z)\leftarrow\Phi(\Gamma)$, where $z:=((\widehat{\Gamma}_1,\ldots,\widehat{\Gamma}_n),\mathcal{M},(I_1,\ldots,I_m))$, and samples $(\{\widehat{\Gamma}'_i\}_{i\in I},st)\leftarrow\mathcal{A}_0(1^k,\{\widehat{\Gamma}_i\}_{i\in I},\Gamma,\widehat{\Gamma})$. Then corrupts the server S_i, for $i\in I$, so that S_i will execute the possibly modified circuit $\widehat{\Gamma}'_i$.

- **(protocol execution)**: $\mathcal{A}'_1$ participates in the protocol $\Pi_{\mathcal{F}}$ choosing the input for client C_1 (the corrupted client), according to the input value chosen by $\mathcal{A}_1$. Concretely, $\mathcal{A}'_1$ executes the following steps: samples $X\leftarrow\mathcal{A}_1(1^k,st)$, defines the input of client C_1 to be equal to X, receives the output of $\Pi_{\mathcal{F}}$ for C_1, Y, for inputs (X,M), and forwards Y to $\mathcal{A}_1$.

We define the random variable $\mathsf{View}_{I_s,I_c}(k, X, M)$, $I_s = \mathcal{C}$, $I_c = \{1\}$, to be the view of $\mathcal{A}$ in the executing defined above. Clearly, by the definition of $\mathcal{A}'$, the view of $\mathcal{A}$ matches its view while executing the real world experiment of Definition 8.2.3, thus we have

$$\mathsf{View}_{I_s,I_c}(k, X, M) = \mathbf{Real}_{\mathcal{A},\Phi,\mathcal{C},\Gamma,M_1}(k). \tag{8.2}$$

Assuming $\Pi_{\mathcal{F}}$ is (t, m)-private against $\mathcal{A}'$, there exists exists a simulator $\mathcal{S}''$ that simulates the view of $\mathcal{A}'$ during the protocol execution. Let $\mathcal{S}'$ be code of $\mathcal{S}''$ that only outputs the view of $\mathcal{A}$. Then we have that for all sufficiently large $k \in \mathbb{N}$,

$$\mathsf{View}_{I_s,I_c}(k, X, M) \approx_c \mathcal{S}'(1^k, I_c, I_s, (X, Y)_{i\in I_c}). \tag{8.3}$$

Now we define the simulator $\mathcal{S}$ for $\mathcal{A}$ against $\Phi_{\Pi_{\mathcal{F}}}$. $\mathcal{S}$ on input $(1^k, \Gamma, \Phi, \mathcal{A}, \mathcal{C}, 0)$ executes the following steps:

- executes $\mathcal{A}_1$ with oracle access to $\Gamma[M_1](\cdot)$, and constructs the pair (X, Y), i.e., it constructs the valid output of $\mathcal{F}$ on input X, chosen by $\mathcal{A}_1$.
- executes $o \leftarrow \mathcal{S}'(1^k, I_c, I_s, (X, Y)_{i\in I_c})$, where $I_s = \mathcal{C}$ and $I_c = \{1\}$, and outputs o.

Clearly, from equation 8.3 we have that $\mathcal{S}'$ produces output which is computationally indistinguishable from $\mathsf{View}_{I_s,I_c}(k, X, M)$, and then using equation 8.2 we receive,

$$\mathbf{Real}_{\mathcal{A},\Phi,\mathcal{C},\Gamma,M_1}(k) \approx_c \mathbf{Ideal}_{\mathcal{S},\mathcal{A},\Phi,\mathcal{C},\Gamma,M_1,\ell}(k),$$

and this concludes the proof for the case of single round adversaries.

For multi-round attackers against the circuit compiler, we need to have multiple, sequential executions, of the same protocol, as a single execution computes a single circuit output. Moreover, the attacker is non-adaptive, and corrupts the servers only before the first protocol execution. By the composition theorem of [Can00], we have that any secure MPC protocol is also secure against sequential composition, even against adaptive adversaries. Using a standard hybrid argument, this gives rise to a simulator, $\mathcal{S}'$, that simulates the view of the attacker for all executions, and the proof idea is identical to one given above: we relate the attacker against the compiler to an attacker against the protocol, and we use $\mathcal{S}'$ to construct a simulator $\mathcal{S}$ for the circuit compiler. □

Our compiler is generic, thus any MPC protocol with an efficient preprocessing phase could be used to instantiate the above compiler, when adapted to the client-server model. For instance, in the recent work by Wang et al. [WRK17], the preprocessing phase has

complexity $O(m^2k/\log(|\Gamma|))$ and achieves security against $m-1$ malicious parties. Note that, cost minimization might not be the primary goal behind circuit outsourcing, as the lack of technical expertise is also an important factor. In this setting, the size of the components built in-house could be proportional to (or greater than) the size of the outsourced ones.

Chapter 9

Conclusions

9.1 Contributions and future directions

The present dissertation deals with the problem of protecting cryptographic hardware against *physical* and *hardware Trojan injection*, attacks, based on the notions of *non-malleable codes* and *multi-party computation*. Besides that, our techniques find useful application the problem of establishing *secure communication* in the presence of *man-in-the-middle adversaries*. Our results are briefly summarized below.

- [KLT18a]: We construct *efficient* non-malleable codes for the class of *partial functions*, with the additional property of *manipulation detection*, which guarantees that any tampered codeword will either decode to the original message, or to $\perp$.
- [KLT16]: We introduce the notion of *ℓ-more weakly extractable, collision resistant, hash functions* (wECRH) and we use it as the main tool for constructing practically efficient non-malleable codes for *split-state* adversaries.
- [KLT18b]: We introduce the notion of *leakage-resilient* ℓ-more wECRH and we use it as the main tool for building efficient, *continuous non-malleable codes*, for split-state adversaries. In addition, we prove that any ℓ-more wECRH, with some additional properties, yields *efficient, succinct, non-interactive non-malleable* commitments.
- [AKM$^+$18]: Finally, we provide a simulation-based definition for the problem of secure circuit outsourcing, ensuring privacy of the circuit's private memory, even in the presence of *hardware Trojans*. As a feasibility result for the problem at study, we propose a *multi-party computation*-based compiler that transforms any crypto-

graphic circuit, into another, that can be securely outsourced, even in the presence of malicious manufacturers.

The current thesis makes an important step towards bridging the gap between theory and practice, especially with respect to the notion of non-malleable codes and its applications in the real-world setting. This is derived from the fact that, besides providing rigorous mathematical models of security, together with provable secure solutions, the current thesis develops practically efficient solutions, under realistic security models that comply with existing attacks. As such, our constructions yield efficient solutions against memory-tampering attacks on cryptographic hardware, as well as for establishing secure communication in the presence of adversarial channels. Non-malleable codes with additional properties, such as manipulation detection, broaden the applicability of the primitive beyond the scope of tamper-resilient cryptography, thus, investigating new, realistic adversarial models, for which manipulation detection, or similar properties, could be achieved, as well all exploring other applications of the primitive, are research directions that worth to be pursued.

Bibliography

[AAG+16] Divesh Aggarwal, Shashank Agrawal, Divya Gupta, Hemanta K. Maji, Omkant Pandey, and Manoj Prabhakaran. Optimal computational split-state non-malleable codes. In Eyal Kushilevitz and Tal Malkin, editors, *TCC 2016-A: 13th Theory of Cryptography Conference, Part II*, volume 9563 of *Lecture Notes in Computer Science*, pages 393–417, Tel Aviv, Israel, January 10–13, 2016. Springer, Heidelberg, Germany.

[ADKO15] Divesh Aggarwal, Yevgeniy Dodis, Tomasz Kazana, and Maciej Obremski. Non-malleable reductions and applications. In Rocco A. Servedio and Ronitt Rubinfeld, editors, *47th Annual ACM Symposium on Theory of Computing*, pages 459–468, Portland, OR, USA, June 14–17, 2015. ACM Press.

[ADL14] Divesh Aggarwal, Yevgeniy Dodis, and Shachar Lovett. Non-malleable codes from additive combinatorics. In David B. Shmoys, editor, *46th Annual ACM Symposium on Theory of Computing*, pages 774–783, New York, NY, USA, May 31 – June 3, 2014. ACM Press.

[AF07] Masayuki Abe and Serge Fehr. Perfect NIZK with adaptive soundness. In Salil P. Vadhan, editor, *TCC 2007: 4th Theory of Cryptography Conference*, volume 4392 of *Lecture Notes in Computer Science*, pages 118–136, Amsterdam, The Netherlands, February 21–24, 2007. Springer, Heidelberg, Germany.

[AGM+15a] Shashank Agrawal, Divya Gupta, Hemanta K. Maji, Omkant Pandey, and Manoj Prabhakaran. Explicit non-malleable codes against bit-wise tampering and permutations. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *Advances in Cryptology – CRYPTO 2015, Part I*, volume 9215 of

Lecture Notes in Computer Science, pages 538–557, Santa Barbara, CA, USA, August 16–20, 2015. Springer, Heidelberg, Germany.

[AGM+15b] Shashank Agrawal, Divya Gupta, Hemanta K. Maji, Omkant Pandey, and Manoj Prabhakaran. A rate-optimizing compiler for non-malleable codes against bit-wise tampering and permutations. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015: 12th Theory of Cryptography Conference, Part I*, volume 9014 of *Lecture Notes in Computer Science*, pages 375–397, Warsaw, Poland, March 23–25, 2015. Springer, Heidelberg, Germany.

[AK96] Ross Anderson and Markus Kuhn. Tamper resistance: A cautionary note. In *Proceedings of the 2Nd Conference on Proceedings of the Second USENIX Workshop on Electronic Commerce - Volume 2*, WOEC'96, pages 1–1, Berkeley, CA, USA, 1996. USENIX Association.

[AKM+18] Giuseppe Ateniese, Aggelos Kiayias, Bernardo Magri, Yiannis Tselekounis, and Daniele Venturi. Secure outsourcing of cryptographic circuits manufacturing. In Joonsang Baek, Willy Susilo, and Jongkil Kim, editors, *Provable Security*, pages 75–93, Cham, 2018. Springer International Publishing.

[AKO17] Divesh Aggarwal, Tomasz Kazana, and Maciej Obremski. Inception makes non-malleable codes stronger. In Yael Kalai and Leonid Reyzin, editors, *TCC 2017: 15th Theory of Cryptography Conference, Part II*, volume 10678 of *Lecture Notes in Computer Science*, pages 319–343, Baltimore, MD, USA, November 12–15, 2017. Springer, Heidelberg, Germany.

[BCC+14] Nir Bitansky, Ran Canetti, Alessandro Chiesa, Shafi Goldwasser, Huijia Lin, Aviad Rubinstein, and Eran Tromer. The hunting of the SNARK. Cryptology ePrint Archive, Report 2014/580, 2014. `http://eprint.iacr.org/2014/580`.

[BCC+17] Nir Bitansky, Ran Canetti, Alessandro Chiesa, Shafi Goldwasser, Huijia Lin, Aviad Rubinstein, and Eran Tromer. The hunting of the SNARK. *Journal of Cryptology*, 30(4):989–1066, October 2017.

[BCCT12] Nir Bitansky, Ran Canetti, Alessandro Chiesa, and Eran Tromer. From extractable collision resistance to succinct non-interactive arguments of knowl-

edge, and back again. In Shafi Goldwasser, editor, *ITCS 2012: 3rd Innovations in Theoretical Computer Science*, pages 326–349, Cambridge, MA, USA, January 8–10, 2012. Association for Computing Machinery.

[BCCT13] Nir Bitansky, Ran Canetti, Alessandro Chiesa, and Eran Tromer. Recursive composition and bootstrapping for SNARKS and proof-carrying data. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *45th Annual ACM Symposium on Theory of Computing*, pages 111–120, Palo Alto, CA, USA, June 1–4, 2013. ACM Press.

[BCPR14] Nir Bitansky, Ran Canetti, Omer Paneth, and Alon Rosen. On the existence of extractable one-way functions. In David B. Shmoys, editor, *46th Annual ACM Symposium on Theory of Computing*, pages 505–514, New York, NY, USA, May 31 – June 3, 2014. ACM Press.

[BDH+98] F. Bao, R. H. Deng, Y. Han, A. Jeng, A. D. Narasimhalu, and T. Ngair. *Breaking public key cryptosystems on tamper resistant devices in the presence of transient faults*, pages 115–124. Springer Berlin Heidelberg, Berlin, Heidelberg, 1998.

[BDK+11] Boaz Barak, Yevgeniy Dodis, Hugo Krawczyk, Olivier Pereira, Krzysztof Pietrzak, François-Xavier Standaert, and Yu Yu. Leftover hash lemma, revisited. In Phillip Rogaway, editor, *Advances in Cryptology – CRYPTO 2011*, volume 6841 of *Lecture Notes in Computer Science*, pages 1–20, Santa Barbara, CA, USA, August 14–18, 2011. Springer, Heidelberg, Germany.

[BDKM16] Marshall Ball, Dana Dachman-Soled, Mukul Kulkarni, and Tal Malkin. Non-malleable codes for bounded depth, bounded fan-in circuits. In Marc Fischlin and Jean-Sébastien Coron, editors, *Advances in Cryptology – EUROCRYPT 2016, Part II*, volume 9666 of *Lecture Notes in Computer Science*, pages 881–908, Vienna, Austria, May 8–12, 2016. Springer, Heidelberg, Germany.

[BDKM18] Marshall Ball, Dana Dachman-Soled, Mukul Kulkarni, and Tal Malkin. Non-malleable codes from average-case hardness: AC^0, decision trees, and streaming space-bounded tampering. In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2018, Part III*, volume

10822 of *Lecture Notes in Computer Science*, pages 618–650, Tel Aviv, Israel, April 29 – May 3, 2018. Springer, Heidelberg, Germany.

[BDL97] Dan Boneh, Richard A. DeMillo, and Richard J. Lipton. On the importance of checking cryptographic protocols for faults (extended abstract). In Walter Fumy, editor, *Advances in Cryptology – EUROCRYPT'97*, volume 1233 of *Lecture Notes in Computer Science*, pages 37–51, Konstanz, Germany, May 11–15, 1997. Springer, Heidelberg, Germany.

[BDL01] Dan Boneh, Richard A. DeMillo, and Richard J. Lipton. On the importance of eliminating errors in cryptographic computations. *Journal of Cryptology*, 14(2):101–119, 2001.

[Bea97] Donald Beaver. Commodity-based cryptography (extended abstract). In *29th Annual ACM Symposium on Theory of Computing*, pages 446–455, El Paso, TX, USA, May 4–6, 1997. ACM Press.

[BGI+01] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. In Joe Kilian, editor, *Advances in Cryptology – CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 1–18, Santa Barbara, CA, USA, August 19–23, 2001. Springer, Heidelberg, Germany.

[BKKV10] Zvika Brakerski, Yael Tauman Kalai, Jonathan Katz, and Vinod Vaikuntanathan. Overcoming the hole in the bucket: Public-key cryptography resilient to continual memory leakage. In *51st Annual Symposium on Foundations of Computer Science*, pages 501–510, Las Vegas, NV, USA, October 23–26, 2010. IEEE Computer Society Press.

[Boy99] Victor Boyko. On the security properties of OAEP as an all-or-nothing transform. In Michael J. Wiener, editor, *Advances in Cryptology – CRYPTO'99*, volume 1666 of *Lecture Notes in Computer Science*, pages 503–518, Santa Barbara, CA, USA, August 15–19, 1999. Springer, Heidelberg, Germany.

[BP04] Mihir Bellare and Adriana Palacio. The knowledge-of-exponent assumptions and 3-round zero-knowledge protocols. In Matthew Franklin, editor, *Advances in Cryptology – CRYPTO 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 273–289, Santa Barbara, CA, USA, August 15–19, 2004. Springer, Heidelberg, Germany.

[BP15] Elette Boyle and Rafael Pass. Limits of extractability assumptions with distributional auxiliary input. In Tetsu Iwata and Jung Hee Cheon, editors, *Advances in Cryptology – ASIACRYPT 2015, Part II*, volume 9453 of *Lecture Notes in Computer Science*, pages 236–261, Auckland, New Zealand, November 30 – December 3, 2015. Springer, Heidelberg, Germany.

[BR15] Shivam Bhasin and Francesco Regazzoni. A survey on hardware trojan detection techniques. In *IEEE ISCAS*, pages 2021–2024, 2015.

[BRPB14] Georg T. Becker, Francesco Regazzoni, Christof Paar, and Wayne P. Burleson. Stealthy dopant-level hardware trojans: extended version. *J. Cryptographic Engineering*, 4(1):19–31, 2014.

[BS97] Eli Biham and Adi Shamir. Differential fault analysis of secret key cryptosystems. In Burton S. Kaliski Jr., editor, *Advances in Cryptology – CRYPTO'97*, volume 1294 of *Lecture Notes in Computer Science*, pages 513–525, Santa Barbara, CA, USA, August 17–21, 1997. Springer, Heidelberg, Germany.

[BTV12] Mihir Bellare, Stefano Tessaro, and Alexander Vardy. Semantic security for the wiretap channel. In Reihaneh Safavi-Naini and Ran Canetti, editors, *Advances in Cryptology – CRYPTO 2012*, volume 7417 of *Lecture Notes in Computer Science*, pages 294–311, Santa Barbara, CA, USA, August 19–23, 2012. Springer, Heidelberg, Germany.

[Can00] Ran Canetti. Security and composition of multiparty cryptographic protocols. *Journal of Cryptology*, 13(1):143–202, 2000.

[CD08] Ran Canetti and Ronny Ramzi Dakdouk. Extractable perfectly one-way functions. In Luca Aceto, Ivan Damgård, Leslie Ann Goldberg, Magnús M. Halldórsson, Anna Ingólfsdóttir, and Igor Walukiewicz, editors, *ICALP 2008: 35th International Colloquium on Automata, Languages and Programming, Part II*, volume 5126 of *Lecture Notes in Computer Science*, pages 449–460, Reykjavik, Iceland, July 7–11, 2008. Springer, Heidelberg, Germany.

[CD09] Ran Canetti and Ronny Ramzi Dakdouk. Towards a theory of extractable functions. In Omer Reingold, editor, *TCC 2009: 6th Theory of Cryptography Conference*, volume 5444 of *Lecture Notes in Computer Science*, pages 595–613. Springer, Heidelberg, Germany, March 15–17, 2009.

[CDF+08] Ronald Cramer, Yevgeniy Dodis, Serge Fehr, Carles Padró, and Daniel Wichs. Detection of algebraic manipulation with applications to robust secret sharing and fuzzy extractors. In Nigel P. Smart, editor, *Advances in Cryptology – EUROCRYPT 2008*, volume 4965 of *Lecture Notes in Computer Science*, pages 471–488, Istanbul, Turkey, April 13–17, 2008. Springer, Heidelberg, Germany.

[CDH+00] Ran Canetti, Yevgeniy Dodis, Shai Halevi, Eyal Kushilevitz, and Amit Sahai. Exposure-resilient functions and all-or-nothing transforms. In Bart Preneel, editor, *Advances in Cryptology – EUROCRYPT 2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 453–469, Bruges, Belgium, May 14–18, 2000. Springer, Heidelberg, Germany.

[CDTV16] Sandro Coretti, Yevgeniy Dodis, Björn Tackmann, and Daniele Venturi. Non-malleable encryption: Simpler, shorter, stronger. In Eyal Kushilevitz and Tal Malkin, editors, *TCC 2016-A: 13th Theory of Cryptography Conference, Part I*, volume 9562 of *Lecture Notes in Computer Science*, pages 306–335, Tel Aviv, Israel, January 10–13, 2016. Springer, Heidelberg, Germany.

[CG14] Mahdi Cheraghchi and Venkatesan Guruswami. Capacity of non-malleable codes. In Moni Naor, editor, *ITCS 2014: 5th Conference on Innovations in Theoretical Computer Science*, pages 155–168, Princeton, NJ, USA, January 12–14, 2014. Association for Computing Machinery.

[CGM+15] Nishanth Chandran, Vipul Goyal, Pratyay Mukherjee, Omkant Pandey, and Jalaj Upadhyay. Block-wise non-malleable codes. Cryptology ePrint Archive, Report 2015/129, 2015. `http://eprint.iacr.org/2015/129`.

[CGM+16] Nishanth Chandran, Vipul Goyal, Pratyay Mukherjee, Omkant Pandey, and Jalaj Upadhyay. Block-wise non-malleable codes. In Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi, editors, *ICALP 2016: 43rd International Colloquium on Automata, Languages and Programming*, volume 55 of *LIPIcs*, pages 31:1–31:14, Rome, Italy, July 11–15, 2016. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik.

[CKM11] Seung Geol Choi, Aggelos Kiayias, and Tal Malkin. BiTR: Built-in tamper resilience. In Dong Hoon Lee and Xiaoyun Wang, editors, *Advances in Cryptology – ASIACRYPT 2011*, volume 7073 of *Lecture Notes in Computer*

Science, pages 740–758, Seoul, South Korea, December 4–8, 2011. Springer, Heidelberg, Germany.

[CKR16] Nishanth Chandran, Bhavana Kanukurthi, and Srinivasan Raghuraman. Information-theoretic local non-malleable codes and their applications. In Eyal Kushilevitz and Tal Malkin, editors, *TCC 2016-A: 13th Theory of Cryptography Conference, Part II*, volume 9563 of *Lecture Notes in Computer Science*, pages 367–392, Tel Aviv, Israel, January 10–13, 2016. Springer, Heidelberg, Germany.

[CMTV15] Sandro Coretti, Ueli Maurer, Björn Tackmann, and Daniele Venturi. From single-bit to multi-bit public-key encryption via non-malleable codes. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015: 12th Theory of Cryptography Conference, Part I*, volume 9014 of *Lecture Notes in Computer Science*, pages 532–560, Warsaw, Poland, March 23–25, 2015. Springer, Heidelberg, Germany.

[CNB09] Rajat Subhra Chakraborty, Seetharam Narasimhan, and Swarup Bhunia. Hardware trojan: Threats and emerging solutions. In *IEEE HLDVT*, pages 166–171, 2009.

[COSV16] Michele Ciampi, Rafail Ostrovsky, Luisa Siniscalchi, and Ivan Visconti. Concurrent non-malleable commitments (and more) in 3 rounds. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology – CRYPTO 2016, Part III*, volume 9816 of *Lecture Notes in Computer Science*, pages 270–299, Santa Barbara, CA, USA, August 14–18, 2016. Springer, Heidelberg, Germany.

[COSV17] Michele Ciampi, Rafail Ostrovsky, Luisa Siniscalchi, and Ivan Visconti. Four-round concurrent non-malleable commitments from one-way functions. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology – CRYPTO 2017, Part II*, volume 10402 of *Lecture Notes in Computer Science*, pages 127–157, Santa Barbara, CA, USA, August 20–24, 2017. Springer, Heidelberg, Germany.

[CVZ10] Zhenfu Cao, Ivan Visconti, and Zongyang Zhang. Constant-round concurrent non-malleable statistically binding commitments and decommitments.

In Phong Q. Nguyen and David Pointcheval, editors, *PKC 2010: 13th International Conference on Theory and Practice of Public Key Cryptography*, volume 6056 of *Lecture Notes in Computer Science*, pages 193–208, Paris, France, May 26–28, 2010. Springer, Heidelberg, Germany.

[CW77] J. Lawrence Carter and Mark N. Wegman. Universal classes of hash functions (extended abstract). In *STOC*, pages 106–112, 1977.

[CZ14] Eshan Chattopadhyay and David Zuckerman. Non-malleable codes against constant split-state tampering. In *55th Annual Symposium on Foundations of Computer Science*, pages 306–315, Philadelphia, PA, USA, October 18–21, 2014. IEEE Computer Society Press.

[Dam92] Ivan Damgård. Towards practical public key systems secure against chosen ciphertext attacks. In Joan Feigenbaum, editor, *Advances in Cryptology – CRYPTO'91*, volume 576 of *Lecture Notes in Computer Science*, pages 445–456, Santa Barbara, CA, USA, August 11–15, 1992. Springer, Heidelberg, Germany.

[DDF14] Alexandre Duc, Stefan Dziembowski, and Sebastian Faust. Unifying leakage models: From probing attacks to noisy leakage. In Phong Q. Nguyen and Elisabeth Oswald, editors, *Advances in Cryptology – EUROCRYPT 2014*, volume 8441 of *Lecture Notes in Computer Science*, pages 423–440, Copenhagen, Denmark, May 11–15, 2014. Springer, Heidelberg, Germany.

[DDN91] Danny Dolev, Cynthia Dwork, and Moni Naor. Non-malleable cryptography (extended abstract). In *23rd Annual ACM Symposium on Theory of Computing*, pages 542–552, New Orleans, LA, USA, May 6–8, 1991. ACM Press.

[DDN98] Danny Dolev, Cynthia Dwork, and Moni Naor. Non-malleable cryptography, 1998. Manuscript.

[DDO+01] Alfredo De Santis, Giovanni Di Crescenzo, Rafail Ostrovsky, Giuseppe Persiano, and Amit Sahai. Robust non-interactive zero knowledge. In Joe Kilian, editor, *Advances in Cryptology – CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 566–598, Santa Barbara, CA, USA, August 19–23, 2001. Springer, Heidelberg, Germany.

[DDV10] Francesco Davì, Stefan Dziembowski, and Daniele Venturi. Leakage-resilient storage. In Juan A. Garay and Roberto De Prisco, editors, *SCN 10: 7th International Conference on Security in Communication Networks*, volume 6280 of *Lecture Notes in Computer Science*, pages 121–137, Amalfi, Italy, September 13–15, 2010. Springer, Heidelberg, Germany.

[DF11] Stefan Dziembowski and Sebastian Faust. Leakage-resilient cryptography from the inner-product extractor. In Dong Hoon Lee and Xiaoyun Wang, editors, *Advances in Cryptology – ASIACRYPT 2011*, volume 7073 of *Lecture Notes in Computer Science*, pages 702–721, Seoul, South Korea, December 4–8, 2011. Springer, Heidelberg, Germany.

[DFS16] Stefan Dziembowski, Sebastian Faust, and François-Xavier Standaert. Private circuits III: hardware trojan-resilience via testing amplification. In *ACM CCS*, pages 142–153, 2016.

[DG03] Ivan Damgård and Jens Groth. Non-interactive and reusable non-malleable commitment schemes. In *35th Annual ACM Symposium on Theory of Computing*, pages 426–437, San Diego, CA, USA, June 9–11, 2003. ACM Press.

[DGL+16] Dana Dachman-Soled, S. Dov Gordon, Feng-Hao Liu, Adam O'Neill, and Hong-Sheng Zhou. Leakage-resilient public-key encryption from obfuscation. In Chen-Mou Cheng, Kai-Min Chung, Giuseppe Persiano, and Bo-Yin Yang, editors, *PKC 2016: 19th International Conference on Theory and Practice of Public Key Cryptography, Part II*, volume 9615 of *Lecture Notes in Computer Science*, pages 101–128, Taipei, Taiwan, March 6–9, 2016. Springer, Heidelberg, Germany.

[DI05] Ivan Damgård and Yuval Ishai. Constant-round multiparty computation using a black-box pseudorandom generator. In *CRYPTO*, pages 378–394, 2005.

[DIO98] Giovanni Di Crescenzo, Yuval Ishai, and Rafail Ostrovsky. Non-interactive and non-malleable commitment. In *30th Annual ACM Symposium on Theory of Computing*, pages 141–150, Dallas, TX, USA, May 23–26, 1998. ACM Press.

[DK12] Dana Dachman-Soled and Yael Tauman Kalai. Securing circuits against constant-rate tampering. In Reihaneh Safavi-Naini and Ran Canetti, editors,

Advances in Cryptology – CRYPTO 2012, volume 7417 of *Lecture Notes in Computer Science*, pages 533–551, Santa Barbara, CA, USA, August 19–23, 2012. Springer, Heidelberg, Germany.

[DK14] Dana Dachman-Soled and Yael Tauman Kalai. Securing circuits and protocols against 1/poly(k) tampering rate. In Yehuda Lindell, editor, *TCC 2014: 11th Theory of Cryptography Conference*, volume 8349 of *Lecture Notes in Computer Science*, pages 540–565, San Diego, CA, USA, February 24–26, 2014. Springer, Heidelberg, Germany.

[DKO13] Stefan Dziembowski, Tomasz Kazana, and Maciej Obremski. Non-malleable codes from two-source extractors. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology – CRYPTO 2013, Part II*, volume 8043 of *Lecture Notes in Computer Science*, pages 239–257, Santa Barbara, CA, USA, August 18–22, 2013. Springer, Heidelberg, Germany.

[DKO+18] Ivan Damgrd, Tomasz Kazana, Maciej Obremski, Varun Raj, and Luisa Siniscalchi. Continuous nmc secure against permutations and overwrites, with applications to cca secure commitments. Cryptology ePrint Archive, Report 2018/596, 2018. `https://eprint.iacr.org/2018/596`.

[DKOS01] Giovanni Di Crescenzo, Jonathan Katz, Rafail Ostrovsky, and Adam Smith. Efficient and non-interactive non-malleable commitment. In Birgit Pfitzmann, editor, *Advances in Cryptology – EUROCRYPT 2001*, volume 2045 of *Lecture Notes in Computer Science*, pages 40–59, Innsbruck, Austria, May 6–10, 2001. Springer, Heidelberg, Germany.

[DKS17a] Dana Dachman-Soled, Mukul Kulkarni, and Aria Shahverdi. Locally decodable and updatable non-malleable codes in the bounded retrieval model. Cryptology ePrint Archive, Report 2017/303, 2017. `http://eprint.iacr.org/2017/303`.

[DKS17b] Dana Dachman-Soled, Mukul Kulkarni, and Aria Shahverdi. Tight upper and lower bounds for leakage-resilient, locally decodable and updatable non-malleable codes. In Serge Fehr, editor, *PKC 2017: 20th International Conference on Theory and Practice of Public Key Cryptography, Part I*, volume 10174 of *Lecture Notes in Computer Science*, pages 310–332, Amsterdam, The Netherlands, March 28–31, 2017. Springer, Heidelberg, Germany.

[DKS18] Dana Dachman-Soled, Mukul Kulkarni, and Aria Shahverdi. Local non-malleable codes in the bounded retrieval model. In Michel Abdalla and Ricardo Dahab, editors, *PKC 2018: 21st International Conference on Theory and Practice of Public Key Cryptography, Part II*, volume 10770 of *Lecture Notes in Computer Science*, pages 281–311, Rio de Janeiro, Brazil, March 25–29, 2018. Springer, Heidelberg, Germany.

[DLSZ15] Dana Dachman-Soled, Feng-Hao Liu, Elaine Shi, and Hong-Sheng Zhou. Locally decodable and updatable non-malleable codes and their applications. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015: 12th Theory of Cryptography Conference, Part I*, volume 9014 of *Lecture Notes in Computer Science*, pages 427–450, Warsaw, Poland, March 23–25, 2015. Springer, Heidelberg, Germany.

[DLZ15] Dana Dachman-Soled, Feng-Hao Liu, and Hong-Sheng Zhou. Leakage-resilient circuits revisited - optimal number of computing components without leak-free hardware. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology – EUROCRYPT 2015, Part II*, volume 9057 of *Lecture Notes in Computer Science*, pages 131–158, Sofia, Bulgaria, April 26–30, 2015. Springer, Heidelberg, Germany.

[DNO17] Nico Döttling, Jesper Buus Nielsen, and Maciej Obremski. Information theoretic continuously non-malleable codes in the constant split-state model. Cryptology ePrint Archive, Report 2017/357, 2017. `http://eprint.iacr.org/2017/357`.

[DP08] Stefan Dziembowski and Krzysztof Pietrzak. Leakage-resilient cryptography. In *49th Annual Symposium on Foundations of Computer Science*, pages 293–302, Philadelphia, PA, USA, October 25–28, 2008. IEEE Computer Society Press.

[DPW10] Stefan Dziembowski, Krzysztof Pietrzak, and Daniel Wichs. Non-malleable codes. In Andrew Chi-Chih Yao, editor, *ICS 2010: 1st Innovations in Computer Science*, pages 434–452, Tsinghua University, Beijing, China, January 5–7, 2010. Tsinghua University Press.

[DRS04] Yevgeniy Dodis, Leonid Reyzin, and Adam Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. In Chris-

tian Cachin and Jan Camenisch, editors, *Advances in Cryptology – EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 523–540, Interlaken, Switzerland, May 2–6, 2004. Springer, Heidelberg, Germany.

[FHMV17] Sebastian Faust, Kristina Hostáková, Pratyay Mukherjee, and Daniele Venturi. Non-malleable codes for space-bounded tampering. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology – CRYPTO 2017, Part II*, volume 10402 of *Lecture Notes in Computer Science*, pages 95–126, Santa Barbara, CA, USA, August 20–24, 2017. Springer, Heidelberg, Germany.

[FKPR10] Sebastian Faust, Eike Kiltz, Krzysztof Pietrzak, and Guy N. Rothblum. Leakage-resilient signatures. In Daniele Micciancio, editor, *TCC 2010: 7th Theory of Cryptography Conference*, volume 5978 of *Lecture Notes in Computer Science*, pages 343–360, Zurich, Switzerland, February 9–11, 2010. Springer, Heidelberg, Germany.

[FMNV14] Sebastian Faust, Pratyay Mukherjee, Jesper Buus Nielsen, and Daniele Venturi. Continuous non-malleable codes. In Yehuda Lindell, editor, *TCC 2014: 11th Theory of Cryptography Conference*, volume 8349 of *Lecture Notes in Computer Science*, pages 465–488, San Diego, CA, USA, February 24–26, 2014. Springer, Heidelberg, Germany.

[FMNV15] Sebastian Faust, Pratyay Mukherjee, Jesper Buus Nielsen, and Daniele Venturi. A tamper and leakage resilient von neumann architecture. In Jonathan Katz, editor, *PKC 2015: 18th International Conference on Theory and Practice of Public Key Cryptography*, volume 9020 of *Lecture Notes in Computer Science*, pages 579–603, Gaithersburg, MD, USA, March 30 – April 1, 2015. Springer, Heidelberg, Germany.

[FMVW14] Sebastian Faust, Pratyay Mukherjee, Daniele Venturi, and Daniel Wichs. Efficient non-malleable codes and key-derivation for poly-size tampering circuits. In Phong Q. Nguyen and Elisabeth Oswald, editors, *Advances in Cryptology – EUROCRYPT 2014*, volume 8441 of *Lecture Notes in Computer Science*, pages 111–128, Copenhagen, Denmark, May 11–15, 2014. Springer, Heidelberg, Germany.

[FN17] Antonio Faonio and Jesper Buus Nielsen. Non-malleable codes with split-state refresh. In Serge Fehr, editor, *PKC 2017: 20th International Conference on Theory and Practice of Public Key Cryptography, Part I*, volume 10174 of *Lecture Notes in Computer Science*, pages 279–309, Amsterdam, The Netherlands, March 28–31, 2017. Springer, Heidelberg, Germany.

[FPV11] Sebastian Faust, Krzysztof Pietrzak, and Daniele Venturi. Tamper-proof circuits: How to trade leakage for tamper-resilience. In Luca Aceto, Monika Henzinger, and Jiri Sgall, editors, *ICALP 2011: 38th International Colloquium on Automata, Languages and Programming, Part I*, volume 6755 of *Lecture Notes in Computer Science*, pages 391–402, Zurich, Switzerland, July 4–8, 2011. Springer, Heidelberg, Germany.

[FRR+10] Sebastian Faust, Tal Rabin, Leonid Reyzin, Eran Tromer, and Vinod Vaikuntanathan. Protecting circuits from leakage: the computationally-bounded and noisy cases. In Henri Gilbert, editor, *Advances in Cryptology – EUROCRYPT 2010*, volume 6110 of *Lecture Notes in Computer Science*, pages 135–156, French Riviera, May 30 – June 3, 2010. Springer, Heidelberg, Germany.

[GA03] Sudhakar Govindavajhala and Andrew W. Appel. Using memory errors to attack a virtual machine. In *2003 IEEE Symposium on Security and Privacy*, pages 154–165, Berkeley, CA, USA, May 11–14, 2003. IEEE Computer Society Press.

[GGP10] Rosario Gennaro, Craig Gentry, and Bryan Parno. Non-interactive verifiable computing: Outsourcing computation to untrusted workers. In *CRYPTO*, pages 465–482, 2010.

[GGPR13] Rosario Gennaro, Craig Gentry, Bryan Parno, and Mariana Raykova. Quadratic span programs and succinct NIZKs without PCPs. In Thomas Johansson and Phong Q. Nguyen, editors, *Advances in Cryptology – EUROCRYPT 2013*, volume 7881 of *Lecture Notes in Computer Science*, pages 626–645, Athens, Greece, May 26–30, 2013. Springer, Heidelberg, Germany.

[GIP+14] Daniel Genkin, Yuval Ishai, Manoj Prabhakaran, Amit Sahai, and Eran Tromer. Circuits resilient to additive attacks with applications to secure computation. In David B. Shmoys, editor, *46th Annual ACM Symposium on*

Theory of Computing, pages 495–504, New York, NY, USA, May 31 – June 3, 2014. ACM Press.

[GKR10] Rosario Gennaro, Hugo Krawczyk, and Tal Rabin. Okamoto-Tanaka revisited: Fully authenticated Diffie-Hellman with minimal overhead. In Jianying Zhou and Moti Yung, editors, *ACNS 10: 8th International Conference on Applied Cryptography and Network Security*, volume 6123 of *Lecture Notes in Computer Science*, pages 309–328, Beijing, China, June 22–25, 2010. Springer, Heidelberg, Germany.

[GLM+04] Rosario Gennaro, Anna Lysyanskaya, Tal Malkin, Silvio Micali, and Tal Rabin. Algorithmic tamper-proof (ATP) security: Theoretical foundations for security against hardware tampering. In Moni Naor, editor, *TCC 2004: 1st Theory of Cryptography Conference*, volume 2951 of *Lecture Notes in Computer Science*, pages 258–277, Cambridge, MA, USA, February 19–21, 2004. Springer, Heidelberg, Germany.

[GLR11] Shafi Goldwasser, Huijia Lin, and Aviad Rubinstein. Delegation of computation without rejection problem from designated verifier CS-Proofs. Cryptology ePrint Archive, Report 2011/456, 2011. `http://eprint.iacr.org/2011/456`.

[GPR16] Vipul Goyal, Omkant Pandey, and Silas Richelson. Textbook non-malleable commitments. In Daniel Wichs and Yishay Mansour, editors, *48th Annual ACM Symposium on Theory of Computing*, pages 1128–1141, Cambridge, MA, USA, June 18–21, 2016. ACM Press.

[Gro10] Jens Groth. Short pairing-based non-interactive zero-knowledge arguments. In Masayuki Abe, editor, *Advances in Cryptology – ASIACRYPT 2010*, volume 6477 of *Lecture Notes in Computer Science*, pages 321–340, Singapore, December 5–9, 2010. Springer, Heidelberg, Germany.

[GS08] Jens Groth and Amit Sahai. Efficient non-interactive proof systems for bilinear groups. In Nigel P. Smart, editor, *Advances in Cryptology – EUROCRYPT 2008*, volume 4965 of *Lecture Notes in Computer Science*, pages 415–432, Istanbul, Turkey, April 13–17, 2008. Springer, Heidelberg, Germany.

[GST14] Daniel Genkin, Adi Shamir, and Eran Tromer. RSA key extraction via low-bandwidth acoustic cryptanalysis. In Juan A. Garay and Rosario Gennaro,

editors, *Advances in Cryptology – CRYPTO 2014, Part I*, volume 8616 of *Lecture Notes in Computer Science*, pages 444–461, Santa Barbara, CA, USA, August 17–21, 2014. Springer, Heidelberg, Germany.

[GST17] Daniel Genkin, Adi Shamir, and Eran Tromer. Acoustic cryptanalysis. *Journal of Cryptology*, 30(2):392–443, April 2017.

[GW11] Craig Gentry and Daniel Wichs. Separating succinct non-interactive arguments from all falsifiable assumptions. In Lance Fortnow and Salil P. Vadhan, editors, *43rd Annual ACM Symposium on Theory of Computing*, pages 99–108, San Jose, CA, USA, June 6–8, 2011. ACM Press.

[HILL99] Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM Journal on Computing*, 28(4):1364–1396, 1999.

[HT98] Satoshi Hada and Toshiaki Tanaka. On the existence of 3-round zero-knowledge protocols. In Hugo Krawczyk, editor, *Advances in Cryptology – CRYPTO'98*, volume 1462 of *Lecture Notes in Computer Science*, pages 408–423, Santa Barbara, CA, USA, August 23–27, 1998. Springer, Heidelberg, Germany.

[IEGT13] Frank Imeson, Ariq Emtenan, Siddharth Garg, and Mahesh V. Tripunitara. Securing computer hardware using 3d integrated circuit (IC) technology and split manufacturing for obfuscation. In *USENIX Security Symposium*, pages 495–510, 2013.

[IOZ14] Yuval Ishai, Rafail Ostrovsky, and Vassilis Zikas. Secure multi-party computation with identifiable abort. In *CRYPTO*, pages 369–386, 2014.

[IPSW06] Yuval Ishai, Manoj Prabhakaran, Amit Sahai, and David Wagner. Private circuits II: Keeping secrets in tamperable circuits. In Serge Vaudenay, editor, *Advances in Cryptology – EUROCRYPT 2006*, volume 4004 of *Lecture Notes in Computer Science*, pages 308–327, St. Petersburg, Russia, May 28 – June 1, 2006. Springer, Heidelberg, Germany.

[ISW03] Yuval Ishai, Amit Sahai, and David Wagner. Private circuits: Securing hardware against probing attacks. In Dan Boneh, editor, *Advances in Cryptology – CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages

463–481, Santa Barbara, CA, USA, August 17–21, 2003. Springer, Heidelberg, Germany.

[JW15] Zahra Jafargholi and Daniel Wichs. Tamper detection and continuous non-malleable codes. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015: 12th Theory of Cryptography Conference, Part I*, volume 9014 of *Lecture Notes in Computer Science*, pages 451–480, Warsaw, Poland, March 23–25, 2015. Springer, Heidelberg, Germany.

[KJJ99] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In Michael J. Wiener, editor, *Advances in Cryptology – CRYPTO'99*, volume 1666 of *Lecture Notes in Computer Science*, pages 388–397, Santa Barbara, CA, USA, August 15–19, 1999. Springer, Heidelberg, Germany.

[KKKT16] Aggelos Kiayias, Elias Koutsoupias, Maria Kyropoulou, and Yiannis Tselekounis. Blockchain mining games. In *Proceedings of the 2016 ACM Conference on Economics and Computation*, EC '16, pages 365–382, New York, NY, USA, 2016. ACM.

[KL14] Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography, Second Edition*. Chapman & Hall/CRC, 2nd edition, 2014.

[KLT16] Aggelos Kiayias, Feng-Hao Liu, and Yiannis Tselekounis. Practical non-malleable codes from l-more extractable hash functions. In Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi, editors, *ACM CCS 16: 23rd Conference on Computer and Communications Security*, pages 1317–1328, Vienna, Austria, October 24–28, 2016. ACM Press.

[KLT18a] Aggelos Kiayias, Feng-Hao Liu, and Yiannis Tselekounis. Non-malleable codes for partial functions with manipulation detection. In Hovav Shacham and Alexandra Boldyreva, editors, *Advances in Cryptology – CRYPTO 2018*, pages 577–607, Cham, 2018. Springer International Publishing.

[KLT18b] Aggelos Kiayias, Feng-Hao Liu, and Yiannis Tselekounis. Non-malleable codes for partial functions with manipulation detection. (under submission) 2018.

[Koc96] Paul C. Kocher. Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In Neal Koblitz, editor, *Advances in Cryptology – CRYPTO'96*, volume 1109 of *Lecture Notes in Computer Science*, pages 104–113, Santa Barbara, CA, USA, August 18–22, 1996. Springer, Heidelberg, Germany.

[KT13] Aggelos Kiayias and Yiannis Tselekounis. Tamper resilient circuits: The adversary at the gates. In Kazue Sako and Palash Sarkar, editors, *Advances in Cryptology – ASIACRYPT 2013, Part II*, volume 8270 of *Lecture Notes in Computer Science*, pages 161–180, Bengalore, India, December 1–5, 2013. Springer, Heidelberg, Germany.

[LJM11] Eric Love, Yier Jin, and Yiorgos Makris. Enhancing security via provably trustworthy hardware intellectual property. In *IEEE HOST*, pages 12–17, 2011.

[LKG+09] Lang Lin, Markus Kasper, Tim Güneysu, Christof Paar, and Wayne Burleson. Trojan side-channels: Lightweight hardware trojans through side-channel engineering. In *CHES*, pages 382–395, 2009.

[LL12] Feng-Hao Liu and Anna Lysyanskaya. Tamper and leakage resilience in the split-state model. In Reihaneh Safavi-Naini and Ran Canetti, editors, *Advances in Cryptology – CRYPTO 2012*, volume 7417 of *Lecture Notes in Computer Science*, pages 517–532, Santa Barbara, CA, USA, August 19–23, 2012. Springer, Heidelberg, Germany.

[LP11] Huijia Lin and Rafael Pass. Constant-round non-malleable commitments from any one-way function. In Lance Fortnow and Salil P. Vadhan, editors, *43rd Annual ACM Symposium on Theory of Computing*, pages 705–714, San Jose, CA, USA, June 6–8, 2011. ACM Press.

[LPS17] Huijia Lin, Rafael Pass, and Pratik Soni. Two-round and non-interactive concurrent non-malleable commitments from time-lock puzzles. In *58th Annual Symposium on Foundations of Computer Science*, pages 576–587. IEEE Computer Society Press, 2017.

[Mak15] Marie A. Mak. Trusted Defense Microelectronics: Future Access and Capabilities Are Uncertain. Technical report, United States Government Accountability Office, 10 2015.

[MCS+17] Vasilios Mavroudis, Andrea Cerulli, Petr Svenda, Dan Cvrcek, Dusan Klinec, and George Danezis. A touch of evil: High-assurance cryptographic hardware from untrusted components. In *ACM CCS*, pages 1583–1600, 2017.

[MR04] Silvio Micali and Leonid Reyzin. Physically observable cryptography (extended abstract). In Moni Naor, editor, *TCC 2004: 1st Theory of Cryptography Conference*, volume 2951 of *Lecture Notes in Computer Science*, pages 278–296, Cambridge, MA, USA, February 19–21, 2004. Springer, Heidelberg, Germany.

[MSD16] Amir S. Mortazavia, Mahmoud Salmasizadeh, and Amir Daneshgar. FMNV continuous non-malleable encoding scheme is more efficient than believed. Cryptology ePrint Archive, Report 2016/604, 2016. `http://eprint.iacr.org/2016/604`.

[MWPB09] David R. McIntyre, Francis G. Wolff, Christos A. Papachristou, and Swarup Bhunia. Dynamic evaluation of hardware trust. In *IEEE HOST*, pages 108–111, 2009.

[Nao03] Moni Naor. On cryptographic assumptions and challenges (invited talk). In Dan Boneh, editor, *Advances in Cryptology – CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 96–109, Santa Barbara, CA, USA, August 17–21, 2003. Springer, Heidelberg, Germany.

[NS09] Moni Naor and Gil Segev. Public-key cryptosystems resilient to key leakage. In Shai Halevi, editor, *Advances in Cryptology – CRYPTO 2009*, volume 5677 of *Lecture Notes in Computer Science*, pages 18–35, Santa Barbara, CA, USA, August 16–20, 2009. Springer, Heidelberg, Germany.

[NZ93] Noam Nisan and David Zuckerman. Randomness is linear in space. *Journal of Computer and System Sciences*, pages 43–52, 1993.

[Oka88] Eiji Okamoto. Key distribution systems based on identification information. In Carl Pomerance, editor, *Advances in Cryptology – CRYPTO'87*, volume 293 of *Lecture Notes in Computer Science*, pages 194–202, Santa Barbara, CA, USA, August 16–20, 1988. Springer, Heidelberg, Germany.

[OPVV18] Rafail Ostrovsky, Giuseppe Persiano, Daniele Venturi, and Ivan Visconti. Continuously non-malleable codes in the split-state model from minimal as-

sumptions. Cryptology ePrint Archive, Report 2018/542, 2018. https://eprint.iacr.org/2018/542.

[OW] L. H. Ozarow and A. D. Wyner. Wire-tap channel ii. *AT T Bell Laboratories Technical Journal*.

[Pas13] Rafael Pass. Unprovable security of perfect NIZK and non-interactive non-malleable commitments. In Amit Sahai, editor, *TCC 2013: 10th Theory of Cryptography Conference*, volume 7785 of *Lecture Notes in Computer Science*, pages 334–354, Tokyo, Japan, March 3–6, 2013. Springer, Heidelberg, Germany.

[PHGR13] Bryan Parno, Jon Howell, Craig Gentry, and Mariana Raykova. Pinocchio: Nearly practical verifiable computation. In *2013 IEEE Symposium on Security and Privacy*, pages 238–252, Berkeley, CA, USA, May 19–22, 2013. IEEE Computer Society Press.

[Pie09] Krzysztof Pietrzak. A leakage-resilient mode of operation. In Antoine Joux, editor, *Advances in Cryptology – EUROCRYPT 2009*, volume 5479 of *Lecture Notes in Computer Science*, pages 462–482, Cologne, Germany, April 26–30, 2009. Springer, Heidelberg, Germany.

[Pot10] Miodrag Potkonjak. Synthesis of trustable ics using untrusted CAD tools. In *DAC*, pages 633–634, 2010.

[PR05] Rafael Pass and Alon Rosen. New and improved constructions of non-malleable cryptographic protocols. In Harold N. Gabow and Ronald Fagin, editors, *37th Annual ACM Symposium on Theory of Computing*, pages 533–542, Baltimore, MA, USA, May 22–24, 2005. ACM Press.

[PX09] Manoj Prabhakaran and Rui Xue. Statistically hiding sets. In Marc Fischlin, editor, *Topics in Cryptology – CT-RSA 2009*, volume 5473 of *Lecture Notes in Computer Science*, pages 100–116, San Francisco, CA, USA, April 20–24, 2009. Springer, Heidelberg, Germany.

[Riv97] Ronald L. Rivest. All-or-nothing encryption and the package transform. In Eli Biham, editor, *Fast Software Encryption – FSE'97*, volume 1267 of *Lecture Notes in Computer Science*, pages 210–218, Haifa, Israel, January 20–22, 1997. Springer, Heidelberg, Germany.

[RP11] Jason K. Resch and James S. Plank. Aont-rs: Blending security and performance in dispersed storage systems. FAST'11, 2011.

[SA03] Sergei P. Skorobogatov and Ross J. Anderson. Optical fault induction attacks. In Burton S. Kaliski Jr., Çetin Kaya Koç, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems – CHES 2002*, volume 2523 of *Lecture Notes in Computer Science*, pages 2–12, Redwood Shores, CA, USA, August 13–15, 2003. Springer, Heidelberg, Germany.

[Sah99] Amit Sahai. Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In *40th Annual Symposium on Foundations of Computer Science*, pages 543–553, New York, NY, USA, October 17–19, 1999. IEEE Computer Society Press.

[SB15] Jean-Pierre Seifert and Christoph Bayer. Trojan-resilient circuits. In Al-Sakib Khan Pathan, editor, *Securing Cyber-Physical Systems*, chapter 14, pages 349–370. CRC Press, Boca Raton, London, New York, 2015.

[Sha07] Brian Sharkey. Trust in Integrated Circuits Program. Technical report, DARPA, 03 2007.

[SPY+10] François-Xavier Standaert, Olivier Pereira, Yu Yu, Jean-Jacques Quisquater, Moti Yung, and Elisabeth Oswald. Leakage resilient cryptography in practice. Information Security and Cryptography, pages 99–134. Springer, Heidelberg, Germany, 2010.

[SS16] Ronen Shaltiel and Jad Silbak. Explicit List-Decodable Codes with Optimal Rate for Computationally Bounded Channels. In *APPROX/RANDOM 2016*, 2016.

[Sti01] D. R. Stinson. Something about all or nothing (transforms). *Designs, Codes and Cryptography*, 22(2):133–138, Mar 2001.

[TMA11] Michael Tunstall, Debdeep Mukhopadhyay, and Subidh Ali. *Differential Fault Analysis of the Advanced Encryption Standard Using a Single Fault*, pages 224–233. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.

[WHG+16] Riad S. Wahby, Max Howald, Siddharth J. Garg, Abhi Shelat, and Michael Walfish. Verifiable asics. In *IEEE S&P*, pages 759–778, 2016.

[WRK17] Xiao Wang, Samuel Ranellucci, and Jonathan Katz. Global-scale secure multiparty computation. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *ACM CCS 17: 24th Conference on Computer and Communications Security*, pages 39–56, Dallas, TX, USA, October 31 – November 2, 2017. ACM Press.

[WS11] Adam Waksman and Simha Sethumadhavan. Silencing hardware backdoors. In *IEEE Symposium on Security and Privacy*, pages 49–63, 2011.

[Wyn75] A. D. Wyner. The wire-tap channel. *The Bell System Technical Journal*, 1975.