# Efficient Cryptographic Constructions with Strong Security Guarantees

Rafael Kurek

University of Wuppertal

Dissertation

submitted to the School of Electrical, Information and Media Engineering
in accordance with the requirements for award of the degree

Doktor der Ingenieurwissenschaften (Dr.-Ing.)

September 2020

The PhD thesis can be quoted as follows:

urn:nbn:de:hbz:468-20210126-102205-4
[http://nbn-resolving.de/urn/resolver.pl?urn=urn%3Anbn%3Ade%3Ahbz%3A468-20210126-102205-4]

DOI: 10.25926/5k9s-v129
[https://doi.org/10.25926/5k9s-v129]

## Erklärung zur Disseration

Hiermit erkläre ich, Rafael Kurek, geboren am 18.11.1989 in Gelsenkirchen,

1. dass ich die eingereichte Arbeit selbstständig verfasst habe;
2. dass ich bei der Abfassung der Arbeit nur die in der Dissertation angegebenen Hilfsmittel benutzt und alle wörtlich oder inhaltlich übernommenen Stellen als solche gekennzeichnet habe;
3. dass ich die Dissertation in der gegenwärtigen oder einer anderen Fassung nicht schon einem anderen Fachbereich einer wissenschaftlichen Hochschule vorgelegen habe.

Ort, Datum

Unterschrift

## Overview of Publications

**Short Digital Signatures and ID-KEMs via Truncation Collision Resistance.** Tibor Jager and Rafael Kurek. In Proceedings of the International Conference on the Theory and Application of Cryptology and Information Security - ASIACRYPT 2018.

**Simple and More Efficient PRFs with Tight Security from LWE and Matrix-DDH.** Tibor Jager, Rafael Kurek, and Jiaxin Pan. In Proceedings of the International Conference on the Theory and Application of Cryptology and Information Security - ASIACRYPT 2018.

**Efficient Forward-Secure Threshold Signatures.** Rafael Kurek. In Proceedings of the International Workshop on Security - IWSEC 2020.

**Efficient Forward-Secure Threshold Public Key Encryption.** Rafael Kurek. In Proceedings of the Australasian Conference on Information Security and Privacy - ACISP 2020.

**Efficient Adaptively-Secure IB-KEMs and VRFs via Near-Collision Resistance.** Tibor Jager, Rafael Kurek, and David Niehues. Preprint.

# Acknowledgements

First and foremost, I would like to thank my supervisor Tibor Jager for introducing me into research and for teaching me so much about cryptography and IT security. I am also very thankful for letting me travel to so many conferences, workshops and in particular to the University of Maryland, College Park. These travels were always very inspiring to me and allowed me to get to know a great and unique research community.
My gratitude goes also to Jiaxin Pan for accepting to co-referee this dissertation. I would also like to thank Tibor Jager, Jiaxin Pan, and David Niehues for being such good coauthors. It was always more fun and more fruitful to work with you than on my own.
I extend my gratitude to Jonathan Katz for hosting me at the University of Maryland, College Park. During this visit, I had a very wonderful and enlightening time.
I would also like to thank Gareth Davies for reading lots of my work and sharing his experience on writing research papers. Thanks to Peter Chvojka and Pascal Bemmann with whom I had the pleasure to travel to Australia for Asiacrypt 2018. This journey would not have made so much fun without you. Also big thanks to all the other members of the ITSC research group in Wuppertal: Saqib Kakvi, Lin Lyu, Denis Diemert, Jan Drees, Kai Gellert, Tobias Handirk, and Jutta Maerten. Thank you for all the great moments we could share.
In the end, I would like to thank my family and my friends outside the university for their understanding over the years and supporting me in everything I decided to do.
Last but not least, I am very grateful that I met Marie while studying in Bochum. Thank you so much for supporting, understanding, and believing in me. Thank you for all the wonderful and great moments we could share.

# Contents

CHAPTER 1

# Introduction

Digital devices, systems, and networks effect more and more parts of everyday life. However, due to the complexity of modern digital technology a variety of techniques and mechanisms are required in order to secure communication and to protect sensitive data. The science dealing with secure communication and data protection is called cryptography.

**Proving security.** In classical cryptography, the approach to build cryptographic schemes can be described as "make it and break it". Schemes were supposed to be secure if they were able to withstand all known attacks. However, this approach did not take into account future attacks and therefore the security guarantees given by classical cryptography were rather limited. The groundbreaking work of Goldwasser and Micali [GM84], which introduced the idea of using rigorous and formal security proofs, led to a paradigm shift in constructing cryptographic schemes and to what is known as modern cryptography. Instead of testing constructions against all known attacks, modern cryptography attempts to provide *security proofs in well-defined models*. In order to guarantee any reasonable security properties, such models need to reflect the real world very accurately. Accordingly, they have to take into account as many conceivable threats and violations of the desired security guarantees as possible. A security proof is designed as an experiment between a challenger and an adversary, who attempts to break the cryptographic scheme. The design of the experiment determines the *security notion*, which covers a *security guarantee* and a *threat model* [KL14]. The security guarantee defines what an adversary should not be able to learn or to do. For digital signature schemes, such a guarantee could be that an adversary should not be able to forge a signature or to extract the secret signing key from the information given to it. The threat model describes the abilities and the strength of the adversary. For digital signature schemes, the adversary might be allowed to query the signatures for arbitrary messages. There are different ways to allow these queries to the adversary. For instance, the adversary could be allowed to make these queries only *selectively*, which means prior to the first output of the challenger. Another option would be to allow to make these queries *adaptively*. In the case of adaptive queries, the adversary can make one query after another and adapt its queries depending on the responses to the previous queries. Since an adaptive adversaries is able to adjust its behavior to the information it gained so far it has more power. Obviously, a threat model that allows adaptive queries is a stronger model and more realistic when dealing with adversarial behavior in the real world. We say that a cryptographic scheme that tolerates adaptive queries provides *strong security guarantees* compared to schemes that tolerate only non-adaptive queries.

**Real-world attacks.** In security proofs the adversary is usually treated blackbox, which means that on the one hand it is unknown how the adversary behaves but on the other hand any behavior within the rules dictated by the threat model is allowed. However, the crucial point about these rules is that they might restrict the adversary to a behavior that deviates from its behavior in the real world. Indeed, a lot of cryptographic schemes which were proven secure were successfully attacked in practice because the attacks were not covered in the threat model. One famous example for such an attack is the so-called Logjam Attack [ABD$^+$15]. The Logjam Attack makes use of precomputation in order to successfully attack the Diffie-Hellman key exchange mechanism [DH76] which is the main key exchange mechanism in TLS and known to be secure in "standard" threat models. Another famous example is the so-called Bleichenbacher Attack of RSA-OAEP[BR95], which describes an *adaptive* chosen ciphertext attack against the RSA Encryption Standard PKCS#1 [Ble98]. This attack makes use of an oracle that gives information about the validity of a ciphertext. By doing so it leaks partial information about the encrypted message and, when queried adaptively, it reveals the entire message. Although there exist several security proofs for RSA-OAEP [FOPS01, Sho01, KOS10], none of them provide a threat model that matches the real-world requirements for the version standardized in PKCS#1.

**Efficiency.** It is desirable to construct cryptographic schemes that can be proven secure within a model that deals with arbitrary adversarial behavior and therefore provide *strong security guarantees* in the real world. However, strong security guarantees are often at the expense of the *efficiency* of a construction. For digital signature schemes, this fact can be illustrated by a comparison between selectively-secure and adaptively-secure digital signature schemes. The selectively-secure digital signature scheme introduced in [BB04c] is much more efficient than conceptual similar digital signature schemes with adaptive security [Wat05, HJK11, BHJ$^+$13]. The public key in the scheme from [BB04c] as well as its signatures consist of much fewer elements and also the underlying groups enable faster computation due to smaller size. Designing efficient cryptographic schemes with strong security guarantees is not only challenging in terms of digital signatures. Similar challenges occur for public key encryption schemes and other cryptographic schemes as well; see [BB04a, Wat05, Wat09, Lew12, CLL$^+$13, AHY15].

**Overview of this thesis.** This thesis bridges the gap between cryptographic constructions with *strong security guarantees* and *efficient* execution. Here, strong security mainly refers to security against adaptive adversaries. For instance, this may refer to several kinds of queries which depend on the respond to previous queries as well as corrupting machines depending on prior corruptions. Efficiency refers to improvements in speed and savings in space. Those are achieved by lower computational costs, fewer communication rounds or smaller elements like keys, ciphertexts, and digital signatures. The thesis is composed of a preliminary chapter and three main parts:

**Chapter 2.** This chapter introduces some basic notation and conventions and recalls basic cryptographic primitives and bilinear pairings.

**Chapter 3.** A pseudorandom function (PRF) is a function that is computationally indistinguishable from a real random function. Due to the fact that a huge amount of applications requires "randomness", PRFs are a very important and founda-

tional cryptographic primitives. Moreover, they can be used to obtain simple and efficient constructions of message authentication codes, symmetric encryption, key derivation algorithms, and form useful building blocks for many other primitives; see [GGM84, BG90, Gol01, Bel06, Kra10].
Chapter 3 introduces efficient and tightly-secure pseudorandom functions (PRFs). These PRFs are secure against adaptive adversaries and have only a logarithmic loss in the security proof, and short secret keys. These PRFs are very simple and efficient variants of well-known constructions, including those of Naor-Reingold [NR97] and Lewko-Waters [LW09]. This chapter also introduces the currently most efficient LWE-based PRF from a weak LWE-assumption. This construction is a variant of the PRF of Banerjee et al. [BPR12] but with a much smaller modulus than the original construction.

Technically, this chapter introduces all-prefix universal hash functions (APUHFs), which are hash functions that are (almost-)universal, even if any prefix of the output is considered. Besides a simple and very efficient construction of APUHFs, it is also presented how APUHFs can be combined with the augmented cascade of Boneh et al. [BMR10].
The results in this chapter are based on collaborations with Tibor Jager and Jiaxin Pan. The notion of all-prefix universality and the augmented cascade with encoded input are mainly due to Tibor Jager and myself. The applications are mainly due to Jiaxin Pan. The results appeared at Asiacrypt 2018 [JKP18].

**Chapter 4.** In order to mitigate the damage due to secret key exposure there exist different approaches. Two of these approaches are *forward-secure* schemes and *threshold* schemes. Forward-secure schemes allow to evolve the secret key in regular time periods, while the public key remains fixed. Thus, every adversary with an outdated secret key cannot forge signatures or decrypt ciphers for time periods in the past. In an $(n, k)$-*threshold* scheme the secret key is distributed among $n$ shares and it requires the presence of at least $k + 1$ key shares to restore the secret key, whereas any subset of $k$ key shares is insufficient. Due to the fact that forward security and thresholds improve security guarantees against secret key exposure in a different manner, their combination to *forward-secure thresholds (FST)* can even reinforce these guarantees.
Chapter 4 introduces a forward-secure threshold signature scheme and a forward-secure threshold encryption scheme. Both are based on bilinear pairings with groups of prime order. Compared to existing schemes, the ones proposed here are much more efficient since they have a non-interactive key update procedure and also a non-interactive signing procedure and decryption procedure, respectively. Additionally, both schemes do not require a trusted dealer and have optimal resilience as well as small signatures and small ciphertexts, respectively. Both schemes are proven secure against adaptive adversaries and robust against malicious adversaries. The signature and the encryption scheme are the first schemes which achieve all of these properties and that can also be implemented on standardized elliptic curves.
The results in this chapter are my own work and are based on [Kur20a, Kur20b]. The articles are going to appear at ACISP 2020 and IWSEC 2020, respectively.

**Chapter 5.** In the random oracle model (ROM) [BR93a] a cryptographic hash function is modeled as an oracle that implements a truly random function. This approach provides a very powerful tool to prove security of cryptosystems. For example, it enables to adaptively “program” a hash function to map certain input values to specific output values in the security proof. However, the random oracle is a hypothetical concept and in practice it requires implementation with a standard cryptographic hash function, like SHA-3. Hence, the security guarantees for the real world are restricted.

Chapter 5 presents adaptively-secure variants of the selectively-secure pairing-based IB-KEM and digital signature scheme of Boneh and Boyen [BB04a, BB04c]. Both are proven secure in the standard model, based on q-type assumptions, and have public key size $O(\lambda)$, where $\lambda$ denotes the security parameter. The IB-KEM and the digital signature scheme have only a single group element as ciphertext and signature, respectively. Moreover, the security reductions are quadratically tighter than in the corresponding schemes by Jager and mysself [JK18, Asiacrypt 2018]. As a technical contribution blockwise partitioning is introduced. It leverages the assumption that a cryptographic hash function is Near-Collision Resistant to prove full adaptive security of cryptosystems.

The results in this chapter are based on collaborations with Tibor Jager and David Niehues. The concept of Near-Collision Resistance is due to David Niehues and inspired by Truncation Collision Resistance from Tibor Jager and myself [JK18, Asiacrypt 2018]. The construction of IB-KEM and digital signatures are mainly due to myself. The results are part of an ongoing submission.

CHAPTER 2

# Preliminaries

We start with introducing some basic notation and conventions in Section 2.1. In Section 2.2, we recall basic cryptographic primitives which are used in the subsequent chapters of this thesis. Additionally, we introduce the concept of bilinear pairings in Section 2.3.

## 2.1 Notation

Let $\lambda \in \mathbb{N}$ denote a security parameter. All our results are in the asymptotic setting, that is, we view all expressions involving $\lambda$ as functions in $\lambda$. This includes the running time $t_{\mathcal{A}} = t_{\mathcal{A}}(\lambda)$ and success probability $\varepsilon_{\mathcal{A}} = \varepsilon_{\mathcal{A}}(\lambda)$ of adversaries. Similarly, all algorithms implicitly receive the security parameter $1^\lambda$ as their first input. We call an algorithm *efficient*, if it runs in (probabilistic) polynomial time in $\lambda$.
For a finite set $A$, we write $a \leftarrow A$ to denote the action of sampling a uniformly random element $a$ from $A$. If $A$ is a probabilistic algorithm, then $a \leftarrow A(x)$ denotes the action of running $A(x)$ on input $x$ with uniform coins and output $a$.
For a bit string $a = (a_1, \ldots, a_n) \in \{0,1\}^n$ and $v, w \in \mathbb{N}$ with $v \leq w$, we write $a_{v:w}$ to denote the substring $(a_v, \ldots, a_w)$ of $a$ and $a_w := a_{1:w}$ . The set $\{0,1\}^*$ denotes the set of all possible bit strings.
For a multiplicative group $\mathbb{G}_X$ with random generator $[1]_X$ and prime order $q$ we write $[a]_1$ shorthand for $[1]_X^a$. More generally, for a matrix $\mathbf{A} = (a_{ij}) \in \mathbb{Z}_q^{n \times m}$, we define $[\mathbf{A}]_1$ as the implicit representation of $\mathbf{A}$ in $\mathbb{G}_X$:

$$[\mathbf{A}]_X := \begin{pmatrix} [a_{11}]_X & \cdots & [a_{1m}]_X \\ \\ [a_{n1}]_X & \cdots & [a_{nm}]_X \end{pmatrix} \in \mathbb{G}_X^{n \times m}.$$

## 2.2 Basic Primitives

In this section, we recall the formal definitions of basic cryptographic primitives as well as their security notions.

**Hash Functions.** Let $\mathcal{H}$ be a family of hash functions $H : \{0,1\}^* \to \mathcal{Y}$, where $\mathcal{Y}$ is a finite set. Collision resistance is defined via the following game between an adversary $\mathcal{A}$ and a challenger. The challenger picks a hash function $H$ from $\mathcal{H}$ uniformly at random and sends a description of $H$ to $\mathcal{A}$. $\mathcal{A}$ outputs two bitstrings $x$ and $x'$.

**Definition 2.1.** Let $\mathcal{A}$ be an adversary. We say that it $(t_{\mathcal{A}}, \varepsilon_{\mathcal{A}})$-breaks the **Collision resistance** of $\mathcal{H}$ if it runs in time $t_{\mathcal{A}}$ and

$$\Pr[\mathcal{A}(H) \to (x, x') \text{ s.t. } H(x) = H(x') \wedge x \neq x')] \geq \varepsilon_{\mathcal{A}}.$$

**Pseudorandom Functions.** Let $\mathcal{K}, \mathcal{D}$ be sets such that there is an efficient algorithm that samples uniformly random elements $k \leftarrow \mathcal{K}$. Let $F : \mathcal{K} \times \mathcal{D} \to \mathcal{G}$ be an efficiently computable function. We define the PRF experiment between a challenger and an adversary $\mathcal{A}$ as follows. Initially, the challenger generates a random key $k \leftarrow \mathcal{K}$ and tosses a coin $b \leftarrow \{0, 1\}$. The challenger provides adversary $\mathcal{A}$ with the following oracles.

- $\underline{\textbf{Query}(x).}$ On input $x \in \mathcal{D}$ the challenger computes $F(k, x)$ if $b = 1$ and $R(x)$ if $b = 0$, where $R : \mathcal{D} \to \mathcal{G}$ is a random function. When the adversary terminates and outputs a bit $b'$, then the experiment outputs 1 if $b = b'$, and 0 otherwise.

- $\underline{\textbf{Guess}(b').}$ On input a bit $b'$ it outputs 1 if $b = b'$, else 0. The experiment terminates.

Let $x_1, \ldots, x_Q \in \mathcal{D}$ be the sequence of queries issued by $\mathcal{A}$ throughout the security experiment. We assume that we always have $Q \geq 1$, as otherwise the output of $\mathcal{A}$ is independent of $b$. Furthermore, we assume that $\mathcal{A}$ never issues the same query twice. More precisely, we assume $x_u \neq x_v$ for $u \neq v$. This is without loss of generality, since both $F(k, \cdot)$ and $R(\cdot)$ are deterministic functions.

**Definition 2.2.** Let $\mathcal{A}$ be an adversary. We say that it $(t_{\mathcal{A}}, \varepsilon_{\mathcal{A}}, Q)$-breaks the **pseudorandomness** of $F$, if it runs in time $t_{\mathcal{A}}$, issues $Q$ queries in the PRF security experiment, and

$$|\Pr[\textbf{Guess}(b') = 1] - 1/2| \geq \varepsilon_{\mathcal{A}}.$$

**Digital Signatures.** A widely used primitive are digital signature schemes, which belong to the class of public-key cryptography [DH76]. Their main goal is to provide authenticity and integrity. The following definitions are adapted from [GMR88].

**Definition 2.3.** A digital signature scheme $\Sigma$ is defined via the following algorithms:

- $\textbf{KeyGen}(1^\lambda) \to (vk, signk)$. The key generation algorithm outputs a pair of verification and signing key $(vk, signk)$.

- $\textbf{Sign}(signk, M) \to \sigma$. On input a message $M$ and a signing key $signk$, it outputs a signature $\sigma$.

- $\textbf{Verify}(vk, M, \sigma) \to b$, where $b \in \{0, 1\}$. On input a verification key $vk$, a message $M$, and a signature $\sigma$, it outputs either 0 or 1.

The correctness can be defined as usual: For all $(vk, signk)$ output by $\textbf{KeyGen}(1^\lambda)$ and all messages $M$ from the message space it holds that

$$\Pr[\textbf{Verify}(vk, M, \textbf{Sign}(signk, M)) = 1] = 1.$$

**Existential Unforgeability under a chosen message attack (EUF-CMA).** The EUF-CMA security game is defined via the following game between a challenger and an adversary $\mathcal{A}$. The challenger runs $\textbf{KeyGen}(1^\lambda)$ to obtain $(vk, signk)$ and forwards $vk$ to $\mathcal{A}$. Then, the adversary has access to the following oracles.

- **SignQuery**$(M)$. On input a message $M$ from the message space the challenger computes and outputs a signatures $\sigma \leftarrow \mathbf{Sign}(signk, M)$.
- **Finalize**$(\sigma^*, M^*)$. If $M$ has not been queried to **SignQuery** then it checks whether $\mathbf{Verify}(vk, M^*, \sigma^*) = 1$. If this is true it outputs 1, else 0. The experiment terminates.

**Definition 2.4.** Let $\mathcal{A}$ be an adversary playing the EUF-CMA security game for a signature scheme $\Sigma$. We say that it $(t_\mathcal{A}, \varepsilon_\mathcal{A})$-breaks the EUF-CMA security of $\Sigma$ if it runs in time $t_\mathcal{A}$, $(M, \sigma) \neq (M^*, \sigma^*)$, and

$$\Pr[\mathbf{Finalize}(\sigma^*, M^*) = 1] \geq \varepsilon_\mathcal{A}.$$

We will also make use of a slightly different security notion for signature schemes. It is weaker on the one hand, because the adversary is only allowed to make one signature query, but it is stronger on the other hand, because it is allowed to forge a signature for the same message it queried a signature. Only the signature must differ.

**Strong Existential Unforgeability under a one chosen message attack (sEUF-1CMA).** The sEUF-1CMA security game is defined via the following game between a challenger and an adversary $\mathcal{A}$: The challenger runs $\mathbf{KeyGen}(1^\lambda)$ to obtain $(vk, signk)$ and forwards $vk$ to adversary $\mathcal{A}$. Then the adversary is allowed to query both of the following oracles only once.

- **SignQuery**$(M)$. On input a message $M$ from the message space the challenger computes and outputs a signatures $\sigma \leftarrow \mathbf{Sign}(signk, M)$.
- **Finalize**$(\sigma^*, M^*)$. If the adversary queried a signature for $M^*$ beforehand and received $\sigma \leftarrow \sigma^*$ then the challenger returns 0. Else the challenger checks whether $\mathbf{Verify}(vk, M^*, \sigma^*) = 1$. If this is true it outputs 1, else 0. The experiment terminates.

**Definition 2.5.** Let $\mathcal{A}$ be an adversary playing the sEUF-1CMA security game for a signature scheme $\Sigma$. We say that it $(t_\mathcal{A}, \varepsilon_\mathcal{A})$-breaks the sEUF-1CMA security of $\Sigma$ if it runs in time $t_\mathcal{A}$, $(M, \sigma) \neq (M^*, \sigma^*)$, and

$$\Pr[\mathbf{Finalize}(\sigma^*, M^*) = 1] \geq \varepsilon_\mathcal{A}.$$

**Identity-Based Encryption (IBE).** The concept of IBE was introduced by Shamir [Sha84]. In IBE schemes the public keys are the identities of users. The corresponding secret keys are issued by a trusted central authority to the users. Due to the fact that public key encryption schemes (PKE) have usually much higher computational costs than symmetric key encryption schemes it is a common approach to use a PKE only to encrypt a short secret key from a symmetric key encryption scheme and then to use the symmetric key encryption scheme together with this key to encrypt the message, which is usually much larger than the symmetric key. This concept is called key encapsulation. The notion of Identity-Based Key Encapsulation (IB-KEM) and the corresponding security notion were introduced by Bentahar et al. [BFMLS05, BFMS08].

**Definition 2.6.** An identity-based key encapsulation mechanism(IB-KEM) $\Pi$ consists of the following four PPT algorithms:

- $\mathbf{Setup}(1^\lambda) \to (\mathsf{mpk}, \mathsf{msk})$. On input the security parameter, it outputs the public parameters $\mathsf{mpk}$ and the master secret key $\mathsf{msk}$.
- $\mathbf{KeyGen}(\mathsf{msk}, id) \to USK_{id}$. On input the master secret key $\mathsf{msk}$ and an identity $id$ it returns the user secret key $USK_{id}$.
- $\mathbf{Encap}(\mathsf{mpk}, id) \to (C, K)$. On input the public parameters $\mathsf{mpk}$ and an identity $id$ it returns a tuple $(C, K)$, where $C$ is ciphertext encapsulating $K$ with respect to identity $id$.
- $\mathbf{Decap}(USK_{id}, C, id) = K$. On input the user secret key $USK_{id}$ together with identity $id$ and a ciphertext $C$ it returns the decapsulated key $K$ or an error symbol $\perp$.

For correctness we require that for all pairs $(\mathsf{mpk}, \mathsf{msk})$ generated by $\mathsf{Setup}(1^\lambda)$, all identities $id \in \mathcal{I}$, all $(K, C)$ output by $\mathsf{Encap}(\mathsf{mpk}, id)$ and all $USK_{id}$ generated by $\mathsf{KeyGen}(\mathsf{msk}, id)$ the following equation is satisfied:

$$\Pr[\mathsf{Decap}(USK_{id}, C, id) = K] = 1.$$

**IND-ID-CPA security.** IND-ID-CPA seurity for an IB-KEM $\Pi$ is defined via the following game between a challenger and an adversary $\mathcal{A}$. The challenger computes $\mathbf{Setup}(1^\lambda) \to (\mathsf{mpk}, \mathsf{msk})$ and sends $\mathsf{mpk}$ to $\mathcal{A}$. The adversary has access to the following oracles.

- $\underline{\mathbf{KeyQuery}(id)}$. On input an identity $id$, the challenger computes and outputs the secret key $\mathbf{KeyGen}(\mathsf{msk}, id) \to USK_{id}$.
- $\underline{\mathbf{Challenge}(id^*)}$. The adversary submits a challenge identity $id^*$. The challenger picks a bit $b$ and a key $K_0$ from the key space uniformly at random. Then, it computes $\mathbf{Encap}(\mathsf{mpk}, id) \to (C, K_1)$. It returns $(C, K_b)$ to $\mathcal{A}$.
- $\underline{\mathbf{Guess}(b')}$. The adversary outputs its guess $b' \in \{0, 1\}$. The challenger outputs 1 if $b = b'$, else 0. The game stops.

The adversary is allowed to make multiple queries to $\mathbf{KeyQuery}(id)$ and one query to $\mathbf{Challenge}(id^*)$ in any order, but with the restriction that it does not query a key for $id^*$. $\mathbf{Guess}(b')$ can only be queried after $\mathbf{Challenge}(id, M_0, M_1)$.

**Definition 2.7.** Let $\mathcal{A}$ be an adversary playing the IND-ID-CPA security game for an IB-KEM $\Pi$. We say that it $(t_\mathcal{A}, \varepsilon_\mathcal{A})$-breaks the IND-ID-CPA security of $\Pi$, if it runs in time $t_\mathcal{A}$ and

$$|\Pr[\mathbf{Guess}(b') = 1] - 1/2| \geq \varepsilon_\mathcal{A}.$$

We include the running time of the security experiment into the running time $t_\mathcal{A}$ of $\mathcal{A}$. This will later allow us to simplify our security analysis.

## 2.3 Bilinear Pairings

Bilinear pairings describe a bilinear mapping that is non-degenerate and efficiently computable. They are a widely used and established tool in modern cryptography. Moreover, they enable an abstract and simple view on elliptic curves.

**Definition 2.8.** Let $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ be cyclic groups of prime order $q$ with generators $g_1, g_2, g_T$. We call $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ a bilinear pairing if:

1. $e(g_1^a, g_2^b) = e(g_1, g_2)^{ab}$ for all $a, b \in \mathbb{Z}_q$,
2. $e(g_1, g_2) \neq 1_T$,
3. $e$ can be efficiently computed.

If $\mathbb{G}_1 = \mathbb{G}_2$ we call it a Type-1 pairing. If $\mathbb{G}_1 \neq \mathbb{G}_2$ and there is an efficiently computable isomorphism from $\mathbb{G}_2$ to $\mathbb{G}_1$ we call it a Type-2 pairing and if there is no efficiently computable isomorphism from $\mathbb{G}_2$ to $\mathbb{G}_1$ we call it a Type-3 pairing. For more information we refer to [BLS01, CHM10].

CHAPTER 3

# Simple and Efficient PRFs with Tight Security via All-Prefix Universal Hash Functions

*We construct efficient and tightly secure pseudorandom functions (PRFs). These PRFs are secure against adaptive adversaries and have only a logarithmic security loss and short secret keys. Moreover, these PRFs are very simple and efficient variants of well-known constructions, including those of Naor-Reingold [NR97] and Lewko-Waters [LW09]. Most importantly, in combination with the construction of Banerjee et al. [BPR12] we obtain the currently most efficient LWE-based PRF from a weak LWE-assumption with a much smaller modulus than the original construction. Technically, we introduce all-prefix universal hash functions (APUHFs), which are hash functions that are (almost-)universal, even if any prefix of the output is considered. We give simple and very efficient constructions of APUHFs, and show how they can be combined with the augmented cascade of Boneh et al. [BMR10]. Along the way, we develop a new and more direct way to prove security of PRFs based on the augmented cascade.*
*The results in this chapter are based on collaborations with Tibor Jager and Jiaxin Pan. The notion of all-prefix universality and the augmented cascade with encoded input are mainly due to Tibor Jager and myself. The applications are mainly due to Jiaxin Pan. The results appeared at Asiacrypt 2018 [JKP18].*

## 3.1 Introduction

A pseudorandom function (PRF) is a function that is computationally indistinguishable from a real random function. Due to the fact that a huge amount of applications requires "randomness" PRFs are a very important foundational cryptographic primitive; see [GGM84, BG90, Gol01, Bel06, Kra10] for example. One possibility to construct PRFs is to construct them from one-way functions (via pseudorandom generators) [GGM86]. However, this approach is rather inefficient. We aim at constructing efficient PRFs from as-weak-as-possible assumptions and with tight security proof.

**Tight security.** In a cryptographic security proof, we often consider an adversary $\mathcal{A}$ against a primitive like a PRF, and describe a reduction $\mathcal{B}$ that runs $\mathcal{A}$ as a subroutine to break some computational problem which is assumed to be hard. Let $(t_\mathcal{A}, \varepsilon_\mathcal{A})$ and $(t_\mathcal{B}, \varepsilon_\mathcal{B})$ denote the running time and success probability of $\mathcal{A}$ and $\mathcal{B}$, respectively. Then we say that the reduction $\mathcal{B}$ *loses* a factor $\ell$, if

$$\frac{t_{\mathcal{B}}}{\varepsilon_{\mathcal{B}}} = \ell \cdot \frac{t_{\mathcal{A}}}{\varepsilon_{\mathcal{A}}}.$$

A reduction is usually considered "efficient", if $\ell$ is bounded by a polynomial in the security parameter. We say that a reduction is "tight", if $\ell$ is small. Our goal is to construct reductions $\mathcal{B}$ such that $\ell$ is as small as possible. Ideally we would like to have $\ell = O(1)$ constant. To illustrate this let us define $\mathcal{WF}(\mathcal{B}) := \frac{t_{\mathcal{B}}}{\varepsilon_{\mathcal{B}}}$ and $\mathcal{WF}(\mathcal{A}) := \frac{t_{\mathcal{A}}}{\varepsilon_{\mathcal{A}}}$ as the work factors of $\mathcal{B}$ and $\mathcal{A}$, respectively. We say that a computational hard problem provides $\kappa$ bits of security against $\mathcal{B}$ if $2^{\kappa} > \mathcal{WF}(\mathcal{B})$ and analogously, that a cryptographic scheme provides $\kappa'$ bits of security against $\mathcal{A}$ if $2^{\kappa'} > \mathcal{WF}(\mathcal{A})$. Let us assume for the security loss that $\ell = 2^{30}$, which is a reasonable and generally considered loss. Now if we want that the cryptographic scheme provides 80 bits of security we need that the computational hard problem provides 110 bits of security, because

$$\mathcal{WF}(\mathcal{B}) = \ell \cdot \mathcal{WF}(\mathcal{A}) < 2^{30} \cdot 2^{80} = 2^{110}.$$

Usually a higher work factor of a computational hard problem can be achieved by larger group sizes, for instance for the discrete logarithm problem or factorization problem. However, larger groups increase computational costs. Hence, if we had $\ell = 1$ we would have $\mathcal{WF}(\mathcal{B}) = 2^{80}$ and thus we could choose smaller algebraic groups to instantiate the cryptographic scheme. It is worth to mention that for many cryptographic constructions and primitives a constant loss of $O(1)$ is impossible to achieve [Cor02, KK12, HJK12, LW14, BJLS16]. However, even in such cases it is worth to seek for the optimal security loss to achieve efficiency improvements.

**State of the art.** The *augmented cascade* framework of Boneh *et al.* [BMR10] covers many constructions of efficient number-theoretic PRFs like the general Matrix-DDH-based construction of [EHK$^+$13a] (with the well-known algebraic constructions of Naor-Reingold [NR97] and Lewko-Waters [LW09] as special cases) and also the LWE-based PRF of Banerjee, Peikert, and Rosen [BPR12].

For these constructions, the size of the secret key and the loss in the security proof grow linearly[1] with the length $n$ of the function input. This means that efficiency and security both depend on $n$ and by this also on the size of the input space. The general approach to extend the input space to $\{0,1\}^*$ is to apply a collision-resistant hash function $H : \{0,1\}^* \to \{0,1\}^n$, where $n = 2\lambda$ and $\lambda$ denotes the security parameter, to the input before processing it in the PRF. This approach results in secret keys consisting of $n = O(\lambda)$ elements (where the concrete type of elements depends on the particular instantiation of the augmented cascade) and a security loss of $\ell = n = O(\lambda)$.

**Contributions.** We introduce *all-prefix universal hash functions* (APUHFs) as a special type of hash functions that are universal, even if the output of the hash function is truncated. We also describe a very simple and efficient construction, which is based on the hash function of Dietzfelbinger *et al.* [DHKP97], as well as a generic construction from pairwise independent hash functions with range $\{0,1\}^n$ for some $n \in \mathbb{N}$.

[1]Here, we count the number of elements, not their bit size that increases with the security parameter. This a common approach in literature.

Then we show that by combining the augmented cascade with an APUHF, we are able to significantly improve both the asymptotic size of secret keys and the security loss of these constructions. Specifically, we achieve keys consisting of only a slightly super-logarithmic number of elements $m = \omega(\log \lambda)$ and an only logarithmic security loss $O(\log \lambda)$. Both the number of elements in the secret key and tightness are *independent* of the input size $n$, except for the key of the APUHF, which consists of $n$ bits when instantiated with the APUHF of Dietzfelbinger *et al.* [DHKP97]. Based on this generic result, we then obtain simple variants of algebraic PRFs based on a large class of Matrix-DDH assumptions [EHK$^+$13a], which include the PRFs of Naor and Reingold [NR97] and its generalization by Lewko and Waters [LW09] as special cases.

Furthermore, we obtain a simple variant of the PRF of Banerjee, Peikert and Rosen [BPR12] (BPR). This PRF is based on the learning-with-errors (LWE) assumption [Reg05], and has the property that the required size of the LWE modulus depends on the length of the PRF input. More precisely, the lower bound on the LWE modulus $p$ is exponential in the input length $n = \Theta(\lambda)$. We observe this in almost all the well-known LWE-based PRFs such as [BLMR13, BP14]. In order to improve efficiency and to base security on a weaker LWE assumption, it is thus desirable to make $p$ as small as possible. We show that simply encoding the PRF input with an APUHF before processing it in the original BPR construction makes it possible to reduce the lower bound on the LWE modulus $p$ from exponential to only slightly super-polynomial in the security parameter, which yields a weaker assumption and a significant efficiency improvement (see Section 3.5.2 for details). Furthermore, even for an arbitrary polynomially-bounded input size $n$, our construction requires to store only $m = \omega(\log \lambda)$ matrices, independent of the size $n$ of the input space $\{0,1\}^n$, plus a single bitstring of length $n$ when instantiated with the APUHF of Dietzfelbinger *et al.* [DHKP97]. In contrast, the original construction from [BPR12] requires $\Theta(n)$ matrices.

A similar improvement of the LWE modulus $p$ was achieved by a different BPR variant due to Döttling and Schröder in [DS15], via a technique called *on-the-fly adaptation.* However, their construction requires to run $\lambda \cdot \omega(\log \lambda)$ copies of the BPR PRF in parallel, while ours requires only a single copy plus an APUHF. Thus, our approach is significantly more efficient, and also more direct, as it essentially corresponds to the original BPR function, except that an APUHF is applied to the input. This simplicity gives not only a useful conceptual perspective on the construction of tightly secure PRFs, but it also makes schemes easier to implement securely.

Another advantage of our approach is that the resulting PRF construction is extremely simple. It is essentially identical to the augmented cascade from [BMR10], except that an APUHF $h$ is applied to the input before it is processed by the PRF. More precisely, let $\hat{F}^m$ be a PRF that is constructed from an $m$-fold application of an underlying function $F$ via the augmented cascade construction from [BMR10]. Then, our construction $\hat{F}(K, x)$ has the form

$$\hat{F}(K, x) := \hat{F}^m(s, h(x)),$$

where the key of our new function is a tuple $K = (s, h)$ consisting of a random key $s$ for the augmented cascade construction and a random function $h \leftarrow \mathcal{H}$ from a family $\mathcal{H} = \{h : \{0,1\}^n \rightarrow \{0,1\}^m\}$ of APUHFs.

We remark that we require an additional property called *perfect one-time security*("1-uniformity") of the underlying function $F$ of the augmented cascade, and thus technically our variant of [BMR10] is slightly less general. However, this is a minor restriction, as we show that this property is satisfied by all known instantiations of the augmented cascade. Furthermore, our security proof assumes that the reduction "knows" sufficiently close approximations of the number of queries $Q$ and the advantage $\varepsilon_{\mathcal{A}}$ of the adversary. Thus, the proof shows how such non-black-box knowledge can be used to achieve more efficient PRFs with short keys and very tight security from weaker assumptions.

**Technical approach.** An augmented cascade PRF with $m$-bit input is a function $\hat{F}^m : S^m \times K \times \{0,1\}^m \to K$ with key space $S^m \times K$. In the sequel, let $(s_1, \ldots, s_m, k) \in S^m \times K$ be a key for $\hat{F}^m$ and $h : \{0,1\}^n \to \{0,1\}^m$. For a string $a \in \{0,1\}^m$ we write $a_{v:w}$ to denote the substring $(a_v, \ldots, a_w) \in \{0,1\}^{w-v+1}$ of $a$. Let $j$ be an integer with $j \leq m$ (we will explain later how to choose $j$ in a suitable way). An augmented cascade PRF $\hat{F}^m$ has the property, that for each $j \in \{1, \ldots, m\}$, it can be implemented equivalently as a two-step algorithm as follows.

1. In the first step, the function $\hat{F}^m$ processes only the first $j$ bits $h(x)_{1:j} \in \{0,1\}^j$ of $h(x)$, to compute an intermediate value $k_x$ that depends only on the first $j$ bits of $h(x)$:
$$k_x = \hat{F}^j((s_1, ..., s_j), k, h(x)_{1:j})$$
2. Then the remaining $m - j$ bits are processed, starting from $k_x$, by computing
$$y = \hat{F}^{m-j}((s_{j+1}, ..., s_m), k_x, h(x)_{j+1:m})$$

The resulting function is identical to the function $\hat{F}^m$. This specific way to implement $\hat{F}^m$ will be useful to describe our approach.

In the security proof, let $x^{(1)}, \ldots, x^{(Q)}$ denote the sequence of pairwise distinct oracle queries issued by the adversary in the PRF security experiment. Further let us assume that $h(x^{(u)})_{1:j} \neq h(x^{(v)})_{1:j}$ for $u \neq v$. We show that the security of $\hat{F}^m$ is implied by the security of $\hat{F}^j$, which is a PRF with shorter input. The intuition behind the security argument can be described in two steps.

1. We replace $\hat{F}^j$ with a random function $R$, which is computationally indistinguishable due to the security of $\hat{F}^j$. The intermediate value $k_x = R(h(x)_{1:j})$ is an independent random value for each oracle query made by the adversary, because we assume $h(x^{(u)})_{1:j} \neq h(x^{(v)})_{1:j}$ for $u \neq v$.
2. Next we argue that $\hat{F}^m$ is distributed like a random function as well. We achieve this by identifying an additional property required from $\hat{F}^{m-j}$ that we call *perfect one-time security.* This property guarantees that
$$\Pr_{k_x \leftarrow K}\left[\hat{F}^{m-j}((s_{j+1}, ..., s_m), k_x, h(x)_{j+1:m}) = y\right] = \frac{1}{|K|}$$

for all $((s_{j+1}, ..., s_m), h(x)_{j+1:m}, y) \in S^{m-j} \times \{0,1\}^{m-j} \times K$. This is sufficient to show that indeed now the function

$$\hat{F}^{m-j}((s_{j+1}, ..., s_m), R(h(x)_{1:j}), h(x)_{j+1:m})$$

is a random function, because we supposed $h(x^{(u)})_{1:j} \neq h(x^{(v)})_{1:j}$ for $u \neq v$.

It remains to ensure the assumption that $h(x^{(u)})_{1:j} \neq h(x^{(v)})_{1:j}$ for all $u \neq v$ is true with "sufficiently large" probability and for some "sufficiently small" value of $j$. In order to do so we use the all-prefix universal hash function, in combination with an argument which on a high level follows similar proofs from [BH12] and [DS15]. The main difference is that we use the all-prefix universality to argue that setting $j := \lceil \log(2Q^2/\varepsilon_{\mathcal{A}}) \rceil = O(\log \lambda)$ is sufficient to guarantee that, $h(x^{(u)})_{1:j} \neq h(x^{(v)})_{1:j}$ with sufficiently large probability for all $u \neq v$. Here, $Q$ is the number of oracle queries made by the adversary in the PRF security experiment and $\varepsilon_{\mathcal{A}}$ is its advantage.

Due to the fact that $j = O(\log \lambda)$ we only have to require security of a "short-input" augmented cascade $\hat{F}^j$ with $j = O(\log \lambda)$. For our algebraic instantiations based on Matrix-DDH problems, this results in a tightness loss of only $O(\log \lambda)$. For our application to the LWE-based PRF of Banerjee, Peikert and Rosen [BPR12], this gives us that we require only a weaker LWE assumption. Furthermore, we need only that $m \geq j$ for all possible values of $j$ and $j = \lceil \log(2Q^2/\varepsilon_{\mathcal{A}}) \rceil = O(\log \lambda)$. Hence, it is sufficient to set $m = \omega(\log \lambda)$ slightly super-logarithmic, which results in short secret keys and efficient evaluation for all instantiations.

**Why all-prefix universal hash functions?** We want to emphasize that we indeed require an *all-prefix* universal hash function that works for any possible prefix length $j$. We need this to enable that the construction and the security proof become independent of particular values $Q$ and $\varepsilon_{\mathcal{A}}$ of a particular adversary, because $j$ depends on these values via the definition $j = \lceil \log(2Q^2/\varepsilon_{\mathcal{A}}) \rceil$. *All-prefix* universality guarantees basically that a suitable value of $j$ exists for any efficient adversary. This is also required to achieve tightness. See Section 3.6 for further discussion.

**More related work.** There were several other works about the domain extension of PRFs. The first one is due to Levin [Lev87]. It shows that larger inputs can be hashed with a universal hash function if the underlying PRF has a sufficiently large domain. Otherwise it is vulnerable to the so called "birthday attack". The framework of Jain, Pietrzak, and Tentes [JPT12] works for small domains, but has a rather lossy security proof and is not very efficient, as it needs $\mathcal{O}(\log q)$ invocations of the underlying pseudo-random generator (PRG), where $q$ is the upper bound of queries to the PRF. Additionally, as the authors already mention, it seems not to work for number-theoretic PRFs like the Naor-Reingold PRF. It was revisited by Chandran and Garg [CG14]. Bernam *et al.* show how to circumvent the "birthday attack" using Cuckoo Hashing [BHKN13] via two invocations of the original PRF.

## 3.2 All-Prefix Universal Hash Functions

In this section, we define all-prefix almost universal hash functions and describe two constructions. The first one is based on the almost-universal hash function of Dietzfelbinger *et al.* [DHKP97], and yields an all-prefix almost-universal hash function. The second one is based on pairwise independent hash functions with suitable range, and yields an all-prefix universal hash function. We start by recalling the standard definition of universal hash functions.

**Definition 3.1[CW79].** A family $\mathcal{H}$ of hash functions mapping finite set $\{0,1\}^n$ to finite set $\{0,1\}^m$ is *universal*, if for all $x, x' \in \{0,1\}^n$ with $x \neq x'$ holds that

$$\Pr_{h \leftarrow \mathcal{H}}[h(x) = h(x')] \leq 2^{-m}.$$

We also consider *almost-universal* hash functions, as defined below.

**Definition 3.2.** A family $\mathcal{H}$ of hash functions mapping finite set $\{0,1\}^n$ to finite set $\{0,1\}^m$ is *almost-universal*, if for all $x, x' \in \{0,1\}^n$ with $x \neq x'$ holds that

$$\Pr_{h \leftarrow \mathcal{H}}[h(x) = h(x')] \leq 2^{-m+1}.$$

Universal and almost-universal hash functions can be constructed efficiently and without additional complexity assumptions, see e.g. [CW79, DHKP97, IKOS08].

We introduce the following definition.

**Definition 3.3.** Let $\mathcal{H}$ be a family of hash functions mapping $\{0,1\}^n$ to $\{0,1\}^m$. We say that $\mathcal{H}$ is a family of *all-prefix universal hash functions*, if for all $x, x' \in \{0,1\}^n$ with $x \neq x'$ and all $w = 1, \ldots, m$ holds that

$$\Pr_{h \leftarrow \mathcal{H}}[h(x)_{1:w} = h(x')_{1:w}] \leq 2^{-w}.$$

Note that all-prefix universality essentially means that for all prefixes of length $w$ the truncation of $h$ to its first $w$ bits $h(x)_{1:w}$ is a universal hash function. We also introduce the slightly weaker notion of all-prefix *almost*-universality.

**Definition 3.4.** Let $\mathcal{H}$ be a family of hash functions mapping $\{0,1\}^n$ to $\{0,1\}^m$. We say that $\mathcal{H}$ is a family of *all-prefix almost-universal hash functions* (APUHFs), if for all $x, x' \in \{0,1\}^n$ with $x \neq x'$ and all $w \in [m]$ holds that

$$\Pr_{h \leftarrow \mathcal{H}}[h(x)_{1:w} = h(x')_{1:w}] \leq 2^{-w+1}.$$

### 3.2.1 First Construction (Almost-Universal)

We construct a simple and efficient APUHF family based on the almost-universal hash function of Dietzfelbinger *et al.* [DHKP97], which is defined as follows. Let $m, n \in \mathbb{N}$ with $m \leq n$. Let

$$\mathcal{H}_{n,m} := \{h^a : a \in [2^n - 1] \text{ and } a \text{ is odd}\} \tag{3.1}$$

be the family of hash functions, which for $x \in \mathbb{Z}_{2^n}$ is defined as

$$h^a(x) := (ax \bmod 2^n) \text{ div } 2^{n-m}, \tag{3.2}$$

where div $2^{n-m}$ keeps the $m$ most significant bits.
Before we prove that this function is all-prefix almost-universal, we first state the following lemma of Dietzfelbinger *et al.* [DHKP97].

**Lemma 3.1** ([DHKP97])**.** *Let $n$ and $m$ be positive integers with $m = 1, \dots, n$. If $x, y \in \mathbb{Z}_{2^n}$ are distinct and $h^a \in \mathcal{H}_{n,m}$ is chosen at random, then*

$$\Pr[h^a(x) = h^a(y)] \leq 2^{-m+1}$$

*Thus, $\mathcal{H}_{n,m}$ is a family of almost-universal hash functions in the sense of Definition 3.2.*

**All-prefix almost-universality of $\mathcal{H}_{n,m}$.** Now we prove that the hash function family $\mathcal{H}_{n,m}$ of Dietzfelbinger *et al.* [DHKP97] is not only almost-universal, but also satisfies the stronger property of *all-prefix* almost-universality.

**Theorem 3.1.** *$\mathcal{H}_{n,m}$ is a family of all-prefix almost-universal hash functions in the sense of Definition 3.4.*

*Proof.* Let $\omega, m, n$ be any positive integers with $\omega \leq m \leq n$. Note that if $h^a(\cdot)$ is a function in $\mathcal{H}_{n,m}$ then $ha(\cdot)_{1:\omega}$ is a function in $\mathcal{H}_{n,\omega}$. Further note that Lemma 3.1 holds for all $\omega = 1, \dots, n$, which proves the claim. $\square$

In the sequel, we will sometimes write $h$ instead of $h^a$, when it is clear from the context that $h$ is be chosen uniformly random from $\mathcal{H}_{n,m}$.

### 3.2.2 Second Construction (Universal)

While the almost-universal construction from Section 3.2.1 is already sufficient for all our applications, it is natural to ask whether also all-prefix *universal* hash functions (not almost-universal) can be constructed. We will show that each pairwise-independent family of hash functions with range $\{0,1\}^n$ is also a family of all-prefix universal hash functions. To this end, let us first recall the notion of pairwise independent hash functions.

**Definition 3.5.** Let $\mathcal{H}$ be a family of hash functions with domain $\{0,1\}^n$ and range $\{0,1\}^m$. We say that $\mathcal{H}$ is *pairwise independent*, if for all $x, x' \in \{0,1\}^n$ with $x \neq x'$ and all $y, z \in \{0,1\}^m$ holds that

$$\Pr_{h \leftarrow \mathcal{H}}[h(x) = y \wedge h(x') = z] = 2^{-2m}.$$

We first show that pairwise independence implies all-prefix pairwise independence, which is defined below. Then we show that this implies all-prefix universality. Let us write $x_i$ to denote the $i$-th bit of the bit string $x$.

**Definition 3.6.** Let $\mathcal{H}$ be a family of hash functions mapping $\{0,1\}^n$ to $\{0,1\}^m$. We say that $\mathcal{H}$ is *all-prefix pairwise independent*, if for all $x, x' \in \{0,1\}^n$ with $x \neq x'$ and all $y, z' \in \{0,1\}^m$ holds that

$$\Pr_{h \leftarrow \mathcal{H}}[h(x)_{1:w} = y_{1:w} \wedge h(x')_{1:w} = z_{1:w}] = 2^{-2w}$$

for all $w = 1, \dots, m$.

**Lemma 3.2.** *If $\mathcal{H}$ is pairwise independent, then it is also all-prefix pairwise independent.*

*Proof.* We have

$$\begin{aligned}
&\Pr_{h \leftarrow \mathcal{H}}[h(x)_{1:j} = y_{1:j} \wedge h(x')_{1:j} = z_{1:j}] \\
&= \Pr_{h \leftarrow \mathcal{H}}\left[\left(\bigcup_{y' \in \{0,1\}^{m-j}} h(x) = (y_{1:j} \parallel y')\right) \wedge \left(\bigcup_{z' \in \{0,1\}^{m-j}} h(x') = (z_{1:j} \parallel z')\right)\right] \\
&= \sum_{y' \in \{0,1\}^{m-j}} \sum_{z' \in \{0,1\}^{m-j}} \Pr_{h \leftarrow \mathcal{H}}\left[h(x) = (y_{1:j} \parallel y') \wedge h(x') = (z_{1:j} \parallel z')\right] \\
&= \sum_{y' \in \{0,1\}^{m-j}} \sum_{z' \in \{0,1\}^{m-j}} \frac{1}{2^{2m}} = \frac{2^{m-j} \cdot 2^{m-j}}{2^{2m}} = \frac{1}{2^{2j}}.
\end{aligned}$$

□

Now, it remains to show that all-prefix pairwise independence implies all-prefix universality.

**Lemma 3.3.** *If $\mathcal{H}$ is all-prefix pairwise independent, then it is also all-prefix universal.*

*Proof.* It holds that

$$\begin{aligned}
\Pr_{h \leftarrow \mathcal{H}}[h(x)_{1:j} = h(x')_{1:j}] &= \sum_{y_{1:j} \in \{0,1\}^j} \Pr_{h \leftarrow \mathcal{H}}[h(x)_{1:j} = y_{1:j} \wedge h(x')_{1:j} = y_{1:j}] \qquad (3.3) \\
&= \sum_{y_{1:j} \in \{0,1\}^j} \frac{1}{2^{2j}} = \frac{1}{2^j},
\end{aligned}$$

where (3.3) holds because of Lemma 3.2. □

**Example instantiation.** Let $n \in \mathbb{N}$ and let

$$\mathcal{H}_n := \{h^{a,b} : a, b \in \{0,1\}^n\}$$

be the family of hash functions

$$h^{a,b} : GF(2^n) \to GF(2^n); x \mapsto ax + b,$$

where the arithmetic operations are in $GF(2^n)$.Since it is well-known that $\mathcal{H}_n$ is pairwise independent we leave the following theorem without proof.

**Theorem 3.2.** *$\mathcal{H}_n$ is a family of all-prefix universal hash functions.*

Note that in the explicit construction of $GF(2^n)$ the choice of the irreducible polynomial has big impact on the efficiency of the arithmetic operations.

```
Input: Key $(s_1, ..., s_m, k_0) \in S^m \times K$ and $(x_1, ..., x_m) \in X^m$
For $i = 1, ..., m$ :
    $k_i \leftarrow F((s_i, k_{i-1}), x_i)$
Return $k_m$.
```

Figure 3.1: Definition of function $\hat{F}^m$ of Boneh *et al.* [BMR10].

## 3.3 Augmented Cascade PRFs

Boneh *et al.* s[BMR10] showed how to construct a PRF

$$\hat{F}^m : (S^m \times K) \times X^m \to K$$

with key space $(S^m \times K)$ and input space $X$ from an augmented cascade of functions

$$F : (S \times K) \times X \to K$$

The augmented cascade construction is described in Figure 3.1. Boneh *et al.* [BMR10] prove that $\hat{F}^m$ is a secure PRF, if $F$ is *parallel secure* in the following sense.

**Definition 3.7[BMR10].** For a function $F : (S \times K) \times X \to K$ define $F^{(Q)}$ as the function

$$F^{(Q)} : (S \times K^Q) \times (X \times [Q]) \to K; \qquad ((s, k_1, ..., k_q), (x, i)) \mapsto F((s, k_i), x) .$$

We say that $\mathcal{A}$ $(t_\mathcal{A}, \varepsilon_\mathcal{A}, Q)$-breaks the *$Q$-parallel security* of $F : (S \times K) \times X \to K$, if it $(t_\mathcal{A}, \varepsilon_\mathcal{A}, Q)$-breaks the pseudorandomness of $F^{(Q)}$ in the sense of Definition 2.2.

**Theorem 3.3** ([BMR10]). *From each adversary $\mathcal{A}$that $(t_\mathcal{A}, \varepsilon_\mathcal{A}, Q)$-breaks the pseudorandomness of $\hat{F}^m$, one can construct an adversary $\mathcal{B}$that $(t_\mathcal{B}, \varepsilon_\mathcal{B}, Q)$-breaks the $Q$-parallel security of $F^{(Q)}$ with*

$$t_\mathcal{B} = \Theta(t_\mathcal{A}) \qquad \text{and} \qquad \varepsilon_\mathcal{B} \geq \frac{\varepsilon_\mathcal{A}}{m}$$

Note that the security loss of this construction is linear in the length $m$ of the input of function $\hat{F}^m$.

## 3.4 The Augmented Cascade with Encoded Input

In this section, we show that APUHFs enable the instantiation of augmented cascade PRFs [BMR10] with shorter keys of sligtly super-logarithmic size $\omega(\log \lambda)$. The security proof loses only a factor $O(\log \lambda)$, independent of the input size of the PRF, assuming that (reasonably close bounds) on the number of queries $Q$ and the success probability $1/2+\varepsilon_\mathcal{A}$ of the PRF adversary $\mathcal{A}$ are known *a priori*. In contrast, the loss of the previous security proof of [BMR10] is linear in the input size of the PRF (which is usually linear in $\lambda$), but does not assume any *a priori* knowledge about $\mathcal{A}$. We consider the special case with input space $X = \{0, 1\}$, which encompasses the MDDH-based construction of

Escala *et al.* [EHK+13a] and thus includes in particular both the instantiations of Naor-Reingold [NR97] and Lewko-Waters [LW09]. Let $\mathcal{H}_{n,m}$ be a family of all-prefix almost-universal hash functions according to Definition 3.4, and let $F : (S \times K) \times \{0,1\} \to K$ be a function. We define the corresponding augmented cascade PRF with $\mathcal{H}_{n,m}$-encoded input as the function

$$\begin{aligned} \hat{F}^{\mathcal{H}_{n,m}} : S^m \times K \times \mathcal{H}_{n,m} \times \{0,1\}^n &\to K \\ ((s_1, ..., s_m), k, h, x) &\mapsto \hat{F}^m((s_1, ..., s_m), k, h(x)) \end{aligned} \tag{3.4}$$

where $\hat{F}^m$ is the augmented cascade construction of Boneh *et al.* [BMR10], applied to $F$ as described in Figure 3.1.

**Remark 3.1.** *Note that evaluating the PRF requires only $m$ recursions in the augmented cascade, and that, accordingly, the secret key consists of only $m$ elements and the description of $h$, while the input size can be any polynomial number of $n$ bits, with possibly $n \gg m$. We will later show that it suffices to set $m = \omega(\log \lambda)$ slightly super-logarithmic, thanks to the input encoding with an all-prefix almost-universal hash function. Also the security loss of this construction is only $O(\log \lambda)$ and independent of the size of the input $n$.*

## 3.4.1 Preparation for the Security Proof

In this section we describe a few technical observations which will simplify the security proof. Furthermore, we define *perfect one-time security* as an additional property of a function $F(s, x, k)$, which will also be required for the proof. We will argue later that the Matrix-DDH-based instantiations of the augmented cascade of [EHK+13a], including the functions of Naor-Reingold [NR97] and Lewko-Waters [LW09], all satisfy this additional notion. Moreover, we will show that the LWE-based PRF of [BPR12] can be viewed as an augmented cascade and it is perfectly one-time secure.

**An observation about the augmented cascade.** The following observation will be useful to follow the security proof more easily. Suppose we want to compute

$$z = \hat{F}^m((s_1, ..., s_m), k, h(x))$$

then, due to the recursive definition of $\hat{F}^m$, we can equivalently proceed in the following two steps.

1. Let $i \in [m]$. We first process the first $i$ bits $h(x)_{1:i}$ of $h(x)$ with $(s_1, \ldots, s_i, k)$, and compute and "intermediate key" $k_x$ as

$$k_x := \hat{F}^i((s_1, \ldots, s_i), k, h(x)_{1:i})$$

2. Then we process the remaining $m - i$ bits $h(x)_{i+1:m}$ of $h(x)$ with the remaining key elements $(s_{i+1}, \ldots, s_m, k_x)$ by computing

$$z = \hat{F}^{m-i}((s_{i+1}, ..., s_m), k_x, h(x)_{i+1:m})$$

We formulate this observation as a lemma.

**Lemma 3.4.** *For all $i = 1, \dots, m$, we have*

$$\hat{F}^m((s_1, ..., s_m), k, h(x)) = \hat{F}^{m-i}((s_{i+1}, ..., s_m), k_x, h(x)_{i+1:m})$$

*where $k_x := \hat{F}^i((s_1, \dots, s_i), k, h(x)_{1:i})$.*

**Perfect One-Time Security.** We will furthermore require an additional security property of $F$, which we call *perfect one-time security*, and show that this property is satisfied by all instantiations of function $F$ considered in this section. We demand that $F(s, x, k)$ is identically distributed to a random function $R(x)$, if it is only evaluated once. This must hold over the uniformly random choice $k \leftarrow K$, and for any $s \in S$ and $x \in \{0, 1\}$.

**Definition 3.8.** We say that a function $F : S \times K \times \{0, 1\}^m \to K$ is *perfectly one-time secure*, if

$$\Pr_{k \leftarrow K} \left[F(s, k, x) = k'\right] = \frac{1}{|K|}$$

for all $(s, x, k') \in S \times \{0, 1\}^m \times K$.

Perfect one-time security basically guarantees uniformity of the hash function, if it is evaluated only once ("1-uniformity").

The following lemma follows directly from Definition 3.8. It will be useful to prove security of our variant of the augmented cascade.

**Lemma 3.5.** *Let $m \in \mathbb{N}$ and $F : S \times K \times \{0, 1\} \to K$ be perfectly one-time secure. Then the augmented cascade $\hat{F}^m$ constructed from $F$ is also perfectly one-time secure. That is*

$$\Pr_{k \leftarrow K} \left[\hat{F}^m((s_1, ..., s_m), k, x) = k'\right] = \frac{1}{|K|}$$

*for all $((s_1, ..., s_m), k', x) \in S^m \times K \times \{0, 1\}^m$.*

*Proof.* For a uniformly random chosen $k$ it holds that $\Pr\left[F(s_1, k, x_1) = k_1\right] = \frac{1}{|K|}$ for all $(s_1, k, x_1) \in S \times K \times \{0, 1\}$ because of the perfect one-time security of $F$. Thus the input for the second iteration stays uniformly random. Due to the recursive construction executing all the following iterations will keep this distribution, which gives us the perfect one-time security of $\hat{F}^m$. $\square$

### 3.4.2 Security Proof

Now we are ready to prove the following theorem.

**Theorem 3.4.** *Let $m = \omega(\log \lambda)$ be (slightly) super-logarithmic, $\mathcal{H}_{n,m}$ be a family of all-prefix almost universal hash functions and $F$ be perfectly one-time secure. From each adversary $\mathcal{A}$ that $(t_\mathcal{A}, \varepsilon_\mathcal{A}, Q)$-breaks the pseudorandomness of $\hat{F}^{\mathcal{H}_{n,m}}$ with $Q/\varepsilon_\mathcal{A} = \mathsf{poly}(\lambda)$ for some polynomial $\mathsf{poly}$, we can construct an adversary $\mathcal{B}$ that $(t_\mathcal{B}, \varepsilon_\mathcal{B}, Q)$-breaks the pseudorandomness of $\hat{F}^j$, where*

$$j = O(\log \lambda) \qquad \text{and} \qquad t_\mathcal{B} = \Theta(t_\mathcal{A}) \qquad \text{and} \qquad \varepsilon_\mathcal{B} \geq \varepsilon_\mathcal{A}/2$$

*Proof.* In the sequel let $j = j(\lambda)$ be defined such that

$$j := \lceil \log(2Q^2/\varepsilon_{\mathcal{A}}) \rceil \tag{3.5}$$

Observe that we have $j(\lambda) \leq m(\lambda)$ for sufficiently large $\lambda$, because the fact that we have $Q/\varepsilon_{\mathcal{A}} = \mathsf{poly}(\lambda)$ for some polynomial $\mathsf{poly}$ and $j < \log(2Q^2/\varepsilon_{\mathcal{A}}) + 1$ together yield that $j = O(\log \lambda)$, while we have $m = \omega(\log \lambda)$.

**Remark 3.2.** *Note that although we have $j = O(\log(2Q^2/\varepsilon_{\mathcal{A}})) = O(\log \lambda)$, the constant hidden in the big-O notation depends on the adversary.*

We describe a sequence of games, where Game 0 is the original PRF security experiment, and in the last game the probability that the experiment outputs 1 is 1/2, such that no adversary can have any advantage. Let $X_i$ denote the event that the experiment outputs 1 in Game $i$, and let $\mathcal{O}_i$ denote the oracle provided by the experiment in Game $i$.

**Game 0.** This is the original security experiment. In particular, we have

$$\mathcal{O}_0(x) = \begin{cases} \hat{F}^{\mathcal{H}_{n,m}}((s_1, ..., s_m), k, h, x) & \text{if } b = 1 \\ R(x) & \text{if } b = 0 \end{cases}$$

where $R$ is a random function. Therefore, by definition, it holds that

$$\Pr[X_0] = 1/2 + \varepsilon_{\mathcal{A}}$$

**Game 1.** We change the way how the oracle implements function $\hat{F}^{\mathcal{H}_{n,m}}$. That is, we modify the behaviour of $\mathcal{O}_1$ in case $b = 1$, while in case $b = 0$ oracle $\mathcal{O}_1$ proceeds identical to $\mathcal{O}_0$. Recall that

$$\hat{F}^{\mathcal{H}_{n,m}}((s_1, ..., s_m), k, h, x) = \hat{F}^m((s_1, ..., s_m), k, h(x))$$

$\mathcal{O}_1$ implements this function in a specific way. Using the observation from Lemma 3.4, it computes $\hat{F}^m((s_1, ..., s_m), k, h(x))$ in two steps:

1. $k_x := \hat{F}^j((s_1, \ldots, s_j), k, h(x)_{1:j})$,
2. $z := \hat{F}^{m-j}((s_{j+1}, ..., s_m, k_x, h(x)_{j+1:m})$,

where $j$ is as defined above, and we use that $j \leq m$. By Lemma 3.4, this is just a specific way to implement function $\hat{F}^m$, so the change is purely conceptual and we have

$$\Pr[X_1] = \Pr[X_0]$$

**Game 2.** This game is identical to Game 1, except that we replace the function $\hat{F}^m$ implemented by oracle $\mathcal{O}_1$ *partially* with a random function. More precisely, oracle $\mathcal{O}_2$ chooses a second random function $R_j : \{0,1\}^j \to K$. If $b = 1$, then it computes $z = \mathcal{O}_2(x)$ as

1. $k_x := R_j(h(x)_{1:j})$
2. $z := \hat{F}^{m-j}((s_{j+1}, ..., s_m), k_x, h(x)_{j+1:m})$

If $b = 0$, then it proceeds exactly like $\mathcal{O}_1$. The proof of the following lemma is postponed to Section 3.4.3.

**Lemma 3.6.** *From each $\mathcal{A}$that runs in time $t_\mathcal{A}$ and issues $Q$ oracle queries one can construct an adversary $\mathcal{B}$ that $(t_\mathcal{B}, \varepsilon_\mathcal{B}, Q)$-breaks the pseudorandomness of $\hat{F}^j$ where*

$$t_\mathcal{B} = \Theta(t_\mathcal{A}) \quad \text{and} \quad \varepsilon_\mathcal{B} = |\Pr[X_1] - \Pr[X_2]| \tag{3.6}$$

**Game 3.** This game is identical to Game 2, but $\mathcal{O}_3$ performs an additional check. Whenever $\mathcal{A}$makes an oracle query $x$, $\mathcal{O}_3$ checks whether there has been a previous oracle query $x'$ such that

$$h(x)_{1:j} = h(x')_{1:j}$$

If this holds, then $\mathcal{O}_3$ raises event $\mathsf{coll}$, and the experiment outputs a random bit and terminates. Note that the check is always performed, for both values $b \in \{0, 1\}$. Since both games are identical until $\mathsf{coll}$, we have

$$|\Pr[X_2] - \Pr[X_3]| \leq \Pr[\mathsf{coll}]$$

Again, the proof of the following lemma is postponed, to Section 3.4.4.

**Lemma 3.7.** *If $F$ is perfectly one-time secure, then* $\Pr[\mathsf{coll}] \leq \varepsilon_\mathcal{A}/2$ *and* $\Pr\left[X_3 \mid \overline{\mathsf{coll}}\right] = 1/2$.

We finish the proof of Theorem 3.4 before we prove Lemmas 3.6 and 3.7. We have

$$\Pr[X_3] = \Pr[X_3 \mid \mathsf{coll}] \cdot \Pr[\mathsf{coll}] + \Pr\left[X_3 \mid \overline{\mathsf{coll}}\right] \cdot (1 - \Pr[\mathsf{coll}]) \tag{3.7}$$

Recall that $X_3$ denotes the probability that the experiment outputs 1, which happens if and only if $\mathcal{A}$ outputs $b'$ with $b = b'$. By construction of the experiment, we abort and output a random bit in Game 3, if $\mathsf{coll}$ occurs. In combination with Lemma 3.7 we thus get

$$\Pr[X_3 \mid \mathsf{coll}] = \Pr\left[X_3 \mid \overline{\mathsf{coll}}\right] = 1/2$$

Plugging this into (3.7) yields

$$\Pr[X_3] = 1/2 \cdot \Pr[\mathsf{coll}] + 1/2 \cdot (1 - \Pr[\mathsf{coll}]) = 1/2 \tag{3.8}$$

**Lower bound on $\varepsilon_\mathcal{B}$.** Finally, using (3.8), the bounds from Lemmas 3.6 and 3.7, and the fact that $\Pr[X_0] = \Pr[X_1]$, we obtain a lower bound on $\varepsilon_\mathcal{B}$:

$$\begin{aligned} 1/2 + \varepsilon_\mathcal{A} = \Pr[X_0] = \Pr[X_1] \leq \Pr[X_2] + \varepsilon_\mathcal{B} \leq 1/2 + \varepsilon_\mathcal{A}/2 + \varepsilon_\mathcal{B} \\ \iff \varepsilon_\mathcal{B} \geq \varepsilon_\mathcal{A}/2 \end{aligned}$$

Furthermore, by Lemma 3.6, algorithm $\mathcal{B}$ runs in time $t_\mathcal{B} = \Theta(t_\mathcal{A})$ and issues $Q$ oracle queries. □

### 3.4.3 Proof of Lemma 3.6

Adversary $\mathcal{B}$ plays the pseudorandomness security experiment with function $\hat{F}^j$. Let $\mathcal{O}$ denote the PRF oracle provided to $\mathcal{B}$ in this game. $\mathcal{B}$ runs $\mathcal{A}$ as a subroutine by simulating the security experiment as follows.

**Initialization.** $\mathcal{B}$ samples a bit $b \leftarrow \{0,1\}$, a hash function $h \leftarrow \mathcal{H}_{n,m}$, and picks $(s_{j+1}, ..., s_m)$, where $s_i \leftarrow S$ for all $i = j+1, \dots, m$

**Handling of oracle queries.** Whenever $\mathcal{A}$ queries $x \in \{0,1\}^n$, $\mathcal{B}$ proceeds as follows.

- If $b = 0$, then $\mathcal{B}$ proceeds exactly like the original experiment. That is, it responds with $R(x)$, where $R : \{0,1\}^n \to K$ is a random function.
- If $b = 1$, then $\mathcal{B}$ computes $h(x)$ and queries $\mathcal{O}$ to obtain $k_x := \mathcal{O}(h(x)_{1:j})$. Then it computes
$$z := \hat{F}^{m-j}((s_{j+1}, ..., s_m), k_x, h(x)_{j+1:m})$$
and returns $z$ to $\mathcal{A}$.

**Finalization.** Finally, when $\mathcal{A}$ terminates, then $\mathcal{B}$ outputs whatever $\mathcal{A}$ outputs, and terminates.

**Analysis of $\mathcal{B}$.** Note that the running time of $\mathcal{B}$ is essentially identical to the running time of $\mathcal{A}$ plus a minor number of additional operations, thus we have $t_{\mathcal{B}} = \Theta(t_{\mathcal{A}})$. If $\mathcal{O}(x) = \hat{F}^j((s_1, ..., s_j, k), h(x)_{1:j})$, then by Lemma 3.4 it holds that $z = \hat{F}^m((s_1, ..., s_m, k), h(x))$. Thus, the view of $\mathcal{A}$ is identical to Game 1. If $\mathcal{O}(x)$ implements a random function, then its view is identical to Game 2. This yields the claim.

### 3.4.4 Proof of Lemma 3.7

In order to show that $\Pr[\mathsf{coll}] \leq \varepsilon_{\mathcal{A}}/2$, we prove that all queries of $\mathcal{A}$are independent of $h$, regardless of $b = 0$ or $b = 1$, until $\mathsf{coll}$ occurs. This allows us to derive an upper bound on $\mathsf{coll}$. Consider the sequence of queries $x_1, \dots, x_Q$ made by $\mathcal{A}$. Recall that we assume $x_u \neq x_v$ for $u \neq v$ without loss of generality.

**The case $b = 0$.** In this case, $\mathcal{O}_3(x_i)$ is a random function $R(x_i)$, and therefore all information observed by $\mathcal{A}$is independent of $h$, until $\mathsf{coll}$ occurs. Thus, the view of $\mathcal{A}$is equivalent to a world in which the experiment does not choose $h$ at the beginning, but only after $\mathcal{A}$has made all queries, and only then computes $h(x_i)_{1:j}$ for all $i = 1, \dots, Q$ and outputs a random bit if a collision occurred. By the almost-universality, we thus obtain that

$$\Pr[\mathsf{coll} \mid b = 0] \leq \sum_{i=2}^{Q} \frac{i-1}{2^{j-1}} \leq \frac{Q^2}{2^j} \leq \frac{Q^2 \varepsilon_{\mathcal{A}}}{2Q^2} = \frac{\varepsilon_{\mathcal{A}}}{2}.$$

Note that we use here that $j \geq \log(2Q^2/\varepsilon_{\mathcal{A}})$, which holds due to the definition of $j$ in (3.5).

**The case** $b = 1$. We may assume without loss of generality that $Q > 0$, as otherwise $\mathcal{A}$ receives no information about $b$ and thus we would have $\varepsilon_{\mathcal{A}} = 0$. Consider the first query $\mathcal{O}_3(x_1)$ of $\mathcal{A}$. The oracle proceeds as follows. At first it computes $k_{x_1} := R_j(h(x_1)_{1:j})$. Since $R_j$ is a random function, this value is independent of $h$. In the next step it computes $z_1 := \hat{F}^{m-j}((s_{j+1}, ..., s_m), k_{x_1}, h(x_1)_{j+1:m})$, which is still uniformly random. To see this, note that the perfect one-time security of $F$ guarantees perfect one-time security of $\hat{F}^{m-j}$ as shown in Lemma 3.5. Thus $\mathcal{A}$ gains no information about $h$ at this point and the next query cannot be adaptive with regard to $h$.

Now if $\mathcal{A}$ queries $\mathcal{O}_3(x_2)$, then the experiment will evaluate the random functions $R_j$ on a different position than in the first query, unless

$$h(x_1)_{1:j} = h(x_2)_{1:j} \tag{3.9}$$

Due to the fact that the response to $x_1$ was independent of $h$ and the almost-universality of $h$, (3.9) happens with probability at most $1/2^{j-1}$. Therefore, again by the perfect one-time security of $F$, $\mathcal{A}$ receives another uniformly random value $z_2$, which is independent of $h$, except with probability at most $1/2^{j-1}$. Continuing this argument inductively over all $Q$ queries of $\mathcal{A}$, we see that on its $i$-th query $\mathcal{A}$ will receive a random response which is independent of $h$, except with probability $(i-1)/2^{j-1}$, provided that all previous responses were independent of $h$. A union bound now yields

$$\Pr[\mathsf{coll} \mid b = 1] \leq \sum_{i=2}^{Q} \frac{i-1}{2^{j-1}} \leq \frac{Q^2}{2^j} \leq \frac{Q^2 \varepsilon_{\mathcal{A}}}{2Q^2} = \frac{\varepsilon_{\mathcal{A}}}{2}.$$

It remains to show that $\Pr\left[X_3 \mid \overline{\mathsf{coll}}\right] = 1/2$. Let us consider the case $b = 1$. If $\overline{\mathsf{coll}}$ occurs, then there are no collisions, such that the oracle calls random function $R_j$ on always different inputs, each time receiving an independent, uniformly random value. Applying the perfect one-time security of $\hat{F}^{m-j}$ again, the response of the oracle to each query is therefore uniformly distributed and independent of all other queries. Thus, provided that no collision occurs, the view in case $b = 1$ is perfectly indistinguishable from the case $b = 0$, which yields the claim.

## 3.5 Applications

### 3.5.1 Efficient and Tightly-Secure PRF from Matrix Diffie-Hellman Assumptions

We recall the definition of the matrix Diffie-Hellman (MDDH) assumption and the pseudorandom function (PRF) from [EHK$^+$13a]. We consider a variant where an all-prefix almost-universal hash function is applied to the input before it is processed by the PRF. We note that the MDDH assumption generalizes the Decisional Diffie-Hellman (DDH) and Decisional $d$-Linear ($d$-LIN) assumptions, and, moreover, it gives us a framework to analyze the algebraic structure behind the Diffie-Hellman-based cryptographic primitives. Thus, our results can be carried on to the Naor-Reingold PRF (based on the DDH assumption) [NR97] and the Lewko-Waters PRF (based on the $d$-LIN assumption) [LW09].

**Notations and the MDDH Assumption.** Let $\mathcal{G} := (\mathbb{G}_1, [1]_1, q)$ be a description of multiplicative group $\mathbb{G}_1$ with random generator $[1]_1$ and prime order $q$. Following the "implicit notation" of [EHK+13b], we write $[a]_1$ shorthand for $[1]_1^a$. More generally, for a matrix $\mathbf{A} = (a_{ij}) \in \mathbb{Z}_q^{n\times m}$, we define $[\mathbf{A}]_1$ as the implicit representation of $\mathbf{A}$ in $\mathbb{G}_1$:

$$[\mathbf{A}]_1 := \begin{pmatrix} [a_{11}]_1 & \dots & [a_{1m}]_1 \\ \\ [a_{n1}]_1 & \dots & [a_{nm}]_1 \end{pmatrix} \in \mathbb{G}_1^{n\times m}.$$

Let us first recall the definition of the matrix Diffie-Hellman (MDDH) assumption [EHK+13b, EHK+13a].

**Definition 3.9. Matrix distribution.** Let $\ell, d \in \mathbb{N}$ and $\ell > d$. We call $\mathcal{D}_{\ell,d}$ a matrix distribution if it outputs matrices in $\mathbb{Z}_q^{\ell\times d}$ of full rank $d$ in polynomial time, namely, it is efficiently samplable. We define $\mathcal{D}_d := \mathcal{D}_{d+1,d}$.

Without loss of generality, we assume the first $d$ rows of $\mathbf{A} \leftarrow \mathcal{D}_{\ell,d}$ form a full-rank and invertible matrix, and we denote it by $\overline{\mathbf{A}}$ and the rest $\ell - d$ rows by $\underline{\mathbf{A}}$.

**Definition 3.10. Transformation matrix.** Let $\mathcal{D}_{\ell,d}$ be a matrix distribution and $\mathbf{A}$ be a matrix from it. The *transformation matrix* of $\mathbf{A}$ is defined as $\mathbf{T} := \underline{\mathbf{A}} \cdot \overline{\mathbf{A}}^{-1} \in \mathbb{Z}_q^{(\ell-d)\times d}$.

The $\mathcal{D}_{\ell,d}$-MDDH problem is to distinguish the two distributions $([\mathbf{A}]_1, [\mathbf{Aw}]_1)$ and $([\mathbf{A}]_1, [\mathbf{u}]_1)$ where $\mathbf{A} \leftarrow \mathcal{D}_{\ell,d}$, $\mathbf{w} \leftarrow \mathbb{Z}_q^d$ and $\mathbf{u} \leftarrow \mathbb{Z}_q^\ell$.

**Definition 3.11$\mathcal{D}_{\ell,d}$-Matrix Diffie-Hellman assumption, $\mathcal{D}_{\ell,d}$-MDDH.** [EHK+13b, EHK+13a] Let $\mathcal{D}_{\ell,d}$ be a matrix distribution. We say that adversary $\mathcal{A}$ $(t_\mathcal{A}, \varepsilon_\mathcal{A})$-breaks the $\mathcal{D}_{\ell,d}$-Matrix Diffie-Hellman ($\mathcal{D}_{\ell,d}$-MDDH) assumption in group $\mathbb{G}_1$, if $\mathcal{A}$ runs in time $t_\mathcal{A}$ and

$$|\Pr[\mathcal{A}(\mathcal{G}, [\mathbf{A}]_1, [\mathbf{Aw}]_1) = 1] - \Pr[\mathcal{A}(\mathcal{G}, [\mathbf{A}]_1, [\mathbf{u}]_1) = 1]| \geq \varepsilon_\mathcal{A},$$

where the probability is taken over $\mathbf{A} \leftarrow \mathcal{D}_{\ell,d}, \mathbf{w} \leftarrow \mathbb{Z}_q^d, \mathbf{u} \leftarrow \mathbb{Z}_q^\ell$.

**Examples of $\mathcal{D}_{\ell,d}$-MDDH.** [EHK+13b, EHK+13a] define distributions $\mathcal{L}_d$, $\mathcal{C}_d$, $\mathcal{SC}_d$, $\mathcal{IL}_d$, and $\mathcal{U}_d$ which corresponds to the $d$-Linear, $d$-Cascade, $d$-Symmetric-Cascade, $d$-Incremental-Linear, and $d$-Uniform assumption, respectively. All these assumptions are proven secure in the generic group model [EHK+13b, EHK+13a] and form a hierarchy of increasingly weaker assumptions.

A simple example is the $\mathcal{L}_1$-MDDH assumption for $d = 1$, which is the DDH assumption: Choose $a, w, z \leftarrow \mathbb{Z}_q$, and the DDH assumption states that the following two distributions are computationally indistinguishable:

$$([1, a, w, aw]_1) \approx_c ([1, a, w, z]_1).$$

This can be represented via the $\mathcal{L}_1$-MDDH assumption which states the following two distributions are computationally indistinguishable:

$$([\begin{smallmatrix} a \\ 1 \end{smallmatrix}]_1, [\begin{smallmatrix} aw \\ w \end{smallmatrix}]_1) =: ([\mathbf{A}]_1, [\mathbf{Aw}]_1) \approx_c ([\mathbf{A}]_1, [\mathbf{u}]_1) := ([\begin{smallmatrix} a \\ 1 \end{smallmatrix}]_1, [\begin{smallmatrix} z \\ w \end{smallmatrix}]_1).$$

For $d = 1$ the transformation matrix $\mathbf{T}$ contains only one element, and for $\mathcal{L}_1$-MDDH the corresponding transformation matrix is $\mathbf{T} = \frac{1}{a}$.

We give more examples of matrix distributions from [EHK$^+$13b, EHK$^+$13a] for $d = 2$ in Section 3.5.3.

**The PRF construction of [EHK$^+$13a] and its security.** Let $\mathcal{G} := (\mathbb{G}_1, P, q)$ be a description of a multiplicative group $\mathbb{G}_1$ with random generator $[1]_1$ and prime order $q$. Let $\mathcal{D}_{\ell,d}$ be a matrix distribution and we assume that $(\ell - d)$ divides $d$ and define $t := d/(\ell - d)$.

Following the approach of Section 5.3 of [EHK$^+$13a], we choose a random vector $\mathbf{h} \leftarrow \mathbb{Z}_q^d$, and, for $i = 1, ..., m$ and $j = 1, ..., t$, we choose $\mathbf{A}_{i,j} \leftarrow \mathcal{D}_{\ell,d}$ and compute transformation matrices $\hat{\mathbf{T}}_{i,j} := \underline{\mathbf{A}}_{i,j}\overline{\mathbf{A}}_{i,j}^{-1} \in \mathbb{Z}_q^{(\ell-d)\times d}$ and define the aggregated transformation matrices

$$\mathbf{T}_i := \begin{pmatrix} \hat{\mathbf{T}}_{i,1} \\ \vdots \\ \hat{\mathbf{T}}_{i,t} \end{pmatrix} \in \mathbb{Z}_q^{d\times d},$$

and $\mathbf{S} := (\mathbf{T}_1, ..., \mathbf{T}_m)$. Here, for $i \in \{1, ..., m\}$, we require that $\mathbf{T}_i$ has full rank. We note that this requirement can be satisfied by all the matrix distributions described in [EHK$^+$13a] with overwhelming probability. This implies the distribution of our $\mathbf{T}_i$'s is statistically close to that in [EHK$^+$13a], up to a negligibly small statistical distance of $1/(q - 1)$. Thus, their security results can be applied here.

Now let $S := \mathbb{Z}_q^{d\times d}$, $K := \mathbb{G}_1^d$, and $X := \{0, 1\}$. The basis of the PRF construction from [EHK$^+$13a] is the function $F_{\mathrm{MDDH}} : S \times K \times X \to K$ defined as

$$F_{\mathrm{MDDH}}(\mathbf{T}, [\mathbf{h}]_1, x) := \begin{cases} [\mathbf{h}]_1 & \text{if } x = 0 \\ [\mathbf{T} \cdot \mathbf{h}]_1 & \text{if } x = 1 \end{cases} \tag{3.10}$$

By applying the augmented cascade of Boneh *et al.* [BMR10] (Figure 3.1) to $F_{\mathrm{MDDH}}$, Escala *et al.* [EHK$^+$13a] obtain their PRF $F^m_{\mathrm{MDDH}}$ with key space $(\mathbb{Z}_q^{(d\times d)})^m \times \mathbb{G}_1^d$ and domain $\{0, 1\}^m$:

$$\begin{aligned} F^m_{\mathrm{MDDH}} &: (\mathbb{Z}_q^{(d\times d)})^m \times \mathbb{G}_1^d \times \{0,1\}^m \to \mathbb{G}_1 \\ F^m_{\mathrm{MDDH}}(\mathbf{S}, [\mathbf{h}]_1, x) &:= \left[\left(\prod_{i:x_i=1} \mathbf{T}_i\right) \cdot \mathbf{h}\right]_1 \end{aligned} \tag{3.11}$$

where $\mathbf{S} := (\mathbf{T}_1, ..., \mathbf{T}_m)$. The following theorem was proven in [EHK$^+$13b, EHK$^+$13a].

**Theorem 3.5** ([EHK$^+$13a, Theorem 12])**.** *From each adversary $\mathcal{A}$ that $(t_{\mathcal{A}}, \varepsilon_{\mathcal{A}}, Q)$-breaks the security of $F^m_{MDDH}$ with input space $\{0,1\}^m$ we can construct an adversary $\mathcal{B}$that $(t_{\mathcal{B}}, \varepsilon_{\mathcal{B}})$-breaks the $\mathcal{D}_{\ell,d}$-MDDH assumption in $\mathbb{G}_1$ with*

$$t_{\mathcal{B}} = \Theta(t_{\mathcal{A}}) \qquad \textit{and} \qquad \varepsilon_{\mathcal{B}} \geq \frac{\varepsilon_{\mathcal{A}}}{dm}$$

Note that $d$ is a constant, so that the security loss is linear in the size $m$ of the input space.

**Our construction.** By additionally encoding the input with an APUHF as described in (3.4), we finally obtain the function $F_{\text{MDDH}}^{\mathcal{H}_{n,m}} : S^m \times K \times \mathcal{H}_{n,m} \times \{0,1\}^n \to K$ as

$$F_{\text{MDDH}}^{\mathcal{H}_{n,m}}(\mathbf{S}, [\mathbf{h}]_1, h, x) = F_{\text{MDDH}}^{m}(\mathbf{S}, [\mathbf{h}]_1, h(x)) = \left[\left(\prod_{i:h(x)_i=1}^{m} \mathbf{T}_i\right) \cdot \mathbf{h}\right]_1 \tag{3.12}$$

In order to apply Theorem 3.4 to show that this particular instance of the augmented cascade with encoded input is a secure PRF with key space $S^m \times K \times \mathcal{H}_{n,m}$ and domain $\{0,1\}^n$, we merely have to prove that function $F_{\text{MDDH}}$ is perfectly one-time secure.

**Lemma 3.8.** *Function $F_{MDDH}$ from (3.10) is perfectly one-time secure.*

*Proof.* We have to show that

$$\Pr_{[\mathbf{h}]_1 \leftarrow \mathbb{G}_1^d}\left[F_{\text{MDDH}}(\mathbf{T}, [\mathbf{h}]_1, x) = [\mathbf{h}']_1\right] = \frac{1}{|\mathbb{G}_1|^d}.$$

for all $(\mathbf{T}, x, [\mathbf{h}']_1) \in S \times \{0,1\} \times \mathbb{G}_1^d$.
If $x = 0$ then $F_{\text{MDDH}}(\mathbf{T}, [\mathbf{h}]_1, 0) = [\mathbf{h}]_1$, which is a random vector in $\mathbb{G}_1^d$ by definition. If $x = 1$ then $F_{\text{MDDH}}(\mathbf{T}, [\mathbf{h}]_1, 1) = [\mathbf{Th}]_1$, which is again a random vector, due to the fact that $\mathbf{T}$ is a full-rank matrix. □

By combining Theorem 3.4 with Theorem 3.5 we now obtain the following result, which shows that setting $m = \omega(\log \lambda)$ is sufficient to achieve tight security.

**Theorem 3.6.** *Let $m = \omega(\log \lambda)$ be (slightly) super-logarithmic and $\mathcal{H}_{n,m}$ be a family of all-prefix almost universal hash functions. From each adversary $\mathcal{A}$ that $(t_{\mathcal{A}}, \varepsilon_{\mathcal{A}}, Q)$-breaks the security of $F_{\text{MDDH}}^{\mathcal{H}_{n,m}}$ with $Q/\varepsilon_{\mathcal{A}} = \mathsf{poly}(\lambda)$ for some polynomial $\mathsf{poly}$ we can construct an adversary $\mathcal{B}'$ that $(t'_{\mathcal{B}}, \varepsilon'_{\mathcal{B}})$-breaks the $\mathcal{D}_{\ell,d}$-MDDH assumption in $\mathbb{G}_1$ with*

$$t'_{\mathcal{B}} = \Theta(t_{\mathcal{A}}) \qquad \textit{and} \qquad \varepsilon'_{\mathcal{B}} \geq \frac{\varepsilon_{\mathcal{A}}}{2dj}$$

*where $j = O(\log \lambda)$.*

*Proof.* Theorem 3.4 shows that from each adversary $\mathcal{A}$ that $(t_{\mathcal{A}}, \varepsilon_{\mathcal{A}}, q)$-breaks the pseudorandomness of $F_{\text{MDDH}}^{\mathcal{H}_{n,m}}$ with $Q/\varepsilon_{\mathcal{A}} = \mathsf{poly}(\lambda)$ for some polynomial $\mathsf{poly}$, we can construct an adversary $\mathcal{B}$ that $(t_{\mathcal{B}}, \varepsilon_{\mathcal{B}}, Q)$-breaks the pseudorandomness of the function $F_{\text{MDDH}}^{j}$ with input space $\{0,1\}^j$, where

$$j = O(\log \lambda) \qquad \text{and} \qquad t_{\mathcal{B}} = \Theta(t_{\mathcal{A}}) \qquad \text{and} \qquad \varepsilon_{\mathcal{B}} \geq \varepsilon_{\mathcal{A}}/2$$

Theorem 3.5 in turn shows that from each adversary $\mathcal{B}$that $(t_{\mathcal{B}}, \varepsilon_{\mathcal{B}}, Q)$-breaks the security of $F_{\text{MDDH}}^{j}$ we can construct an adversary $\mathcal{B}$' that $(t'_{\mathcal{B}}, \varepsilon'_{\mathcal{B}})$-breaks the $\mathcal{D}_{\ell,d}$-MDDH assumption in $\mathbb{G}_1$ with

$$t'_{\mathcal{B}} = \Theta(t_{\mathcal{B}}) \qquad \text{and} \qquad \varepsilon'_{\mathcal{B}} \geq \frac{\varepsilon_{\mathcal{B}}}{dj} \geq \frac{\varepsilon_{\mathcal{A}}}{2dj}$$

which yields the claim. □

**Comparison to the DDH-based PRF of [NR97].** One particularly interesting instantiation of $F^m_{\mathrm{MDDH}}$ is based on the $\mathcal{L}_1$-MDDH assumption, which is an improvement over the famous Naor-Reingold construction based on the DDH (namely, $\mathcal{L}_1$-MDDH) assumption from [NR97]. In $F^m_{\mathrm{MDDH}}$, we sample $\mathbf{A}_i$ from $\mathcal{D}_{\ell,d}$ and then compute the aggregated transformation matrices $\mathbf{T}_i$. For the $\mathcal{L}_1$ distribution, we can equivalently pick random elements $T_i$ from $\mathbb{Z}_q$.

Let $\mathbb{G}_1$ be a group of prime order $q$, $S := \mathbb{Z}_q$, $K := \mathbb{G}_1$, $X := \{0,1\}^n$ and $m = \omega(\log\lambda)$ as above. Then we choose $T_1, ..., T_m, a \leftarrow \mathbb{Z}_q$ and obtain a PRF with domain $\{0,1\}^n$ as

$$F^{\mathcal{H}_{n,m}}_{\mathrm{DDH}}(\mathbf{S}, [a]_1, h, x) = \left[ \left( \prod_{i:h(x)_i=1}^{m} T_i \right) \cdot a \right]_1 .$$

Note that the resulting PRF is identical to the original Naor-Reingold function [NR97], except that an APUHF $h$ is applied to the input $x$ before it is processed in the Naor-Reingold construction. For the original construction from [NR97] both the size of the secret key and the tightness loss of the security proof (based on the DDH assumption in $\mathbb{G}_1$) are *linear* in the bit-length of the function input. We show that merely by encoding the input with an APUHF one can obtain shorter secret keys of size $m = \omega(\log\lambda)$ and with security loss $O(\log\lambda)$ (based on the same assumption as [NR97]), even for input size $n \gg m$.

**Comparison to the Matrix-DDH PRF of [DS15].** Döttling and Schröder [DS15] also described a variant of the Matrix-DDH-based PRF of [EHK+13b]. Their PRF is the function

$$F^{\mathsf{DS15}}_{\mathrm{MDDH}}(\mathbf{S}, [\mathbf{h}]_1, x) := \left[ \left( \prod_{j=1}^{m} (\mathbf{T}_i + x^{2^j} \cdot \mathbf{I}) \right) \cdot \mathbf{h} \right]_1 \tag{3.13}$$

where $\mathbf{S}$, $[\mathbf{h}]$, and $m$ are as in our construction, and $x \in \mathbb{Z}_q$. Thus, in comparison, our construction from (3.12) uses the same value of $m$, but is somewhat simpler that (3.13) and also slightly more efficient to evaluate. In particular, the computation of the terms of the form $(x^{2^j} \cdot \mathbf{I})$ is replaced with a single evaluation of the APUHF $h$. Another difference is that the domain of their function is restricted to $x \in \mathbb{Z}_q$, while in our case $x \in \{0,1\}^n$ can be any bit string of polynomially-bounded length $n = n(\lambda)$.

## 3.5.2 More Efficient LWE-based PRFs

We recall the learning with error (LWE) assumption. Then we apply our results to the LWE-based PRF from Banerjee, Peikert and Rosen [BPR12].

**Definition 3.12. Learning With Errors assumption, LWE** Let $p$ be a modulus, $N$ be a positive integer, and $\chi_\alpha := D_{\mathbb{Z}_p,\alpha}$ be a Gaussian distribution with noise parameter $\alpha$. Let $\mathbf{h} \leftarrow \mathbb{Z}_p^N$ be a random vector. We say that adversary $\mathcal{A}$ $(t_\mathcal{A}, \varepsilon_\mathcal{A})$-breaks the $\mathrm{LWE}_{p,N,\alpha}$ assumption if it runs in time $t_\mathcal{A}$ and

$$|\Pr[\mathcal{A}(\mathbf{h}, \mathbf{h}^\top\mathbf{s} + e) = 1] - \Pr[\mathcal{A}(\mathbf{h}, u) = 1]| \geq \varepsilon_\mathcal{A},$$

where $\mathbf{h} \leftarrow \mathbb{Z}_p^N$, $\mathbf{s} \leftarrow \mathbb{Z}_p^N$, $e \leftarrow \chi_\alpha$ and $u \leftarrow \mathbb{Z}_p$.

Let $\lfloor \cdot \rceil$ be the rounding function, which rounds a real number to the largest integer which does not exceed it. Let $p \geq q$. For an element $h \in \mathbb{Z}_p$, we define the rounding function $\lfloor \cdot \rceil_q : \mathbb{Z}_p \to \mathbb{Z}_q$ as $\lfloor h \rceil_q := \lfloor (q/p)h \rceil$, and for a vector $\mathbf{h} \in \mathbb{Z}_p^N$, the rounding function $\lfloor \mathbf{h} \rceil_q$ is defined component-wise.

**The PRF construction of [BPR12] and its security.** Let $S := \chi_\alpha^{N \times N}$ and $K := \mathbb{Z}_p^N$, and $X := \{0,1\}$. We assume that $\mathbf{S} \in S$ has full rank. The basis of the PRF of [BPR12] is the function $F_{\mathrm{LWE}} : S \times K \times X \to K$,

$$F_{\mathrm{LWE}}(\mathbf{S}, \mathbf{h}, x) := \begin{cases} \mathbf{h} & \text{if } x = 0 \\ \mathbf{S} \cdot \mathbf{h} & \text{if } x = 1 \end{cases} \tag{3.14}$$

We apply a slightly different augmented cascade transformation in Figure 3.1 to obtain the PRF of [BPR12] with key space $(\chi_\alpha^{(N \times N)})^m \times \mathbb{Z}_p^N$ and domain $\{0,1\}^m$:

$$\begin{aligned} F_{\mathrm{LWE}}^m &: (\chi_\alpha^{(N \times N)})^m \times \mathbb{Z}_p^N \times \{0,1\}^m \to \mathbb{Z}_q \\ F_{\mathrm{LWE}}^m(\mathbf{S}, \mathbf{h}, x) &:= \left\lfloor \left( \prod_{i:x_i=1}^{m} \mathbf{S}_i \right) \cdot \mathbf{h} \right\rceil_q \end{aligned} \tag{3.15}$$

where $\mathbf{S} := (\mathbf{S}_1, ..., \mathbf{S}_m)$ and $\mathbf{h} \leftarrow \mathbb{Z}_p^N$. Different to Figure 3.1, we apply the rounding function on the output of Figure 3.1.

**Theorem 3.7** ([BPR12, Theorem 5.2]). *Let $\chi_\alpha = D_{\mathbb{Z},\alpha}$ be a Gaussian distribution with parameter $\alpha > 0$, let $m$ be a positive integer that denotes the length of message inputs. Define $B := m(C\alpha\sqrt{N})^m$ for a suitable universal constant $C$. Let $p, q$ be two moduli such that $p > q \cdot B \cdot N^{\omega(1)}$.*

*From each adversary $\mathcal{A}$ that $(t_\mathcal{A}, \varepsilon_\mathcal{A}, Q)$-breaks the security of $F_{\mathrm{LWE}}^m$ with input space $\{0,1\}^m$ (for an arbitrary positive integer $m$) we can construct an adversary $\mathcal{B}$that $(t_\mathcal{B}, \varepsilon_\mathcal{B})$-breaks the LWE$_{p,N,\alpha}$ assumption with*

$$t_\mathcal{B} = \Theta(t_\mathcal{A}) \qquad \text{and} \qquad \varepsilon_\mathcal{B} \geq \frac{\varepsilon_\mathcal{A}}{m \cdot N}$$

Note that $B$ is an important parameter, since it determines the size of the LWE modulus $p$ and contains the expensive term $N^{m/2}$, which is exponential in $m$. Thus, a smaller $m$ can give us a smaller $p$, which in turn yields a weaker LWE assumption and a much more efficient PRF. In the following, we apply our results to $F_{\mathrm{LWE}}^m$ to reduce $m$ from polynomial to logarithmic in security parameter $\lambda$.

**Our construction.** By additionally encoding the input with an APUHF as described in (3.4), we finally obtain $F_{\mathrm{LWE}}^{\mathcal{H}_{n,m}} : (\chi_\alpha^{(N \times N)})^m \times \mathbb{Z}_p^N \times \mathcal{H}_{n,m} \times \{0,1\}^m \to \mathbb{Z}_q^N$ as

$$F_{\mathrm{LWE}}^m(\mathbf{S}, \mathbf{h}, h(x)) := \left\lfloor \left( \prod_{i:h(x)_i=1}^{m} \mathbf{S}_i \right) \cdot \mathbf{h} \right\rceil_q \tag{3.16}$$

In order to apply Theorem 3.4 to show that this particular instance of the augmented cascade with encoded input is a secure PRF with key space $S^m \times K \times \mathcal{H}_{n,m}$ and domain $\{0,1\}^n$, we have to prove that function $F_{\mathrm{LWE}}$ is perfectly one-time secure.

**Lemma 3.9.** *Function $F_{LWE}$ from (3.14) is perfectly one-time secure.*

*Proof.* We have to show that

$$\Pr_{\mathbf{h} \leftarrow \mathbb{Z}_p}\left[F_{\mathrm{LWE}}(\mathbf{S}, \mathbf{h}, x) = \mathbf{h}'\right] = \frac{1}{p^N}.$$

for all $(\mathbf{S}, x, \mathbf{h}') \in S \times \{0,1\} \times \mathbb{Z}_p^N$.

If $x = 0$ then $F_{\mathrm{LWE}}(\mathbf{S}, \mathbf{h}, 0) = \mathbf{h}$, which is a random vector in $\mathbb{Z}_p^N$ by definition. If $x = 1$ then $F_{\mathrm{LWE}}(\mathbf{S}, \mathbf{h}, 1) = \mathbf{S} \cdot \mathbf{h}$, which is again a random vector, due to the fact that $\mathbf{S}$ is a full-rank matrix. □

We recall the following useful notations and corollary for the proof of Theorem 3.8 given below. We define an error sampling function $E : \{0,1\}^j \to \mathbb{Z}^N$ and for $x \in \{0,1\}^j$ and $j \in= 1, \ldots, m$ we define the randomized version of $F^j_{\mathrm{LWE}}$ as $\tilde{F}^j_{\mathrm{LWE}}(x) = \left(\prod_{i:x_i=1}^{j} \mathbf{S}_i\right) \cdot \mathbf{h} + E(x)$. The proof of Theorem 5.2 and Lemma 5.5 in [BPR12] show that $\tilde{F}^j_{\mathrm{LWE}}$ is pseudorandom based on the decisional LWE assumption and it holds that $F^m_{\mathrm{LWE}}(x) = \left\lfloor \left(\prod_{i>j \wedge x_i=1}^{m} \mathbf{S}_i\right) \cdot \tilde{F}^j_{\mathrm{LWE}}(x) \right\rceil_q$, except with negligible probability. We summarize this in the following corollary.

**Corollary 3.1.** *Let all the parameters be defined as in Theorem 3.7. There exists an efficiently randomized error sampling function $E : \{0,1\}^j \to \mathbb{Z}^N$, such that, from each adversary $\mathcal{A}$ that $(t_{\mathcal{A}}, \varepsilon_{\mathcal{A}}, Q)$-breaks the security of $\tilde{F}^j_{LWE}(x) = \left(\prod_{i:x_i=1}^{j} \mathbf{S}_i\right) \cdot \mathbf{h} + E(x)$ with input $x \in \{0,1\}^j$ (for $j \in \{1, \ldots, m\}$) we can construct an adversary $\mathcal{B}$that $(t_{\mathcal{B}}, \varepsilon_{\mathcal{B}})$-breaks the $LWE_{p,N,\alpha}$ assumption with*

$$t_{\mathcal{B}} = \Theta(t_{\mathcal{A}}) \qquad \text{and} \qquad \varepsilon_{\mathcal{B}} \geq \frac{\varepsilon_{\mathcal{A}}}{m \cdot N}.$$

*Moreover, except with probability $2^{-\Omega(N)}$, we have*

$$F^m_{LWE}(x) = \left\lfloor \left(\prod_{i>j \wedge x_i=1}^{m} \mathbf{S}_i\right) \cdot \tilde{F}^j_{LWE}(x) \right\rceil_q.$$

**Theorem 3.8.** *Let $m = \omega(\log \lambda)$ be (slightly) super-logarithmic and $\mathcal{H}_{n,m}$ be a family of all-prefix almost universal hash functions. Let $\chi_\alpha = D_{\mathbb{Z},\alpha}$ be a Gaussian distribution with parameter $\alpha > 0$, let $m$ be a positive integer denotes the length of message inputs. Define $B := m(C\alpha\sqrt{N})^m$ for a suitable universal constant $C$. Let $p, q$ be two moduli such that $p > q \cdot B \cdot N^{\omega(1)}$.*

*From each adversary $\mathcal{A}$ that $(t_{\mathcal{A}}, \varepsilon_{\mathcal{A}}, Q)$-breaks the security of $F^{\mathcal{H}_{n,m}}_{LWE}$ with $Q/\varepsilon_{\mathcal{A}} = \mathsf{poly}(\lambda)$ for some polynomial $\mathsf{poly}$ we can construct an adversary $\mathcal{B}'$ that $(t'_{\mathcal{B}}, \varepsilon'_{\mathcal{B}})$-breaks* *the* $\mathit{LWE}_{p,N,\alpha}$ *assumption with*

$$t'_{\mathcal{B}} = \Theta(t_{\mathcal{A}}) \qquad \text{and} \qquad \varepsilon'_{\mathcal{B}} \geq \frac{\varepsilon_{\mathcal{A}}}{2j \cdot N} - 2^{-\Omega(N)}$$

*where* $j = O(\log \lambda)$.

*Proof.* The proof is the same as the one for Theorem 3.4. The only difference is between Games 1 and 2. Here we do one intermediate game transition Game 1': We simulate $\mathcal{O}_1(x)$ by returning $F^m_{\mathrm{LWE}}(x) = \left\lfloor \left( \prod^m_{i>j \wedge x_i=1} \mathbf{S}_i \right) \cdot \tilde{F}^j_{\mathrm{LWE}}(x) \right\rceil_q$ and $\mathcal{O}_0$ by returning a random vector in $\mathbb{Z}_q^N$.

By the second statement of Corollary 3.1, the difference between Games 1 and 1' is bounded by the statistical difference $2^{-\Omega(N)}$. Moreover, the difference between Games 1' and 2 is bounded by the security of $\tilde{F}^j_{\mathrm{LWE}}$. By the first statement of Corollary 3.1 we can conclude the proof. □

**Comparison to the LWE PRF of [DS15].** Döttling and Schröder [DS15] describe a different variant of the BPR PRF. Their approach is to instantiate their Construction 1 with the BPR PRF and then obtain the following function

$$F^{\mathsf{DS15}}_{\mathrm{LWE}}(K, \mathbf{h}, x) = \bigoplus_{i=1}^{L} \bigoplus_{j=1}^{\lambda} F^{2^i}_{\mathrm{LWE}}(\mathbf{S}, \mathbf{h}, \mathsf{Bin}(j) || H_{2^i,j}(x))$$

where $L = \omega(\log \lambda)$, for each $j = 1, \ldots, \lambda$ the function $H_{2^i,j} : \{0,1\}^n \to \{0,1\}^{i+1}$ is chosen from a suitable universal hash function family with range $\{0,1\}^{i+1}$, and $\mathbf{S}$ is chosen the same as ours.

Compared with $F^{\mathsf{DS15}}_{\mathrm{LWE}}$, our variant has shorter secret keys: instead of having $L \cdot \lambda$ many hash functions, we only have a single one. In terms of computation efficiency, instead of running $H_i$ and $F^i_{\mathrm{LWE}}$ for $L \cdot \lambda$ times, we only run the hash function and $F^m_{\mathrm{LWE}}$ once.

### 3.5.3 Further Examples of Matrix Distributions

Let us recall some further examples for matrix distributions from [EHK+13b, EHK+13a] for completeness and self-containedness.

$$\mathcal{L}_2 : \mathbf{A} = \begin{pmatrix} a_1 & 0 \\ 0 & a_2 \\ 1 & 1 \end{pmatrix}, \quad \mathcal{C}_2 : \mathbf{A} = \begin{pmatrix} a_1 & 0 \\ 1 & a_2 \\ 0 & 1 \end{pmatrix}, \quad \mathcal{IL}_2 : \mathbf{A} = \begin{pmatrix} a_1 & 0 \\ 0 & a_1{+}1 \\ 1 & 1 \end{pmatrix},$$

$$\mathcal{SC}_2 : \mathbf{A} = \begin{pmatrix} a_1 & 0 \\ 1 & a_1 \\ 0 & 1 \end{pmatrix}, \quad \mathcal{U}_2 : \mathbf{A} = \begin{pmatrix} a_1 & a_2 \\ a_3 & a_4 \\ a_5 & a_6 \end{pmatrix},$$

where $a_1, \ldots, a_6 \leftarrow \mathbb{Z}_q$. The corresponding transformation matrices are as follow,

$$\mathcal{L}_2 : \mathbf{T} = (\frac{1}{a_1}, \frac{1}{a_2}), \quad \mathcal{C}_2 : \mathbf{T} = (\pm\frac{1}{a_1 a_2}, \mp\frac{1}{a_2}), \quad \mathcal{IL}_2 : \mathbf{T} = (\frac{1}{a_1}, \frac{1}{a_1+1})$$

$$\mathcal{SC}_2 : \mathbf{T} = (\pm\frac{1}{a_1^2}, \mp\frac{1}{a_1}), \quad \mathcal{U}_2 : \mathbf{T} = (\frac{a_4 a_5 - a_3 a_6}{a_1 a_4 - a_2 a_3}, \frac{a_1 a_6 - a_2 a_5}{a_1 a_4 - a_2 a_3}).$$

The advantage of $\mathcal{SC}_d$ and $\mathcal{IL}_d$ is that they can be represented by one group element and have the same security guarantee as the $d$-Linear assumption.

## 3.6 Discussions

**On the necessity of the "all-prefix" property.** A natural question that arises is whether the *"all-prefix"* property is really necessary, or whether it is sufficient to use a standard universal hash function with fixed output space $\{0,1\}^j$ instead. However, it turns out that it is indeed necessary. To see this, recall that $j$ depends on the particular values of $Q$ and $\varepsilon_{\mathcal{A}}$ of a particular given adversary, via the definition $j = \lceil \log(2Q^2/\varepsilon_{\mathcal{A}}) \rceil$ in (3.5). One may wonder why we set $j$ so precisely, depending on the given adversary, rather than simply choosing $j$ sufficiently large such that it would work for *any* efficient adversary. The purpose of this precise choice is to balance between two properties that we need to obtain *tight* security:

1. On the one hand, we need $j$ to be sufficiently large, such that the probability of a collision of (the $j$-bit prefix of) the universal hash function is sufficiently unlikely.
2. On the other hand, we have to keep $j$ short enough, in order to get a tight reduction.

This is why the value $j$ depends on the given adversary, specifically on the particular values of $Q$ and $\varepsilon_{\mathcal{A}}$. We stress that this particular choice of $j$ is only required in the security proof, but not in the PRF *construction* itself. That is, we do not simply fix $j$ to be the largest value of $j$ such that the collision probability is sufficiently small for any adversary, because then for certain adversaries $j$, could be "too large" such that the reduction would not be tight. Similarly, if we used a standard universal hash function with output length $j$, $j$ would also fixed to some specific value in the *construction* of the PRF, and thus would again make the PRF construction only tightly secure for certain adversaries that match this particular choice of $j$, but not necessarily for all efficient adversaries.

For example, using a standard UHF with $m = \omega(\log \lambda)$ suffices to bound the collision probaility, but this yields only super-logarithmic tightness, and thus would be worse than in the construction of Döttling and Schröder [DS15], while we achieve logarithmic tightness with an APUHF.

Hence, the important new feature that *all-prefix* universality provides is that it guarantees that a suitable choice of $j$ exists for *any* efficient adversary. This makes the construction independent of a particular class of adversaries that match a certain fixed value of $j$, while at the same time it ensures that the security proof depends *tightly* on the particularly given adversary. Hence, using an APUHF instead of a standard universal hash function is not just sufficient, but also necessary in order to capture all efficient adversaries and to keep the security proof tight.

We note that Döttling and Schröder [DS15] also use multiple instances of the underlying pseudorandom function, with increasing security, in order to achieve tightness. Essentially, we replace these multiple instances with a single instance, in combination with an all-prefix universal hash function. From an abstract high-level perspective, in our approach each prefix implicitly corresponds to one PRF instance of [DS15]. This makes our construction significantly more efficient.

**Conclusion.** We have introduced all-prefix (almost-)universal hash functions (APUHFs) as a tool to generically improve the augmented cascade construction of pseudorandom functions by Boneh, Montgomery, and Raghunathan [BMR10]. By generically applying an APUHF to the function input before processing it in the augmented cascade, we are able to reduce both the key size and the tightness of the security proof by one order of magnitude. We gave simple and very efficient constructions of such function families, based on the almost-universal hash function family of Dietzfelbinger *et al.* [DHKP97], which can be evaluated by essentially a single modular multiplication, and generically on pairwise-independent hash functions.

For the instantiation based on Matrix-DDH assumptions of [EHK+13b], which includes the classical constructions of Naor-Reingold [NR97] and the Lewko-Waters [LW09] as special cases, this yields asymptotically short keys consisting of only $\omega(\log \lambda)$ elements and tight security with loss only $O(\log \lambda)$. These parameters are similar to the respective constructions of Döttling and Schröder [DS15], but our instantiation is conceptually much simpler and slightly more efficient.

For the LWE-based instantiation based on Banerjee, Peikert and Rosen [BPR12] (BPR), we are able to reduce the required size of the LWE modulus $p$ from exponential to super-polynomial in the security parameter, which significantly improves efficiency and allows to prove security under a weaker LWE assumption. Again, the latter is similar to a result from [DS15], but we replace their relatively expensive generic construction, which requires to run $\lambda \cdot \omega(\log \lambda)$ instances of the BPR function in parallel, with a *single* instance plus an all-prefix almost-universal hash function.

We believe that APUHFs may have many further applications in cryptography beyond pseudorandom functions. This may include, for example, constructions of more efficient cryptosystems with tight provable security, such as digital signatures or public-key encryption schemes. In particular constructions using arguments similar to pseudorandom functions based on the augmented cascade, such as [CW13, GHKW16], seem to be promising targets.

CHAPTER 4

# Efficient Forward-Secure Threshold Signature and Public-Key Encryption Schemes

*The purpose of forward-secure threshold schemes is to mitigate the damage of secret key exposure. We construct a forward-secure threshold signature scheme and a forward-secure threshold encryption scheme. Both of them are based on bilinear pairings with groups of prime order. Compared to existing schemes, our schemes are much more efficient since they have a non-interactive key update procedure and also a non-interactive signing procedure and decryption procedure, respectively. Additionally, our schemes do not require a trusted dealer and have optimal resilience as well as small signatures and small ciphertexts, respectively. We prove our signature scheme EUF-CMA forward secure and our encryption scheme CCA forward secure, both against adaptive adversaries. Moreover, both schemes are robust against malicious adversaries. Our schemes are the first which achieve all of these and that can also be implemented on standardized elliptic curves.*
*The results in this chapter are my own work and are based on [Kur20a, Kur20b]. The articles are going to appear at ACISP 2020 and IWSEC 2020, respectively.*

## 4.1 Introduction

**Forward-secure threshold schemes.** In a standard digital signature or public-key encryption scheme, exposure of the secret key results in the adversary being able to sign or decrypt arbitrary messages and all hope of security is lost. There are different approaches to mitigate the damage due to an adversary that gains access to stored keys. Two of these approaches are *forward-secure* schemes and *threshold* schemes. Forward-secure schemes allow to evolve the secret key in regular time periods, while the public key remains fixed. Thus, every adversary with an outdated secret key cannot forge signatures or decrypt ciphers for time periods in the past. In an $(n, k)$-*threshold* scheme the secret key is distributed among $n$ shares and it requires the presence of at least $k+1$ key shares to restore the secret key, whereas any subset of $k$ key shares is insufficient. Due to the fact that forward security and thresholds improve security guarantees against secret key exposure in a different manner, their combination to *forward-secure thresholds(FST)* can even reinforce these guarantees. Given a forward-secure threshold signature scheme, an adversary that aims at forging a signature for time period $t$ has not only to compromise $k+1$ of the stored keys to gain sufficient key material but it also has to gain all of this key material until time period $t$ expires. Depending on the scheme, the adversary has to gain all of this key material even in one specific time period. Since the capacities of an adversary can be assumed to be restricted, this combination provides additional security. The same holds for forward-secure encryption schemes where an adversary needs to compromise $k + 1$ of the stored keys until time period $t$ expires in order to decrypt a cipher for time period $t$. Beyond the generic protection against key exposure there are further use cases of forward-secure threshold schemes. In general, every use case of thresholds applies also to forward-secure thresholds as the time component adds only security, and vice versa. However, one can also imagine systems where both properties are required. For instance, imagine a company $A$ which is going to sell one of its divisions to company $B$ if at least $k$ out of $n$ managers agree on the sale. If a competing company $C$ also wants to buy this division from $A$, it might try to forge a signed contract which says that $A$ sells to $C$. However, such a contract would only be valid if $A$ was still in possession of this division, that is if the time period of the forgery was prior to the time period of signing the original contract.

**Difficulties of constructing forward-secure threshold schemes.** Unfortunately, the added benefits of combining forward-secure and threshold schemes invoke some technical challenges. These challenges occur especially in regard to efficiency.

**Necessity of a trusted dealer.** There is a potential risk in relying on trusted dealers to distribute the secret key shares among all parties such that they should be avoided.

**Necessity of secret point-to-point connections.** Due to the design of the protocol, secret point-to-point connections might be required for key generation, key update or even signing(decrypting). Since these connections are expensive and might be prone to attacks they should be avoided.

**Communication rounds.** The efficiency of a scheme is crucially determined by the number of communication rounds. Whereas the key generation happens only once and is not a major issue, the costs for signing(decrypting) and especially key update play a much bigger role. In signing(decrypting), the optimum would be that valid signature(decryption) shares can be computed non-interactively. That is, a signature(decryption) is requested and the signature(decryption) shares are sent back without further interaction among the parties. Thus, there is only one communication round and no overhead. In literature, such schemes are also called non-interactive threshold schemes [LJY16, LY13a, Sho00].[1] For key update, it is desirable to have no communication at all, i.e. a non-interactive procedure. Besides efficiency, the reason is that some key storage hosts might be offline or temporary unavailable. In the case of interactive key update this unavailability could block the update procedure entirely or exclude the unavailable parties from further signing procedures because of outdated key material.

**Sizes of keys, signatures, and ciphertexts** As for ordinary digital signature (PKE) schemes, the challenge to construct a scheme with small signatures(ciphertexts) and small

[1]The term "non-interactive" stems from the fact that users are able to deliver valid signature shares without interacting with each other. The communication round we count here is still there.

keys also arises and seems to be even harder. To illustrate the difficulties it suffices to take a look at naïve solutions: in terms of forward security, a naïve solution would be to generate and define a pair of public and secret key for each time period and then to delete the secret keys successively. Though in practice, this would yield huge parameters. For signature schemes, a naïve solution to create a $(n, k)$-threshold signature scheme would be to define a signature as valid if and only if at least $k + 1$ out of $n$ parties sign a message with a single user scheme. However, this would not only increase the total signature size but also leak which parties signed and which refused to do so. Consequently, both of these solutions, especially the combination of them, are highly undesirable. For this reason, more effort is required to construct a satisfactory solution.

**Our contribution.** We present a highly efficient *forward-secure threshold* signature scheme and a highly efficient *forward-secure threshold* encryption scheme. Both of our schemes are based on bilinear pairings, which can be implemented on standardized pairing-friendly curves with groups of prime order. More precisely, the schemes require *no trusted dealer*. Secret channels are only required during the key generation, i.e. there are *no secret channels for signing (decrypting) and key updating* required. They provide a *non-interactive key update and signing(decryption) procedure*. In addition, the schemes provide *short signatures* and *ciphertexts* of constant size. The schemes have *optimal resilience*, i.e., they can tolerate $(n - 1)/2$ compromised parties and are proven secure against adaptive and robust against malicious adversaries. We give a precise description of the adversary types in Section 4.2.1 and a precise comparison with previous work in Fig. 4.1 for the signature scheme and in Fig. 4.2 for the encryption scheme. Furthermore, it is possible to add pro-active security to our scheme. This enables security against mobile adversaries, i.e. against adversaries which can switch between the parties they corrupt. We discuss this concept and our techniques in Section 4.7.

**Technical approach.** At first glance, it might seem that it suffices to combine a key distribution protocol with an arbitrary forward-secure signature scheme and decryption scheme, respectively. However, this approach might result in an inefficient FST scheme, where the number of communication rounds in signing(decrypting) would be higher than in the secret extraction from the distribution protocol. Also, the key update procedure could be interactive, instead of non-interactive. Furthermore, it might be even impossible to simulate the security experiment in the security proof. In this case the security of the scheme could not be guaranteed.

Since similar arguments hold for our encryption scheme we focus here only on the signature scheme. We show a simply modified version of the distributed key generation (**DKG**) protocol by Gennaro et al. [GJKR07] which allows to transform the forward-secure single user signature scheme by Drijvers and Neven [DN19] into a highly efficient forward-secure threshold signature scheme. Basically, the reason for the efficiency of our FST scheme is that the scheme by Drijvers and Neven can be viewed as an aggregatable or key-homomorphic signature scheme. That is, if $(\sigma_1, \sigma_2)$ is a valid signature under public key $pk$ and $(\sigma'_1, \sigma'_2)$ is a valid signature under public key $pk'$, then $(\sigma_1\sigma'_1, \sigma_2\sigma'_2)$ is a valid signature under public key $pk \cdot pk'$. Additionally, the public and initial secret key are elements in different groups from a bilinear pairing: $g_2^x$ and $h^x$. These facts enable distribution of the initial secret $h^x$ into $n$ shares $h^{x_i}, i \in [n]$, which can be used to reconstruct $h^x$ via Lagrange interpolation (in the exponent). Subsequently, the partial signatures can be aggregated *non-interactively* into a signature for public key $g_2^x$.

More precisely, the single user scheme by Drijvers and Neven is defined over a bilinear pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$. The public key is defined as $g_2^x \in \mathbb{G}_2$, where $x \leftarrow \mathbb{Z}_q$. The initial secret key is $h^x \in \mathbb{G}_1$ and is frequently updated in a binary tree-based fashion like in the forward-secure encryption scheme from HIBE due to Canetti et al. [CHK03a].[2] In order to sign a message $m$ at time period $t$, we extract a secret key with the following structure:

$$(g_2^r, h^x \cdot F(t)^r, g_1^r) \in \mathbb{G}_2 \times \mathbb{G}_1 \times \mathbb{G}_1,$$

where $r \leftarrow \mathbb{Z}_q$ is picked uniformly at random and $F$ is a function depending on the time $t$. Then, the signature can be computed as

$$(h^x \cdot F(t)^{r+r'} \cdot g_1^{(r+r')\cdot m}, g_2^{r+r'}) = (h^x \cdot \tilde{F}(t,m)^{r+r'}, g_2^{r+r'}) =: (\sigma_1, \sigma_2) \in \mathbb{G}_1 \times \mathbb{G}_2,$$

where $r' \leftarrow \mathbb{Z}_q$ is picked uniformly at random and $\tilde{F}(t,m) := F(t) \cdot g_1^m$. For verification it is checked if the following equality holds:

$$e(\sigma_1, g_2) = e(h, g_2^x) \cdot e(\tilde{F}(t,m), \sigma_2).$$

In order to perform key distribution, we make use of the modified **DKG** protocol. In the original protocol all parties get a share $x_i \in \mathbb{Z}_q$ of the secret $x \in \mathbb{Z}_q$. However, in order to guarantee forward security, we have to define the initial secret as $h^x \in \mathbb{G}_1$ and for all $i \in [n]$ the initial secret shares as $h^{x_i} \in \mathbb{G}_1$ instead of $x$ and $x_i$, respectively. The initial secret shares are adapted in the binary tree-bases fashion and all values $h^{x_i}$ and $x_i$ have to be erased. We discuss the security of this modified version in Section 4.2.3.

For threshold $k$ and a set $\mathcal{V}$ of at least $k+1$ parties, the signature can then be computed as

$$\begin{aligned}(\sigma_1, \sigma_2) &= \left(\prod_{i\in\mathcal{V}} \sigma_{1,i}^{L_i} \ , \ \prod_{i\in\mathcal{V}} \sigma_{2,i}^{L_i}\right) = \left(\prod_{i\in\mathcal{V}} h^{x_i\cdot L_i} \cdot F(t,m)^{R_i\cdot L_i} \ , \ \prod_{i\in\mathcal{V}} g_2^{R_i\cdot L_i}\right) \\ &= \left(h^x \cdot F(t,m)^R \ , \ g_2^R\right),\end{aligned}$$

where the $L_i$ are interpolation coefficients and $R = \sum_{i\in\mathcal{V}} R_i \cdot L_i$. Indeed, this is a valid signature for public key $g_2^x$, which is dropped automatically by the **DKG** protocol.

It is worth mentioning the fact that during key update the initial secret key is not randomized in the exponent but in the "main" group $\mathbb{G}_1$, i.e. $h^x \cdot \tilde{R}$, where $\tilde{R}$ is a random value. This way of randomization enables a *non-interactive key update* since otherwise all parties had to secretly share the random exponent $r \in \mathbb{Z}_q$ in order to be able to interpolate their partial signatures. Additionally, the multiplication of two secrets in the exponent, $x \cdot r$, could require $2k+1$ instead of $k+1$ parties for reconstruction. For instance this is the case for polynomial sharing schemes like Shamir's secret sharing scheme. In terms of signing, it is easy to see that this forward-secure threshold scheme is *non-interactive*: getting a signing request for a message $m$, all parties $P_i$, $i = 1, \ldots, n$

[2]Here we describe only a simplified view. For more details see [CHK03a].

can either just broadcast their signature share $(\sigma_{1,i}, \sigma_{2,i})$ or deny signing. Then, the signature can be easily computed via Lagrange interpolation. The *robustness* against malicious behavior stems from the fact that the **DKG** protocol not only drops the common public key $g_2^x \in \mathbb{G}_2$ but also the individual public keys $g_2^{x_i} \in \mathbb{G}_2$. Thus, it is possible to check every individual signature share for validity:

$$\begin{aligned} e(\sigma_{1,i}, g_2) &= e(h, g_2^{x_i}) \cdot e(F(t,m), \sigma_{2,i}) \\ \Leftrightarrow e(h^{x_i} \cdot F(t,m)^{R_i}, g_2) &= e(h, g_2^{x_i}) \cdot e(F(t,m), g_2^{R_i}). \end{aligned}$$

**Related work.** The first works on threshold signature schemes are due to Boyd [BOY86] and Desmedt and Frankel [DF90]. In [LY13a], Libert and Yung construct a non-interactive threshold-signature scheme based on bilinear pairings with groups of composite order, which is secure against adaptive and robust against malicious adversaries.[3] Further, they suggest how a FST signature scheme with these properties could be built in composite order groups. However, when compared to groups of prime order groups of composite order must be very large in order to guarantee security. Large groups result in very expensive computation and much larger keys as well as signaturues and ciphertexts,respectvely. In [LJY16], Libert et al. construct a non-interactive threshold-signature scheme, which is secure against adaptive and robust against malicious adversaries and which requires only groups of prime order. The first forward-secure signature scheme is presented by Bellare and Miner [BM99] and later improved by Abdalla and Reyzin [AR00]. Krawczyk [Kra00] presents a general framework to make a signature scheme forward secure. However, the extension to FST signature schemes is restricted to eavesdropping adversaries and lacks efficiency. Drijvers and Neven [DN19] propose a forward-secure single user signature scheme and a forward-secure multi user signature scheme in the random oracle model [BR93b]. These schemes are based on bilinear pairings with groups of prime order. The first FST signature schemes are due to Abdalla et al. [AMN01] and Tzeng and Tzeng [TT01]. The latter is broken by Wang et al. [WQFX06]. In [LCT03], Liu et al. propose a forward-secure threshold scheme based on the standard single user signature scheme by Guillou and Quisquater [GQ90]. Unfortunately, Liu et al. do not provide a security proof for their scheme. Chow et al. [CGHY08] present a FST scheme based on the forward-secure scheme by Abdalla and Reyzin [AR00]. Yu and Kong [YK07] propose a FST signature scheme based on bilinear pairings with prime order groups. In Figure 4.1, we compare the existing FST signature schemes with our scheme in more detail. We only analyze the second scheme from Abdalla et al. since the first one requires all parties to be available for signing. The scheme by Liu et al. is omitted since it cannot be properly specified without a security proof. The scheme by Tzeng and Tzeng was broken and is omitted aswell.
For PKE, the combination of forward-secure and threshold mechanisms to a *forward-secure threshold PKE* (FST-PKE) was proposed by Libert and Yung [LY13b]. Their proposed scheme is CCA forward-secure against adaptive adversaries and robust against malicious adversaries. One drawback is that their scheme requires bilinear pairings with

[3]Adaptive adversaries can corrupt parties at any time. Static adversaries need to corrupt the parties before the protocol execution begins. Malicious adversaries can influence the output behavior of the corrupted parties. Eavesdropping can only read internal data. For a proper overview see Section 4.2.1.

| | [AMN01] | [CGHY08] | [YK07] | ours |
|---|---|---|---|---|
| Type | RSA | RSA | pairing | pairing |
| Trusted dealer | yes | yes | yes | no |
| Adversary type | mobile halting | adaptive malic. | static malic. | adaptive malic. |
| Key update | interactive | non-int. | interactive | non-int. |
| Compromised parties tolerated | $(n-1)/3$ | $(n-1)/2$ | $(n-1)/2$ | $(n-1)/2$ |
| Uncompr. parties for signing | $2k+1$ | $k+1$ | $k+1$ | $k+1$ |
| Communication rounds in signing | $2L$ | 2 | 4 +1 private | 1 |
| Number of signature elements | 3 | 3 | 3 | 3 |

Figure 4.1: $L$ denotes the security parameter, $n$ the number of parties, $k$ the threshold. The adversary types are described in Section 4.2.1. Malic. and non-int. abbreviate malicious and non-interactive, respectively. One communication round is defined as sending one message and getting one message back. If not specified, the communication rounds are broadcasts. The number of signature elements also includes one element for the time as it is needed for verification.

groups of composite order and a trusted dealer. The only FST-PKE based on pairings with groups of prime order is due to Zhang et al.[ZXZ13]. It is proven CCA forward secure against static and robust against malicious adversaries in the Random Oracle Model and requires a trusted dealer as well. In Figure 4.2. we compare these existing FST-PKE schemes with our scheme in more detail.

## 4.2 Thresholds and Key Distribution

In this section, we introduce the common typification of adversaries in threshold settings. This typification is crucial to derive concrete security guarantees from the security proof. Additionally, we introduce the required communication model for the key distribution protocol we use in our FST schemes and the key distribution protocol itself.

### 4.2.1 Adversary Types in the Threshold Setting.

We rephrase the adversary typification from [AMN01]. We assume a network of $n$ parties with a threshold $k < n$. The typification is arranged into two categories. The first category describes in which manner an adversary has to pick which parties it wants to compromise. We distinguish between the following manners:

- *static*: The adversary has to pick $k$ of the parties it wants to compromise before the first execution of a protocol. The remaining parties can be compromised at any time after this execution.

| | [LY13b] | [ZXZ13] | ours |
|---|---|---|---|
| Group order | composite | prime | prime |
| Trusted dealer | yes | yes | no |
| Adversary type | adaptive malic. | static malic. | adaptive malic. |
| Key update and decryption | non-int. | non-int. | non-int. |
| Compromised parties tolerated | $(n-1)/2$ | $(n-1)/2$ | $(n-1)/2$ |
| Uncompr. parties for decryption | $k+1$ | $k+1$ | $k+1$ |
| $\|pk\|$ | $\log T$ | $T$ | $\log T$ |
| $\|sk\|$ | $\log^2 T$ | $T$ | $\log^2 T$ |

Figure 4.2: $n$ denotes the number of parties, $k$ the threshold. The adversary types are described in Section 4.2.1. Malic. and non-int. abbreviate malicious and non-interactive, respectively. Note that [ZXZ13] requires the public key for updating the secret key, which leads to secret keys of size $T$ as well.

- *adaptive*: The adversary can pick which party it wants to compromise at any time during the execution of a protocol. Especially the adversary can decide which party it wants to compromise next based on information it gained so far.
- *mobile*: As adaptive but it can also switch from compromising one party to another one as long as it does not exceed the threshold during a time period. Whenever it compromises a party during a potential key update protocol this party is counted as compromised during both time periods: before and after the update.

The second category describes the power an adversary might have over a compromised party. We distinguish between the following adversaries:

- *eavesdropping*: The adversary learns all the secret information of a party but cannot change its behavior.
- *halting*: As eavesdropping but it can also stop this party from participating in the execution of protocols.
- *malicious*: As halting but it can also cause this party to divert from a protocol in any possible fashion.

Additionally, every type of adversary can listen to all broadcast communications. Both categories are arranged in order of increasing strength, i.e. the strongest adversary is mobile malicious. We prove our scheme secure against adaptive and robust against malicious adversaries. We explain in the discussions why mobile adversaries can generally not be tolerated while having a non-interactive key update procedure.

### 4.2.2 Communication Model.

We assume the existence of a broadcast channel between all parties in our protocol. Additionally, we assume the possibility to create private point-to-point connections between all parties during key generation. We work in a synchronous model, i.e. all parties are synchronized in time with a global clock. No trusted third party or dealer is required.

### 4.2.3 A Concrete Distributed Key Generation Protocol

Here, we introduce a modified variant of the distributed key generation (**DKG**) protocol by Gennaro et al.[GJKR07], which will be a major building block of our FST schemes. The basic idea behind the **DKG** protocol by Gennaro et al. [GJKR07] is to make use of Shamir's secret sharing scheme [Sha79] in the exponent of a cyclic group. In the original **DKG** protocol a set of $n$ parties generates a pair of secret and public key $(x, g^x) \in \mathbb{Z}_q \times \mathbb{G}$, where $\mathbb{G}$ is a multiplicative group of prime order $q$ and generated by $g$. The public output of the protocol provides not only the common public key $g^x$, but also user public keys $g^{x_i}$, where $x_i$ is the secret share of party $P_i$. This protocol does not require a trusted dealer and is robust against malicious behavior. We will make small modifications to the original **DKG** protocol to satisfy the requirements for forward security.

**Shamir's secret sharing.** Our FST schemes make use of Shamir's secret sharing technique [Sha79]. The idea behind Shamir's approach is to use the fact that it takes at least $k+1$ points to (re-)construct a polynomial of degree $k$, where $k$ points are not sufficient. An $(n, k)$-threshold is created as follows: The secret is defined as the constant term $x$ of a polynomial of degree $k$. The other coefficients $a_i$, $i = 1, \ldots, k$ are chosen uniformly at random. Let $a(Z) = x + a_1 Z + \cdots + a_k Z^k$. Then, for $i = 1, \ldots, n$ a share $(i, a(i))$ is secretly given to party $P_i$. Afterwards, any set of at least $k + 1$ parties can reconstruct the secret $s$ by interpolating the polynomial $a(0)$ which equals by definition $x$, whereas any set of at most $k$ parties cannot.

**Modifications to the DKG protocol.** We instantiate the protocol with the groups $\mathbb{G}_1$ and $\mathbb{G}_2$ from the bilinear pairing we will use in our FST schemes. Therefore, it provides $[x]_2 \in \mathbb{G}_2$, which serves as the public key in our FST schemes, as well as the values $[x_i]_2 \in \mathbb{G}_2$ for all parties $P_i$, which serve to verify the signature and decryption shares, respectively, and to provide robustness of the schemes. Furthermore, the secret is defined as $[hx]_1 \in \mathbb{G}_1$ for random $h, x \in \mathbb{Z}_q$ instead of simply $x \in \mathbb{Z}_q$ as in the original protocol. The corresponding secret key shares are $[hx_i]_1 \in \mathbb{G}_1$ instead of $x_i \in \mathbb{Z}_q$ and in order to fulfill the requirements of our FST schemes, they are adapted to a binary tree fashion. The initial secret key share for party $P_i$ is:

$$sk_{0,i} := ([1]_2^{r_i}, [hx_i]_1[h_0]_1^{r_i}, [h_1]_1^{r_i}, \ldots, [h_{\ell+1}]_1^{r_i}) \in \mathbb{G}_2 \times \mathbb{G}_1^{\ell+2},$$

where $r_i \leftarrow \mathbb{Z}_q$ is picked uniformly at random and where each value $[h_X]$, $X = 0, \ldots, \ell+1$ corresponds to one level of depth in a binary tree. The value $[1]_2^{r_i}$ is required to verify signatures. To guarantee forward security, it is crucial that the plain values $[hx_i]_1$ and $x_i$ for $i = 1, \ldots, n$ are erased from every storage.

**Protocol DKG$(n,k)$:**

**Generation of shared secret $x$:**

1. a) Each party $P_i$ chooses two random polynomials $a_i(Z)$ and $b_i(Z)$ over $\mathbb{Z}_q$ of degree $k$:

$$a_i(Z) = a_{i0} + a_{i1}Z + \cdots + a_{ik}Z^k \text{and}$$
$$b_i(Z) = b_{i0} + b_{i1}Z + \cdots + b_{ik}Z^k.$$

   b) $P_i$ computes and broadcasts $C_{is} = [1]_2^{a_{is}}[\tilde{h}]_2^{b_{is}} \in \mathbb{G}_2$ for $s = 0,\ldots,k$.

   c) $P_i$ computes $s_{ij} = a_i(j)$ and $s'_{ij} = b_i(j) \bmod q$ for $j = 1,\ldots,n$ and sends $s_{ij}, s'_{ij}$ secretly to $P_j$.

   d) For $i = 1,\ldots,n$ each party $P_j$ checks whether or not

$$[s_{ij}]_2[\tilde{h}]_2^{s'_{ij}} = \prod_{s=0}^{k}(C_{is})^{j^s}. \tag{4.1}$$

   If there is an $i \in [n]$ such that the check fails, $P_j$ broadcasts a *complaint* against $P_i$.

   e) If a dealer $P_i$ receives a complaint from $P_j$, he broadcasts the values $s_{ij}$ and $s'_{ij}$ satisfying Equation (4.1).

   f) Each party disqualifies any player that either received more than $k$ *complains* or answered to a complaint with values that does not satisfy Equation (4.1).

2. Each party defines the set $QUAL$, which indicates all non-disqualified parties.

3. The shared secret is defined as $[hx]_1 = [h\sum_{i\in QUAL} a_{i0}]_1 \in \mathbb{G}_1$. Each party $P_i$ computes $[hx_i]_1 := [h]_1^{\sum_{j\in QUAL} s_{ij}} \in \mathbb{G}_1$ and sets its initial share of this secret as

$$sk_{0,i} := ([1]_2^{r_i}, [hx_i]_1[h_0]_1^{r_i}, [h_1]_1^{r_i}, \ldots, [h_{\ell+1}]_1^{r_i}) \in \mathbb{G}_2 \times \mathbb{G}_1^{\ell+2},$$

   where $r_i \leftarrow \mathbb{Z}_q$ is picked uniformly at random.

   **Extracting $y := [x]_2 \in \mathbb{G}_2$:**

4. a) Each party $P_i, i \in QUAL$ computes and broadcasts the values $A_{is} = [a_{is}]_2 \in \mathbb{G}_2$ for $s = 0,\ldots,k$.

   b) For each $i \in QUAL$, each party $P_j$ checks whether or not

$$[s_{ij}]_2 = \prod_{s=0}^{k}(A_{is})^{j^s}. \tag{4.2}$$

   If there is an $i \in QUAL$ such that the check fails, $P_j$ *complaints* about $P_i$ by broadcasting $s_{ij}$ and $s'_{ij}$ that satisfy Eq. (4.1) but not Eq. (4.2).

   c) For all parties $P_i$ who received at least one valid complaint in this phase, the other parties run a reconstruction of $a_i(Z)$ and $A_{is}$ for $s = 0,\ldots,k$ in the clear, using the values $s_{ij}$.

   d) Each party computes the common public key as $y = \prod_{i\in QUAL} A_{i0} \in \mathbb{G}_2$ and the user public keys $pk_j$ as $[x_j]_2 = \prod_{i\in QUAL}\prod_{k=0}^{t}(A_{ik})^{j^k}$ for all $j \in [n]$. Except of all public keys and the initial secret share all parties erase all other values from their storage.

## 4.3 Forward-Secure Signature Schemes

In this section, we provide definitions of single user and threshold signature schemes with forward security as well as their security definitions. Eventuallly, we present the single user signature scheme with forward seucrity from [DN19], which is one building block of our FST signature scheme. The following definitions of forward-secure digital signatures and its security are due to Bellare and Miner [BM99].

**Definition 4.1. Forward-secure digital signatures.** A forward-secure digital signature scheme $\Sigma$ for $T$ time periods is defined as a quadruple of algorithms: $\Sigma = (\mathbf{KeyGen}, \mathbf{KeyUpdate}, \mathbf{Sign}, \mathbf{Verify})$, where:

- $\mathbf{KeyGen}(1^\lambda, T) \to (pk, sk_0)$. On input the security parameter and the maximum number of time periods $T$ it outputs a public key $pk$ and an initial secret key $sk_0$.
- $\mathbf{KeyUpdate}(sk_t) \to sk_{t+1}$. If the input is a secret key for a time period $t < T$, it outputs a secret key for the next time period $t + 1$ and deletes the input from its storage. Else it outputs $\perp$.
- $\mathbf{Sign}(t, sk_t, m) \to \sigma$. On input a time period $t$ with the corresponding secret key $sk_t$ and a message $m$ from the message space it outputs a signature $\sigma$ together with the time period $t$.
- $\mathbf{Verify}(pk, t, m, \sigma) \to b$, where $b \in \{0, 1\}$. If $\sigma$ is a valid signature for $m$ at time period $t$ and under public key $pk$, then **Verify** outputs 1, else 0.

The signing procedure is a protocol and contains of various steps: share-sign, share-verify, and combine. For simplicity we defined the input only as a time period and message and omit the key material held by all participating parties

**Definition 4.2. Correctness.** Let $(pk, sk_0)$ and $sk_t$ be the output of **KeyGen** and $\mathbf{KeyUpdate}(sk_i)$ for $i = 0, \ldots, t-1$, respectively. We call $\Sigma$ correct if for all messages $m$ from the message space and all time periods $t \in \{0, \ldots, T-1\}$ it holds that

$$\Pr[\mathbf{Verify}(pk, t, m, \mathbf{Sign}(t, sk_t, m))] = 1.$$

The EUF-CMA security for forward-secure signatures is similar to the one for standard digital signatures. In the case of forward-secure signatures, we allow the adversary not only to query signatures, but also to update the secret key to the next time period and to obtain the secret key at a time period of its choice. The adversary has two ways to win. First, by delivering a valid signature for a time period prior to the one of compromising the secret key. Second, by delivering a valid signature for an arbitrary time period if it never compromised the secret key. In both cases, the adversary cannot win by delivering a message and signature for a specific time period if it had queried a signature for this message during this time period. **EUF-CMA security.** The EUF-CMA security experiment for a forward-secure signature scheme $\Sigma$ is defined as follows: The challenger defines an (initially) empty set $\mathcal{S}$ and runs the $\mathbf{KeyGen}(1^\lambda, T)$ algorithm to obtain a pair of public and initial secret key $(pk, sk_0)$. Then, it sends the public key $pk$ to the adversary $\mathcal{A}$. The adversary has access to the following oracles:

- **KeyUpdate.** For all time periods $t < T - 1$ it updates the current time period $t$ and the secret key $sk_t$ to $t + 1$ and $sk_{t+1}$, respectively.
- **Signing**$(t, m)$**.** On input the current time period $t$ and a message $m$ it adds $(t, m)$ to the set $\mathcal{S}$ and runs **Sign**$(t, sk_t, M)$. Then, it returns the resulting signature $\sigma$ to $\mathcal{A}$.
- **Break-In.** The challenger records the break-in time $\tilde{t} \leftarrow t$ and sends the secret key $sk_{\tilde{t}}$ to $\mathcal{A}$. This oracle can only be queried once. Afterwards, **KeyUpdate** and **Sign** cannot be queried anymore.
- **Finalize**$(t^*, m^*, \sigma^*)$**.** If $(t^*, m^*) \in \mathcal{S}$, then it returns 0. If $\tilde{t}$ is defined and $t^* > \tilde{t}$, then it returns 0. If $\tilde{t}$ is defined and $t^* < \tilde{t}$, then it returns **Verify**$(pk, t^*, m^*)$. If $\tilde{t}$ is not defined (i.e. Break-In was never queried), then it returns **Verify**$(pk, t^*, m^*)$. Afterwards, the game terminates.

**Definition 4.3. EUF-CMA forward security.** Let $\mathcal{A}$ be an adversary playing the EUF-CMA security experiment for a forward-secure digital signature scheme $\Sigma$ for $T$ time periods. It $(t_{\mathcal{A}}, \varepsilon_{\mathcal{A}})$-breaks the EUF-CMA forward security of $\Sigma$ for $T$ time periods, if it runs in time $t_{\mathcal{A}}$ and

$$\Pr[\textbf{Finalize}(t^*, m^*, \sigma^*) = 1] \geqslant \varepsilon_{\mathcal{A}}.$$

### 4.3.1 A Concrete Single User Scheme

Here, we present the forward-secure digital signature scheme $\Sigma$ for $T = 2^\ell$ time periods by Drijvers and Neven [DN19]. It is defined via the following common parameters and algorithms:

- **Common parameters.** Let $\mathcal{M}$ be the message space and let $H : \mathcal{M} \to \{0, 1\}^\kappa$ be a hash function mapping messages to bits strings of length $\kappa$ such that $2^\kappa < q$. The common parameters consist of $\mathcal{PG}$, the description of a cryptographic Type-3 pairing group, the description of $H$, the maximum number of time periods $T = 2^\ell$ as well as random group elements $[1]_1, [h]_1, [h_0]_1, \ldots, [h_{\ell+1}]_1 \leftarrow \mathbb{G}_1$, and $[1]_2, [\tilde{h}]_2 \leftarrow \mathbb{G}_2$, where $h, h_0, \ldots, h_{\ell+1} \in \mathbb{Z}_q$ and where $[1]_1$ and $[1]_2$ are generators of $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively.
- **KeyGen**$(1^\lambda, T)$. Pick $x \leftarrow \mathbb{Z}_q$ uniformly at random. Then, compute the public key $pk := [x]_2$ and the initial secret key

  $$sk_0 := ([1]_2^r, [hx]_1[h_0]_1^r, [h_1]_1^r, \ldots, [h_{\ell+1}]_1^r) \in \mathbb{G}_2 \times \mathbb{G}_1^{\ell+2},$$

  where $r \leftarrow \mathbb{Z}_q$ is picked uniformly at random. Delete $x$ from the storage.
- **KeyUpdate**$(sk_t)$. We assume the time periods $0, \ldots, 2^\ell - 1$ as being organized as leaves of a binary tree of depth $\ell$, sorted in increasing order from left to right. That is, $000 \ldots 0$ is the first and $111 \ldots 1$ is the last time period. Further, we interpret the path from the root of the tree to a leaf node $t$ as binary representation $t = t_1 \ldots t_\ell$, where we take the left branch for $t_i = 0$ and the right one for $t_i = 1$. We proceed

in the same way for internal nodes $\omega = \omega_1 \dots \omega_s$, where $s < \ell$. Let $r \leftarrow \mathbb{Z}_q$ be picked uniformly at random. Then, we associate to each node $\omega$ a secret key:

$$\begin{aligned}(c, d, e_{s+1}, \dots, e_{\ell+1}) &= \left( [1]_2^r, [hx]_1 \big([h_0]_1 \prod_{v=1}^{s} [h_v]_1^{w_v}\big)^r, [h_{s+1}]_1^r, \dots, [h_{\ell+1}]_1^r \right) \\ &= \left( [r]_2, [hx + h_0 r + \sum_{v=1}^{s} h_v w_v r]_1, [h_{s+1} r]_1, \dots, [h_{\ell+1} r]_1 \right).\end{aligned}$$

Given such a secret key, we produce a secret key for a descendant node $\omega' = \omega_1 \dots \omega_{s'}$, where $s' > s$ as

$$\begin{aligned}&(c', d', e'_{s+1}, \dots, e'_{\ell+1}) \\ &= \left( c[1]_2^{r'}, d \prod_{v=s+1}^{s'} e_v^{w_v} \big([h_0]_1 \prod_{v=1}^{s'} [h_v]_1^{w_v}\big)^{r'}, e_{s'+1}[h_{s'+1}]_1^{r'}, \dots, e_{\ell+1}[h_{\ell+1}]_1^{r'} \right) \\ &= \Big( [r + r']_2, [hx + h_0 r + \sum_{v=1}^{s} h_v w_v r]_1 [\sum_{v=s+1}^{s'} h_v w_v r]_1 [h_0 r' + \sum_{v=1}^{s'} h_v w_v r']_1, \\ &[h_{s'+1}(r + r')]_1, \dots, [h_{\ell+1}(r + r')]_1 \Big) \\ &= \Big( [r + r']_2, [hx + h_0(r + r') + \sum_{v=1}^{s'} h_v w_v (r + r')]_1, \\ &[h_{s'+1}(r + r')]_1, \dots, [h_{\ell+1}(r + r')]_1 \Big),\end{aligned}$$

where $r' \leftarrow \mathbb{Z}_q$ is picked uniformly at random.
Let $C_t$ be the smallest subset of nodes that contains for each time period $t, \dots, T-1$ an ancestor or the leaf itself, but no nodes of ancestors or leafs for time periods $0, \dots, t-1$. For time period $t$ we define the secret key $sk_t$ as the set of the secret keys associated to all nodes in $C_t$. In order to update the secret key to time period $t+1$ determine $C_{t+1}$ and compute the secret keys for all nodes in $C_{t+1} \setminus C_t$. Afterwards, delete $sk_t$ and all used re-randomization exponents $r'$. [4]

- **Sign**$(t, sk_t, m)$. Let $M := H(m) \in \{0,1\}^\kappa$ be the hash value of message $m \in \mathcal{M}$ and $t_1 \dots t_\ell$ the bit representation of time period $t$. Derive the signing key $sk_t =$

[4] Example: Let $T = 2^3$. Then $t_0 = 000$, $t_1 = 100$, $t_2 = 010$,... . Given a substring $xy$, we can compute $xy0$ and $xy1$. Hence, for time period $t_2$ the set $C_{t_2}$ consists of the node keys for 01 and 1. From 01 it can compute the secret key for $t_2 = 010$ and $t_3 = 011$. From 1 it can compute the secret key for all time periods greater $t_3$: 100, 101, 110, 111. The keys for time periods $t_0 = 000$ and $t_1 = 001$ cannot be computed from this set. If we update to time period $t_3$, we need to compute 011 and erase the node key for 01. Thus $C_{t_3}$ consists of the key for 011 and the node key 1. Then, also the key for 010 cannot be computed anymore.

$(c, d, e_{\ell+1})$ and pick $r' \leftarrow \mathbb{Z}_q$. Then, compute and output

$$(\sigma_1, \sigma_2) := \left( d \cdot e_{\ell+1}^{M} \big([h_0]_1 \prod_{v=1}^{\ell} [h_v]_1^{t_v} [h_{\ell+1}]_1^{M}\big)^{r'}, c[1]_2^{r'} \right)$$
$$= \Bigg( [hx + h_0 r + \sum_{v=1}^{\ell} h_v t_v r]_1 [h_{\ell+1} r M]_1 [h_0 r' + \sum_{v=1}^{\ell} h_v t_v r' + h_{\ell+1} M r']_1$$
$$, [r + r']_2 \Bigg)$$
$$= \left( [hx + h_0(r + r') + \sum_{v=1}^{\ell} h_v t_v (r + r') + h_{\ell+1}(r + r')M]_1, [r + r']_2 \right).$$

- **Verify**$(pk, t, m, \sigma)$. On input $\sigma = (\sigma_1, \sigma_2) \in \mathbb{G}_1 \times \mathbb{G}_2$ message $m$, public key $pk$ and time period $t$ compute $M = H(m)$ and output 1 if

$$e(\sigma_1, [1]_2) = e([h]_1, pk) \cdot e([h_0]_1 \prod_{v=1}^{\ell} [h_v]_1^{t_v} [h_{\ell+1}]_1^{M}, \sigma_2),$$

else output 0.

**Remark 4.1.** *Note that we interpret the hashes as elements in $\mathbb{Z}_q$, which is possible since the image of our hash function is of size $2^{\kappa} < q$. Henceforth, we will describe a message only as $M$ and implicitly assume this to be the hash of a real message $m \in \mathcal{M}$, that is $M := H(m)$.*

This scheme was shown secure under a Type-3 pairing variant of the Bilinear Diffie-Hellman Inversion Assumption $\bar{q}$-BDHI* [BB04a, BBG05], where $\bar{q} = \ell + 1$ and $\ell = \log T$:

**Definition 4.4. $\bar{q}$-BDHI$_3^*$ Assumption.**
Let $\mathcal{PG} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, q)$ be the description of a cryptographic Type-3 pairing group and let $[1]_1$ and $[1]_2$ be random generators of $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively. Let $\mathcal{A}$ be an adversary. We say that it $(t_{\mathcal{A}}, \varepsilon_{\mathcal{A}})$-breaks the $\bar{q}$-BDHI$_3^*$ assumption, if it runs in time $t_{\mathcal{A}}$ and

$$\Pr[\mathcal{A}(\mathcal{PG}, [1]_1, [\alpha^1]_1, [\alpha^2]_1, \ldots, [\alpha^{\bar{q}}]_1, [1]_2) = e([1]_1, [1]_2)^{(\alpha^{\bar{q}+1})}] \geq \varepsilon_{\mathcal{A}},$$

where $\alpha \leftarrow \mathbb{Z}_q$.

**Theorem 4.1** ([DN19, Theorem 1])**.** *Let $\mathcal{A}$ be an adaptive adversary that $(t_{\mathcal{A}}, \varepsilon_{\mathcal{A}})$-breaks the EUF-CMA forward security of $\Sigma$ from [DN19] for $T$ time periods. Given $\mathcal{A}$, we can build an adversary $\mathcal{B}$ that $(t_{\mathcal{B}}, \varepsilon_{\mathcal{B}})$-breaks the $\bar{q}$-BDHI$_3^*$ assumption, where $\bar{q} = \ell + 1$ such that*

$$t_{\mathcal{B}} = O(t_{\mathcal{A}}) \text{ and } \varepsilon_{\mathcal{B}} \geq \frac{1}{T \cdot Q_H} \varepsilon_{\mathcal{A}} - \mathsf{negl}(\lambda),$$

*where $Q_H$ is the number of random-oracle queries and $\mathsf{negl}(\lambda)$ is negligible.*

## 4.4 Hierarchical Identity-Based Encryption Schemes (HIBE)

In this section, we provide definitions of hierarchical identity-based encryption (HIBE) schemes and their security. Eventuallly, we present the HIBE scheme from [BBG05], which is one building block of our FST encryption scheme. The following definitions of hierarchical identity-based encryption schemes and their security are reproduced from [CHK03b].

**Definition 4.5. Hierarchical identity-based encryption (HIBE).** A hierarchical identity-based encryption scheme (HIBE) $\Pi_{HIBE}$ of depth $\ell$ is defined via the following algorithms:

**Setup**$(1^\lambda, \ell) \rightarrow (pk, sk_0)$. On input the security parameter and the depth of the tree, it returns a master public key $mpk$ and a master secret key $sk_\varepsilon$.

**KeyDerive**$(id, sk_{id'}) \rightarrow sk_{id'}$. On input an identity $id$ and a secret key for identity $id'$, which is a prefix of $id$, it outputs a secret key for identity $id'$.

**Enc**$(id, pk, M) \rightarrow C$. On input an identity $id$, master public key $mpk$, and a message $M$, it outputs a cipher text $C$.

**Dec**$(id, sk_{id}, C) \rightarrow M$. On input an identity $id$, the corresponding secret key $sk_{id}$, and a ciphertext $C$, it outputs a message $M$.

**Definition 4.6. Correctness.** Let $(mpk, sk_\varepsilon) \leftarrow$**KeyGen** and $sk_{id} \leftarrow$ **KeyDerive**$(id, sk_{id'})$ for $id' = \varepsilon$ or any $id'$, which is a prefix of $id$. We call $\Pi_{HIBE}$ correct if for all messages $M$ and identities $id$ it holds that

$$\Pr[\mathbf{Dec}(id, sk_{id}, \mathbf{Enc}(id, mpk, M))] = 1.$$

**IND-ID-CPA security.** Security against chosen plaintext attacks (IND-ID-CPA) for an HIBE $\Pi_{HIBE}$ is defined by the following game between a challenger and an adversary $\mathcal{A}$. The challenger computes **Setup**$(\ell) \rightarrow (mpk, sk_\varepsilon)$ and sends $mpk$ to $\mathcal{A}$. The adversary has access to the following oracles.

- $\underline{\mathbf{KeyQuery}(id).}$ On input an identity $id$, the challenger computes and outputs the secret key $sk_{id}$.
- $\underline{\mathbf{Challenge}(id^*, M_0, M_1).}$ The adversary submits a challenge identity $id^*$ and two messages $M_0, M_1$. The challenger picks a bit $b$ uniformly at random and responds with a challenge ciphertext $C^* = \mathbf{HIBE.Enc}(id, M_b)$.
- $\underline{\mathbf{Guess}(b').}$ The adversary outputs its guess $b' \in \{0, 1\}$. The challenger outputs 1 if $b = b'$, else 0. The game stops.

The adversary is allowed to make multiple queries to **KeyQuery**$(id)$ and one query to **Challenge**$(id^*, m_0, m_1)$ in any order, but with the restriction that it does not query a key for $id^*$ or a prefix of it. **Guess**$(b')$ can only be queried after **Challenge**$(id, M_0, M_1)$.

**Definition 4.7.** Let $\mathcal{A}$ be an adversary playing the CPA security game for an HIBE $\Pi_{HIBE}$. It $(t_\mathcal{A}, \varepsilon_\mathcal{A})$-breaks the IND-ID-CPA security of $\Pi_{HIBE}$, if it runs in time $t_\mathcal{A}$ and

$$|\Pr[\mathbf{Guess}(b') = 1] - 1/2| \geq \varepsilon_\mathcal{A}.$$

### 4.4.1 A Concrete HIBE Scheme

Here, we present an instantiation of the HIBE encryption scheme $\Pi_{HIBE}$ by Boneh et al.[BBG05]. In general, this scheme allows the identities to be vectors $id = (t_1, \dots, t_s) \in \mathbb{Z}_q^s$, where $s = 1, \dots, \ell + 1$. In order to represent the time periods in our FST scheme, we restrict the identities at the first $\ell$ levels to single bits. The lowest level remains unrestricted. In our FST encryption scheme it will contain the hashed verification key of a strong one-time signature scheme. Overall, identities at depth $s = 1, \dots, \ell + 1$ are vectors $id = (t_1, \dots, t_s)$, such that $t_i \in \{0, 1\}$ for $i = 1 \dots \ell$ and $t_{\ell+1} \in \mathbb{Z}_q$. The message space is assumed to be $\mathbb{G}_3$. The scheme $\Pi_{HIBE}$ is defined via the following algorithms:

- **Common parameters.** The common parameters consist of $\mathcal{PG}$, the description of a cryptographic Type-3 pairing group as well as random group elements $[1]_1, [h]_1, [h_0]_1, \dots, [h_{\ell+1}]_1 \leftarrow \mathbb{G}_1$, and $[1]_2 \leftarrow \mathbb{G}_2$, where $h, h_0, \dots, h_{\ell+1} \in \mathbb{Z}_q$ and where $[1]_1$ and $[1]_2$ are generators of $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively.
- **Setup**$(1^\lambda, \ell)$. Pick $x \leftarrow \mathbb{Z}_q$ uniformly at random. Then, compute the master public key $pk := [x]_2$ and the master secret key $msk = [hx]_1$.
- **KeyDerive**$(sk_{t_1,\dots,t_s}, (t_{s+1}, \dots, t_{s'}))$. The secret key $sk_{t_1,\dots,t_s}$ for an identity of depth $s \leq \ell + 1$ is defined as

$$(c, d, e_{s+1}, \dots, e_{\ell+1}) = \left( [r]_2, [hx + (h_0 \prod_{v=1}^{s} h_v t_v) r)]_1, [h_{s+1} r]_1, \dots, [h_{\ell+1} r]_1 \right), \tag{4.3}$$

where $r \leftarrow \mathbb{Z}_q$ is picked uniformly at random.
Given the parent secret key $sk_{t_1,\dots,t_s}$, all descendant secret keys $sk_{t_1,\dots,t_s,\dots t_{s'}}$, where $s < s' \leq \ell + 1$ can be computed as

$$\begin{aligned}
&(c', d', e'_{s'+1}, \dots, e'_{\ell+1}) \\
&= \left( c[1]_2^{r'}, d \prod_{v=s+1}^{s'} e_v^{w_v} \left( [h_0]_1 \prod_{v=1}^{s'} [h_v]_1^{w_v} \right)^{r'}, e_{s'+1}[h_{s'+1}]_1^{r'}, \dots, e_{\ell+1}[h_{\ell+1}]_1^{r'} \right) \\
&= \left( [r + r']_2, [hx + h_0(r + r') + \sum_{v=1}^{s'} h_v w_v (r + r')]_1, \right. \\
&\left. [h_{s'+1}(r + r')]_1, \dots, [h_{\ell+1}(r + r')]_1 \right),
\end{aligned}$$

where $r' \leftarrow \mathbb{Z}_q$ is picked uniformly at random.
Having the master secret key $[hx]_1$, all secret keys can be computed directly.
- **Enc**$(pk, id, M)$. Let $M \in \mathbb{G}_3$ and $id = (t_1, \dots, t_s)$, where $s \in [\ell + 1]$. Compute and output

$$C = \left( e([h]_1, pk)^r \cdot M,\ [r]_2,\ [h_0 + \sum_{v=1}^{s} h_v t_v) r]_1 \right) \in \mathbb{G}_3 \times \mathbb{G}_2 \times \mathbb{G}_1$$

where $r \leftarrow \mathbb{Z}_q$ is picked uniformly at random.

- **Dec**$(id, sk_{id}, C)$. On input an identity $id = (t_1, \ldots, t_s)$, the corresponding secret key $sk_{id} = (c, d, e_{s+1}, \ldots, e_{\ell+1})$, and a ciphertext $C = (C_1, C_2, C_3)$ compute and output:

$$C_1 \cdot e(C_3, c)/e(d, C_2) = M. \tag{4.4}$$

**Correctness.** For a valid ciphertext and a valid secret key, we have

$$\begin{aligned}\frac{e(C_3, c)}{e(d, C_2)} &= \frac{e([(h_0 + \sum_{v=1}^{s} h_v t_v)r]_1, [r]_2)}{e([hx + (h_0 + \sum_{v=1}^{s} h_v t_v)r]_1, [r]_2)} &(4.5)\\ &= \frac{1}{e([hx]_1, [r]_2)} = \frac{1}{e([h]_1, npk)^r}, &(4.6)\end{aligned}$$

where $mpk = [x]_2$. Putting Eq. (4.5) in Eq. (4.4) gives us

$$C_1 \cdot \frac{1}{e([h]_1, pk)^r} = \frac{e([h]_1, pk)^r \cdot M}{e([h]_1, pk)^r} = M.$$

This scheme was shown secure under a Type-3 pairing variant of the Decisional Bilinear Diffie-Hellman Inversion Assumption $\bar{q}$-DBDHI* [BB04a, BBG05], where $\bar{q} = \ell + 1$:

**Definition 4.8. $\bar{q}$-DBDHI$_3^*$ Assumption.**
Let $\mathcal{PG} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, q)$ be the description of a cryptographic Type-3 pairing group and let $[1]_1$ and $[1]_2$ be random generators of $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively. Let $\mathcal{A}$ be an adversary. We say that it $(t_\mathcal{A}, \varepsilon_\mathcal{A})$-breaks the $\ell$-DBDHI$_3^*$ assumption, if it runs in time $t_\mathcal{A}$ and

$$\Big| \Pr\left[\mathcal{A}\left(\mathcal{PG}, [1]_1, [\alpha]_1, [\alpha^2]_1, \ldots, [\alpha^{\bar{q}}]_1, [1]_2, V_0\right) = 1\right] - \Pr\left[\mathcal{A}\left(\mathcal{PG}, [1]_1, [\alpha]_1, [\alpha^2]_1, \ldots, [\alpha^{\bar{q}}]_1, [1]_2, V_1\right) = 1\right] \Big| \geq \varepsilon_\mathcal{A}$$

where $\alpha \leftarrow \mathbb{Z}_q, V_0 = e([1]_1, [1]_2)^{\alpha^{\bar{q}+1}}$ and $V_1 \leftarrow \mathbb{G}_T$.

**Theorem 4.2** ([BBG05]). *Let $\mathcal{A}$ be an adaptive adversary that $(t_\mathcal{A}, \varepsilon_\mathcal{A})$-breaks the IND-ID-CPA security of $\Pi_{HIBE}$ from [BBG05] with depth $\ell + 1$. Given $\mathcal{A}$, we can build an adversary $\mathcal{B}$ that $(t_\mathcal{B}, \varepsilon_\mathcal{B})$-breaks the $\bar{q}$-DBDHI$_3^*$ assumption, where $\bar{q} = \ell + 1$ such that*

$$t_\mathcal{B} = O(t_\mathcal{A}) \text{ and } \varepsilon_\mathcal{B} \geq \frac{1}{2^{\ell+1}} \cdot \varepsilon_\mathcal{A}.$$

Note that Theorem 3.1 in [BBG05] refers to the selective security notion called IND-sID-CPA security. Here, we have the adaptive security notion IND-ID-CPA, which is implied by IND-sID-CPA with an exponential loss. For the concrete definition of IND-sID-CPA security see [BBG05].

## 4.5 Forward-Secure Threshold Signature Schemes

In this section, we introduce our FST signature scheme, which is much more efficient than the state of the art. This scheme is basically a threshold-version of the single user scheme of Drijvers and Neven [DN19]; see Sec. 4.3.1. We start with formal definitions of FST signature schemes and their security.
The first definition of FST signature schemes was introduced by Abdalla et al. [AMN01]. Here, we adopt the definition by Chow et al. [CGHY08]. The only difference between these two is that updating the secret key shares does not require an interactive protocol between all parties and can be accomplished by each party on its own.

**Definition 4.9. Forward-secure threshold signatures.** A forward-secure threshold signature scheme $(T, n, k)$-$\Sigma_{FST}$ for $T$ time periods, $n$ parties, and threshold $k$ is defined as a quadruple of the following components:

- $\mathbf{KeyGen}(1^\lambda, T, n, k) \rightarrow (pk, pk_1, sk_{0,1}, \dots, pk_n, sk_{0,n})$. On input the security parameter $\lambda$ in unary, the maximum number of time periods $T$, the total number of parties $n$, and the threshold $k$, this protocol produces a common public key $pk$ and user public keys $pk_i$ and initial secret key shares $sk_{0,i}$ for all parties $i \in [n]$. The secret shares are only known to the belonging parties.
- $\mathbf{KeyUpdate}(sk_{t,j}) \rightarrow sk_{t+1,j}$. If the input is a secret key share for a time period $t < T - 1$, it outputs a secret key share for the next time period $t + 1$ and deletes the input from its storage. Else it outputs $\perp$.
- $\mathbf{Sign}(t, m) \rightarrow (t, \sigma)$. If run by at least $k + 1$ honest and uncompromised signers on input the current time period $t$ and a message $m$ from the message space this protocol outputs a signature $\sigma$ together with the time period $t$.
- $\mathbf{Verify}(pk, t, m, \sigma) \rightarrow b$, where $b \in \{0, 1\}$. If $\sigma$ is a valid signature for $m$ at time period $t$ and under public key $pk$, then **Verify** outputs 1, else 0.

**Definition 4.10. Correctness.** Let $(pk, (pk_{0,i})_{i\in[n]}, (sk_{0,i})_{i\in[n]}) \leftarrow$**KeyGen** and $(sk_{t,i}) \leftarrow \mathbf{KeyUpdate}(sk_{j-1,i})$ for $j = 1, \dots, t$ and $i \in [n]$. We call $(T, n, k)$-$\Sigma_{FST}$ correct if for all messages $m$, all time periods $t \in \{0, \dots, T - 1\}$, and all subsets $\mathcal{U} \subseteq \{sk_{t,1}, \dots sk_{t,n}\}$ of size at least $k + 1$ held by uncorrupted parties it holds that

$$\Pr[\mathbf{Verify}(pk, t, m, \mathbf{Sign}(t, m)) = 1] = 1.$$

Note that the secret keys are an implicit input to **Sign**.

**EUF-CMA forward security.** EUF-CMA security against adaptive(static) [5] adversaries is defined by the following security experiment between a challenger and an adversary $\mathcal{A}$. Let $\mathcal{S}$ be an (initially) empty set, which denotes the signature queries and let $\mathcal{B}$ and $\mathcal{G}$ be the sets of indices, which denote the corrupted and uncorrupted parties, respectively. At the beginning $\mathcal{B}$ is empty and $\mathcal{G} = \{1, \dots, n\}$. The challenger (on behalf of the uncorrupted parties) and the adversary (on behalf of the corrupted parties) run the key

[5]In the static security experiment, the adversary has to decide which $k$ parties it wants to corrupt before **KeyGen** gets executed.

generation protocol $\mathbf{KeyGen}(1^\lambda, n, k, T)$. The adversary gets the common public key $pk$, all user public keys $(pk_i)_{i\in[n]}$ as well as the initial secret key shares of the corrupted parties $(sk_{0,j})_{j\in\mathcal{B}}$. The adversary has access to the following oracles:

- **Break-In**$(t, j)$. On input the current time period $t$ and index $j \in \mathcal{G}$ the challenger checks if $|\mathcal{B}| < k$. If this holds, the challenger removes $j$ out of $\mathcal{G}$ and adds it to $\mathcal{B}$. If $sk_{t,j}$ is already defined, it is delivered to $\mathcal{A}$. If $|\mathcal{B}| = k$, the challenger sets $\tilde{t} \leftarrow t$ and outputs $(sk_{\tilde{t},j})_{j\in\mathcal{G}}$. [6] Afterwards, this oracle cannot be queried anymore.
- **KeyUpdate.** For all time periods $t < T - 1$ it updates the current time period $t$ and the secret key shares $(sk_{t,j})_{j\in\mathcal{G}}$ to $t + 1$ and $sk_{t+1,j}$, respectively.
- **Signing**$(t, m)$**.** On input the current time period $t$ and a message $m$ from the message space it adds $(t, m)$ to the set $\mathcal{S}$. Then, the challenger (on behalf of the uncorrupted parties) and the adversary $\mathcal{A}$ (on behalf of the corrupted parties) run the signing protocol **Sign**. The output of this execution is delivered to $\mathcal{A}$.
- **Finalize**$(t^*, m^*, \sigma^*)$**.** If $(t^*, m^*) \in \mathcal{S}$, then it returns 0. If $\tilde{t}$ is defined and $t^* > \tilde{t}$, then it returns 0. If $\tilde{t}$ is defined and $t^* < \tilde{t}$, then it returns $\mathbf{Verify}(pk, t^*, m^*)$. If $\tilde{t}$ is not defined (i.e. Break-In was never queried), then it returns $\mathbf{Verify}(pk, t^*, m^*)$. Afterwards, the game terminates.

**Definition 4.11. EUF-CMA forward security.** Let $\mathcal{A}$ be an adaptive (a static) adversary playing the EUF-CMA security experiment for a forward-secure threshold signature scheme $(T, n, k)$-$\Sigma_{FST}$. It $(t_\mathcal{A}, \varepsilon_\mathcal{A})$-breaks the EUF-CMA security of $(T, n, k)$-$\Sigma_{FST}$, if it runs in time $t_\mathcal{A}$ and

$$\Pr[\mathbf{Finalize}(t^*, m^*, \sigma^*) = 1] \geqslant \varepsilon_\mathcal{A}.$$

We extend the robustness notion of threshold signature schemes by Gennaro et al.[GJKR07] to FST signature schemes. Essentially, the difference is the added **KeyUpdate** procedure.

**Definition 4.12. Robustness.** A forward-secure threshold signature scheme $\Sigma_{FST}$ is $(n, k_1, k_2)$-robust if in a group of $n$ parties, even in the presence of an adversary who halts up to $k_1$ and corrupts maliciously $k_2$ parties, **Keygen**, **KeyUpdate**, and **Sign** complete successfully.

### 4.5.1 New Forward-Secure Threshold Signature Scheme

Here, we introduce our FST signature scheme. We show how the modified **DKG** protocol from Fig. 4.3 allows to create a FST signature scheme, which is much more efficient than the state of the art. Eventually, we prove our FST scheme EUF-CMA forward secure against adaptive and robust against malicious adversaries.

[6] Note that this case is only possible for time periods $t > 0$, i.e. after **KeyGen** had finished. Else, the adversary would have no way to win the security experiment.

**The scheme.** For simplicity, we assume the common parameters to be given in our scheme. Note that this assumption does not contradict no need of a trusted dealer as such a dealer is not required for generation and sharing the public and secret key material. The common parameters could be either from a proposed standardization or commonly generated by all parties. For instance, this can be achieved by a coin-flipping protocol as in [BOGW88]. Let $(T, n, k)$-$\Sigma_{FST} = (\mathbf{KeyGen}, \mathbf{KeyUpdate}, \mathbf{Sign}, \mathbf{Verify})$ be a FST signature scheme defined as follows:

- **Common parameters.** Let $\mathcal{H}$ be a family of hash functions $H : \{0,1\}^* \to \mathbb{Z}_q$ mapping bits strings to elements in $\mathbb{Z}_q$. The common parameters consist of $\mathcal{PG}$, the description of a cryptographic Type-3 pairing group, the description of $H$ picked randomly from $\mathcal{H}$, the maximum number of time periods $T = 2^\ell$ as well as random group elements $[1]_1, [h]_1, [h_0]_1, \ldots, [h_{\ell+1}]_1 \leftarrow \mathbb{G}_1$, and $[1]_2, [\tilde{h}]_2 \leftarrow \mathbb{G}_2$, where $h, h_0, \ldots, h_{\ell+1}, \tilde{h} \in \mathbb{Z}_q$ and where $[1]_1$ and $[1]_2$ are generators of $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively.

- $\mathbf{KeyGen}(1^\lambda, T, n, k)$. The $n$ parties run the $\mathbf{DKG}(n, k)$ protocol from Figure 4.3. Subsequently each party $P_i$ holds an initial secret key share

$$sk_{0,i} := ([r_i]_2, [hx_i]_1[h_0r_i]_1, [h_1r_i]_1, \ldots, [h_{\ell+1}r_i]_1) \in \mathbb{G}_2 \times \mathbb{G}_1^{\ell+2}$$

  as well as the public keys $pk_j = [x_j]_2$ for all $j \in [n]$. The common public key $pk = [x]_2 \in \mathbb{G}_2$ is published.[7]

- $\mathbf{KeyUpdate}(sk_{t,i})$. We assume the time periods $0, \ldots, 2^\ell - 1$ are organized as leaves of a binary tree of depth $\ell$, sorted in increasing order from left to right. That is, $000\ldots0$ is the first and $111\ldots1$ is the last time period. Further, we interpret the path from the root of the tree to a leaf node $t$ as binary representation $t = t_1 \ldots t_\ell$, where we take the left branch for $t_z = 0$ and the right one for $t_z = 1$. We proceed in the same way for internal nodes $\omega = \omega_1 \ldots \omega_s$, where $s < \ell$. Let $r_i \leftarrow \mathbb{Z}_q$ be picked uniformly at random. Then, we associate to each party $P_i$ and each node $\omega$ a secret key:

$$(c_i, d_i, e_{i,s+1}, \ldots, e_{i,\ell+1}) = \left( [r_i]_2, [hx_i + h_0r_i + \sum_{v=1}^{s} h_v w_v r_i]_1, [h_{s+1}r_i]_1, \ldots, [h_{\ell+1}r_i]_1 \right).$$

  Given such a secret key, we produce a secret key for a descendant node $\omega' = \omega_1 \ldots \omega_{s'}$, where $s' > s$ as

$$\begin{aligned}&(c_i', d_i', e_{i,s'+1}', \ldots, e_{i,\ell+1}') \\ &= \left( c_i[1]_2^{r_i'}, d_i \prod_{v=s+1}^{s'} e_{i,v}^{w_v} ([h_0]_1 \prod_{v=1}^{s'} [h_v]_1^{w_v})^{r_i'}, e_{i,s'+1}[h_{s'+1}]_1^{r_i'}, \ldots, e_{i,\ell+1}[h_{i,\ell+1}]_1^{r_i'} \right)\end{aligned}$$

[7] Note that the secret share $x_i$ is computed commonly by all parties and the randomness $r_i$ is computed locally by party $P_i$ and is not a share of another random value. This approach is more efficient than computing random values commonly, especially to different bases.

$$= \Big( [r_i + r'_i]_2, [hx_i + h_0(r_i + r'_i) + \sum_{v=1}^{s'} h_v w_v (r_i + r'_i)]_1,$$
$$[h_{s'+1}(r_i + r'_i)]_1, \dots, [h_{\ell+1}(r_i + r'_i)]_1 \Big),$$

where $r'_i \leftarrow \mathbb{Z}_q$ is picked uniformly at random.
Let $C_t$ be the smallest subset of nodes that contains for each time period $t, \dots, T-1$ an ancestor or the leaf itself, but no nodes of ancestors or leafs for time periods $0, \dots, t-1$. For time period $t$, we define the secret key $sk_{t,i}$ of party $P_i$ as the set of its secret keys associated to all nodes in $C_t$. In order to update the secret key to time period $t+1$ it determines $C_{t+1}$ and computes the secret keys for all nodes in $C_{t+1} \setminus C_t$. Afterwards, it deletes $sk_{t,i}$ and all used re-randomization exponents $r'_i$.

- **Sign**$(t, m)$. Let $M := H(m)$ be the hash value of message $m \in \mathcal{M}$ and $t = t_1 \dots t_\ell$ the bit representation of time period $t$. Let $\mathcal{W}$ be the set of indices of all parties participating in signing $M$. W.l.o.g. we assume that $\mathcal{W}$ contains at least $k+1$ distinct indices. The participating parties run the signing protocol from Figure 4.4.[8]

[8] Note that signing happens with respect to a time period. Since time periods are encoded in full bit length (even if they start with zero) they are low in the binary tree. Hence, they only have left $e_{i,\ell+1}$ as going down one level in depth erases one value $e_{i,x}, x \in \{1, \dots \ell, \}$.

**Sign**$(t, M, P_1, ..., P_n)$**:**

1. **Share-Sign.** At request of a signature for message $M$ at time $t = t_1 \dots t_\ell$, each party $P_i, i \in \mathcal{W}$ signs $M$ under its signing key $(c_i, d_i, e_{i,\ell+1})$ derived from $sk_{t,i}$. The resulting signatures are

$$(\sigma_{1,i}, \sigma_{2,i}) = \left( d_i \cdot e_{i,\ell+1}^M \cdot [(h_0 + \sum_{v=1}^{\ell} h_v t_v + h_{\ell+1} M)^{r'_i}]_1, c_i \cdot [r'_i]_2 \right),$$

   where $r'_i \leftarrow \mathbb{Z}_q$ is picked uniformly at random. Each party $P_i, i \in \mathcal{W}$ sends its signature to all other parties.

2. **Share-Verify.** All parties in $\mathcal{W}$ use the public keys $pk_i, i \in \mathcal{W}$ to verify if all the received signatures are valid. That is,

$$e(\sigma_{1,i}, [1]_2) = e([h]_1, pk_i) \cdot e([h_0 + \prod_{v=1}^{\ell} h_v t_v + h_{\ell+1} M]_1, \sigma_{2,i}). \tag{4.7}$$

3. **Combine.** Let $\mathcal{V} \subseteq \mathcal{W}$ indicate a set of users sending valid signatures. Assuming $\mathcal{V}$ contains at least $k+1$ distinct indices. Each party can now compute the Lagrange coefficients $L_i$ and aggregate the signatures as follows:

$$(\sigma_1, \sigma_2) := (\prod_{i \in \mathcal{V}} \sigma_{1,i}^{L_i}, \prod_{i \in \mathcal{V}} \sigma_{2,i}^{L_i}).$$

Figure 4.4: The signing protocol of our FST scheme.

**Remark 4.2.** *Note that the set $\mathcal{V}$ might be of size greater than $k+1$, while $k+1$ partial signatures are sufficient to construct the final signature. Hence, the final signature is not unique if not every party takes all partial signatures for constructing it. However, all signatures constructed in Step 4 are nevertheless indistinguishable from a signature created in a single user protocol because it is easy to re-randomize both $\sigma_1$ and $\sigma_2$. Especially seeing two valid signatures $(\sigma_1, \sigma_2)$ and $(\sigma'_1, \sigma'_2)$ it cannot be deduced how many signers were involved in signing. For this reason we do not have to force the parties to aggregate all partial signatures and thus, avoid overhead in computation.*

- **Verify**$(pk, t, m, \sigma)$. On input $\sigma = (\sigma_1, \sigma_2) \in \mathbb{G}_1 \times \mathbb{G}_2$ message $m$, public key $pk$ and time period $t$ compute $M = H(m)$ and output 1 if

$$e(\sigma_1, [1]_2) = e([h]_1, pk) \cdot e([h_0]_1 \prod_{v=1}^{\ell} [h_v]_1^{t_v} [h_{\ell+1}]_1^M, \sigma_2) \tag{4.8}$$

else output 0.

### 4.5.2 Proof of Correctness.

In order to show that $(\sigma_1, \sigma_2)$ is a valid signature for message $M$ at time period $t$ under the common public key $pk = [x]_2$ we have to show first that we can determine the set $\mathcal{V}$, which indicates the correct signature shares. That is, we have to show that we can check whether the partial signatures $(\sigma_{1,i}, \sigma_{2,i})$, $i \in \mathcal{W}$ are correct. Let $(\sigma_{1,i}, \sigma_{2,i})$ be an honestly generated signature. Then,

$$\begin{aligned}
\sigma_{1,i} &= d_i e_{i,\ell+1}^M ([h_0]_1 \prod_{v=1}^{\ell} [h_v]_1^{t_v} [h_{\ell+1}]_1^M)^{r'_i} \\
&= [hx_i + h_0 r_i + \sum_{v=1}^{\ell} h_v t_v r_i]_1 [h_{i,\ell+1} r_i M]_1 \cdot [h_0 r'_i + \sum_{v=1}^{\ell} h_v t_v r'_i + h_{\ell+1} r'_i M]_1 \\
&= [hx_i + (h_0 + \sum_{v=1}^{\ell} h_v t_v + h_{i,\ell+1} M)(r_i + r'_i)]_1, \qquad (4.9)
\end{aligned}$$

where we used the fact that $e_{i,\ell+1} = [h_{i,\ell+1} r_i]_1$ and $\sigma_{2,i} = [r_i + r'_i]_2$. The signature $(\sigma_{1,i}, \sigma_{2,i})$ with $\sigma_{1,i}$ as in (4.9) and with $\sigma_{2,i}$ satisfies (4.7), since

$$\begin{aligned}
&e([hx_i + (h_0 + \sum_{v=1}^{\ell} h_v t_v + h_{i,\ell+1} M)(r_i + r'_i)]_1, [1]_2) \\
&= e([hx_i]_1, [1]_2) \cdot e([(h_0 + \sum_{v=1}^{\ell} h_v t_v + h_{i,\ell+1} M)(r_i + r'_i)]_1, [1]_2) \\
&= e([h]_1, [x_i]_2) \cdot e([h_0 + \sum_{v=1}^{\ell} h_v t_v + h_{i,\ell+1} M]_1, [r_i + r'_i]_2).
\end{aligned}$$

For this reason, we can indeed check whether a signature $(\sigma_{1,i}, \sigma_{2,i})$ for message $M$ at time period $t$ is valid under public key $[x_i]_2$ and thus include $i$ into set $\mathcal{V}$. It remains to show that all partial signatures $(\sigma_{1,i}, \sigma_{2,i})$, $i \in \mathcal{V}$ interpolate to a valid signature under the common public key $[x]_2$. For this purpose we set $R := \sum_{i \in \mathcal{V}} L_i (r_i + r'_i)$. Then,

$$\begin{aligned}
\sigma_1 = \prod_{i \in \mathcal{V}} \sigma_{1,i}^{L_i} &= \prod_{i \in \mathcal{V}} [hx_i + (h_0 + \sum_{v=1}^{\ell} h_v t_v + h_{\ell+1} M)(r_i + r'_i)]_1^{L_i} \\
&= [h \sum_{i \in \mathcal{V}} L_i \cdot x_i]_1 [(h_0 + \sum_{v=1}^{\ell} h_v t_v + h_{\ell+1} M)(\sum_{i \in \mathcal{V}} L_i (r_i + r'_i))]_1 \\
&= [hx]_1 [(h_0 + \sum_{v=1}^{\ell} h_v t_v + h_{\ell+1} M) R]_1,
\end{aligned}$$

where $\sum_{i \in \mathcal{V}} L_i \cdot x_i = x$. Furthermore,

$$\sigma_2 = \prod_{i \in \mathcal{V}} \sigma_{2,i}^{L_i} = [\sum_{i \in \mathcal{V}} L_i (r_i + r'_i)]_2 = [R]_2.$$

Overall, we have

$$
\begin{aligned}
e(\sigma_1, [1]_2) &= e([hx + (h_0 + \sum_{v=1}^{\ell} h_v t_v + h_{\ell+1} M)R]_1, [1]_2) \\
&= e([hx]_1, [1]_2) \cdot e([(h_0 + \sum_{v=1}^{\ell} h_v t_v + h_{\ell+1} M)R]_1, [1]_2) \\
&= e([h]_1, [x]_2) \cdot e([h_0 + \sum_{v=1}^{\ell} h_v t_v + h_{\ell+1} M]_1, \sigma_2),
\end{aligned}
$$

which shows that $(\sigma_1, \sigma_2)$ fulfills (4.8) and therefore is a valid signature for message $M$ at time period $t$ under public key $pk = [x]_2$.

## 4.5.3 Proof of Security.

As preparation for the security reduction we describe in Figure 4.5 the simulation of the key generation protocol **DKG** and in Figure 4.6 the simulation of the signing protocol **Sign**. The simulation of the **DKG** protocol is basically due to Gennaro et al. [GJKR07]. As the **DKG** protocol we instantiate the simulation with the groups $\mathbb{G}_1$ and $\mathbb{G}_2$ from the bilinear pairing we use in our FST signature scheme and make the same adjustments in order to enable forward security. We explain these adjustments in Sec. 4.2.3.

**Simulation of DKG.** During key generation, the adversary is allowed to control at most $k$ parties. Otherwise, it would gain access to the secret key at time period 0 and by definition it would have no possibility to win the security experiment of the signature scheme. First, we assume a static adversary as in the proof in [GJKR07]. Then, we show how to extend the security to adaptive adversaries. We explain in Section 4.7 why this approach is reasonable. W.l.o.g. we assume the compromised parties to be $P_1, \dots, P_k$. Let $\mathcal{B} := \{1, \dots, k\}$ indicate the set of parties controlled by the adversary $\mathcal{A}$ and let $\mathcal{G} := \{k+1, \dots, n\}$ indicate the set of honest parties, which are run by the simulator.

**Protocol DKGSim**$(y = [x]_2, n, k)$**:**

1. The simulator performs the Steps 1a-1f, 2, and, 3 on behalf of the uncorrupted parties exactly as in the **DKG**$(n, k)$ protocol. Additionally, it reconstructs the polynomials $a_i(Z), b_i(Z)$ for $i \in \mathcal{B}$. Then:
   - The set $QUAL$ is well-defined and $\mathcal{G} \subseteq QUAL$ and all polynomials are random for all $i \in \mathcal{G}$.
   - The adversary sees $a_i(Z), b_i(Z)$ for $i \in \mathcal{B}$, the shares $(s_{ij}, s'_{ij}) = (a_i(j), b_i(j))$ for $i \in QUAL$, $j \in \mathcal{B}$ and $C_{is}$ for $i \in QUAL$, $s = 0, \ldots, k$.
   - The simulator knows all polynomials $a_i(Z), b_i(Z)$ for $i \in QUAL$ as well as all shares $s_{ij}, s'_{ij}$, all coefficients $a_{is}, b_{is}$ and the public values $C_{is}$.
2. The simulator performs as folllows:
   - Computes $A_{is} = [a_{is}]_2 \in \mathbb{G}_2$ for $i \in QUAL \setminus \{n\}$, $s = 0, \ldots, k$.
   - Sets $A^*_{n0} = y \cdot \prod_{i \in QUAL \setminus \{n\}} (A^{-1}_{i0})$.
   - Sets $s^*_{nj} = s_{nj} = a_n(j)$ for $j = 1, \ldots, k$.
   - Computes $A^*_{ns} = (A^*_{n0})^{\lambda_{s0}} \cdot \prod_{i=1}^{k} ([s^*_{ni}]_2)^{\lambda_{si}} \in \mathbb{G}_2$ for $s = 1, \ldots, k$, where the $\lambda_{is}$s are the Lagrange interpolation coefficients.

   a) The simulator broadcasts $A_{is}$ for $i \in \mathcal{G} \setminus \{n\}$ and $A^*_{ns}$ for $s = 0, \ldots, k$.

   b) It performs for all uncorrupted parties the verification of (4.2) on the values $A_{ij}$ for $i \in \mathcal{B}$. In case of a fail it broadcasts a complaint $(s_{ij}, s'_{ij})$. Since the adversary controls at most $k$ parties and the simulator behaves honestly, only secret shares of corrupted parties can be reconstructed.

   c) Afterwards it performs the Steps 4c and 4d of the **DKG**$(n, k)$ protocol.

Figure 4.5: The simulation of the DKG protocol from Figure 4.3.

**SignSim**$(t, M, (\sigma_1, \sigma_2))$

1. The signature is requested. The simulator is in possession of the polynomials $a_i(Z), b_i(Z)$ for all $i \in [n]$. At request of a signature for message $M$ at time $t = t_1 \ldots t_\ell$ it uses these information to construct valid signature shares for $M$ on behalf of all uncorrupted parties: for $j = k+1, \ldots, n$ it computes $\sigma'_{1,j}$ as

$$\prod_{i \in QUAL\setminus\{n\}} \prod_{s=0}^{k} (H_{is})^{j^s} \cdot \big(\hat{H}_{n0}\big) \cdot \prod_{s=1}^{k} \Big((\hat{H}_{n0})^{\lambda_{s0}} \cdot \prod_{i=1}^{k} ([h]_1^{s^*_{ni}})^{\lambda_{si}}\Big)^{j^s}.$$

   Then, it picks $r_j \leftarrow \mathbb{Z}_q$ uniformly at random and computes $\sigma_{1,j}$ as

$$\sigma'_{1,j} \cdot ([(h_0 + \sum_{v=1}^{\ell} h_v t_v + h_{\ell+1} M) r_j]_1.$$

   Afterwards, it computes $\sigma_{2,j}$ as:

$$\sigma_2^{\sum_{s=1}^{k} \lambda_{s0} \cdot j^s + 1} \cdot [r_j]_2.$$

2. The simulator sends $(\sigma_{1,i}, \sigma_{2,i})$ for all $i = k+1, \ldots, n$ to the corrupted parties $P_1, \ldots, P_k$. The simulator might receive partial signatures on behalf of the corrupted parties.

3. The simulator does nothing.

4. The adversary can use any set $\mathcal{V}$ of at least $k+1$ partial signatures to construct the final signature:

$$(\sigma_1, \sigma_2) = (\prod_{i \in \mathcal{V}} \sigma_{1,i}^{L_i}, \prod_{i \in \mathcal{V}} \sigma_{2,i}^{L_i}),$$

   where $L_i$ are Lagrange coefficients. The simulator does nothing.

Figure 4.6: The simulation of the signing protocol from Figure 4.4.

**Simulation of Sign.** Note that during the simulation of the signing protocol the simulator already simulated the distributed key generation protocol and is therefore in possession of the secret shares $x_1, \ldots, x_k$ and the polynomials $a_i(Z), b_i(Z)$ for all $i \in [n]$.
It follows from the **DKG** protocol that the secret shares for all parties $P_j, j \in [n]$ are defined as $x_j := \sum_{i \in QUAL} s_{ij} \bmod q$. For **DKGSim** these are defined as $x_j := \sum_{i \in QUAL\setminus\{n\}} s_{ij} + s^*_{nj} \bmod q$, where the values $s^*_{nj}$ for $j = k+1, \ldots, n$, i.e. for $j \in \mathcal{G}$, are not explicitly known. From **DKGSim** we also derive that the corresponding public

keys equal:

$$[x_j]_2 = \prod_{i \in QUAL \setminus \{n\}} [s_{ij}]_2 \cdot [s^*_{nj}]_2 = \prod_{i \in QUAL \setminus \{n\}} \prod_{s=0}^{k} (A_{is})^{j^s} \prod_{s=0}^{k} (A^*_{ns})^{j^s},$$

where the values $A^*_{ns}$ include the common public key $y = [x]_2$. Hence, in order to compute the secret share $[hx_j]_1$ for $j \in \mathcal{G}$ either the corresponding value $[hx]_1$ or $s^*_{nj}$ seem to be required. Although these values are unknown to the simulator it is possible to use a valid signature $(\sigma_1, \sigma_2)$ for message $M$ to extract a valid signatures for all parties $P_j, j \in \mathcal{G}$. For this purpose define:

- $H_{is} := [ha_{is}]_1 \in \mathbb{G}_1$ for all $i \in QUAL \setminus \{n\}, s = 0, \ldots, k$, where $a_{is}$ are the coefficients of the polynomials computed during **DKGSim**
- $H^*_{n0} := \prod_{i \in QUAL \setminus \{n\}} H^{-1}_{i0} [hx]_1 \in \mathbb{G}_1$.
- $s^*_{nj} := s_{nj} = a_n(j)$ for $j = 1, \ldots, k$
- $H^*_{ns} := (H^*_{n0})^{\lambda_{s0}} \cdot \prod_{i=1}^{k} [hs^*_{ni}]_1^{\lambda_{si}} \in \mathbb{G}_1$ for $k = 1, \ldots, t$, where $\lambda_{is}$s are the Lagrange interpolation coefficients
- $\hat{H}_{n0} := \prod_{i \in QUAL \setminus \{n\}} (H^{-1}_{i0}) \sigma_1 \in \mathbb{G}_1$

Analogously to $[x_j]_2$ the value $[hx_j]_1$ is defined as

$$\prod_{i \in QUAL \setminus \{n\}} \prod_{s=0}^{k} (H_{is})^{j^s} \cdot \prod_{s=0}^{k} (H^*_{ns})^{j^s}.$$

To obtain a valid signature under public key $pk_j$ compute an intermediate $\sigma'_{1,j}$ as:

$$\begin{aligned}
&\prod_{i \in QUAL \setminus \{n\}} \prod_{s=0}^{k} (H_{is})^{j^s} \hat{H}_{n0} \prod_{s=1}^{k} \left( (\hat{H}_{n0})^{\lambda_{s0}} \prod_{i=1}^{k} [hs^*_{ni}]_1^{\lambda_{si}} \right)^{j^s} \\
&= \prod_{i \in QUAL \setminus \{n\}} \prod_{s=0}^{k} (H_{is})^{j^s} \prod_{i \in QUAL \setminus \{n\}} H^{-1}_{i0} \sigma_1 \prod_{s=1}^{k} \left( ( \prod_{i \in QUAL \setminus \{n\}} H^{-1}_{i0} \sigma_1)^{\lambda_{s0}} \prod_{i=1}^{k} [hs^*_{ni}]_1^{\lambda_{si}} \right)^{j^s} \\
&= \prod_{i \in QUAL \setminus \{n\}} \prod_{s=0}^{k} (H_{is})^{j^s} \left( \prod_{i \in QUAL \setminus \{n\}} H^{-1}_{i0} [hx + (h_0 + \sum_{v=1}^{\ell} h_v t_v + h_{\ell+1} M) r]_1 \right) \\
&\cdot \prod_{s=1}^{k} \left( ( \prod_{i \in QUAL \setminus \{n\}} (H^{-1}_{i0}) [hx + (h_0 + \sum_{v=1}^{\ell} h_v t_v + h_{\ell+1} M) r]_1)^{\lambda_{s0}} \prod_{i=1}^{k} [hs^*_{ni}]^{\lambda_{si}} \right)^{j^s} \\
&= \prod_{i \in QUAL \setminus \{n\}} \prod_{s=0}^{k} (H_{is})^{j^s} \left( \prod_{i \in QUAL \setminus \{n\}} (H^{-1}_{i0}) [hx]_1 \right) [(h_0 + \sum_{v=1}^{\ell} h_v t_v + h_{\ell+1} M) r]_1 \\
&\cdot \prod_{s=1}^{k} \left( ( \prod_{i \in QUAL \setminus \{n\}} (H^{-1}_{i0}) [hx]_1)^{\lambda_{s0}} \prod_{i=1}^{k} [hs^*_{ni}]_1^{\lambda_{si}} \right)^{j^s} \prod_{s=1}^{k} \left( [(h_0 + \sum_{v=1}^{\ell} h_v t_v + h_{\ell+1} M) r \lambda_{s0}]_1 \right)^{j^s}
\end{aligned}$$

$$= \prod_{i \in QUAL \setminus \{n\}} \prod_{s=0}^{k} (H_{is})^{j^s} H^*_{n0} \prod_{s=1}^{k} \left(H^*_{ns}\right)^{j^s} [(h_0 + \sum_{j=1}^{\ell} h_j t_j + h_{\ell+1} M)(r \sum_{s=1}^{k} \lambda_{s0} j^s + r)]_1$$

$$= \prod_{i \in QUAL \setminus \{n\}} \prod_{s=0}^{k} (H_{is})^{j^s} ( \prod_{s=0}^{k} H^*_{ns})^{j^s} [(h_0 + \sum_{v=1}^{\ell} h_v t_v + h_{\ell+1} M)(r \sum_{s=1}^{k} \lambda_{s0} j^s + r)]_1$$

$$= [hx_j]_1 [(h_0 + \sum_{v=1}^{\ell} h_v t_v + h_{\ell+1} M)(r \sum_{s=1}^{k} \lambda_{s0} j^s + r)]_1.$$

Then, pick a uniformly random $r_j \leftarrow \mathbb{Z}_p$ and re-randomize $\sigma_{1,j}$ and $\sigma_{2,j}$ as:

$$\sigma'_{1,j} [(h_0 + \sum_{v=1}^{\ell} h_v t_v + h_{\ell+1} M) r_j]_1 \quad \text{and} \quad \sigma_2^{\sum_{s=1}^{k} \lambda_{s0} \cdot j^s + 1} [r_j]_2. \tag{4.10}$$

Note that without re-randomization the computations would all be deterministic. This would allow an adversary two re-compute the original signature from two different signature shares and thus to distinguish the simulated game from the original security game.

**Lemma 4.1.** *The protocols **DKG** and **DKGSim** are indistinguishable.*

*Proof.* Compared to the original protocol, our modifications to **DKG** and **DKGSim** have no impact on the adversarie's view. That is, the broadcasted messages are as in the original protocol instantiated with $\mathbb{G}_2$, the view of the corrupted parties is also the same, and the storing of the secret key shares by the uncorrupted parties is done internally and cannot be seen. Thus, for static adversaries the proof is the same as in the original version and can be found in Theorem 2 of [GJKR07]. In order to manage adaptive adversaries, the simulator has to guess prior to step 1 of **DKGSim** which parties the adversary is going to corrupt. Whenever the adversary corrupts a party $P_j$, $j \in \mathcal{B}$ the simulator delivers all values computed and stored on behalf of this party. The adversary takes over the role of $P_j$. The simulator aborts if its guess was wrong. The simulation is successful with probability at least $1/\binom{n}{k}$. □

**Lemma 4.2.** *The protocols **Sign** and **SignSim** are indistinguishable.*

*Proof.* We compare the information seen by the adversary in each step of both protocols.

1. In both protocols the adversary sees or sends the signing request for message $M$ at time $t$.

2. The adversary receives signature shares, which are all valid and uniformly random. The adversary is allowed to send valid or invalid signature shares on behalf of the corrupted parties or to halt these parties. This has no impact as their are sufficient, i.e. $k+1$, uncorrupted parties left to sign the message.

3. The adversary can check the validity of the received signature shares. All the shares sent by the simulator are valid and uniformly random.

4. For every set of at least $k+1$ valid partial signatures the adversary can construct a valid signature for the public key $y$. Since all partial signatures are valid and uniformly random all possible final signatures are valid and uniformly random, as well.

We conclude that in each step both protocols have the same probability distribution and that the view of the adversary is identical. Thus, both of them are indistinguishable. □

**Theorem 4.3.** *The scheme $(T, n, k)$-$\Sigma_{FST}$ is $(n, k_1, k_2)$-robust, if $k_1 + k_2 \leq k$ and $n \geq 2k + 1$. In particular, the scheme is $(n, 0, k)$-robust, i.e. robust against malicious adversaries.*

*Proof.* Here, we argue for the strongest case, i.e. $(n, 0, k)$. The only protocols where the adversary on behalf of the compromised parties may interact with honest ones are **KeyGen** and **Sign**. Hence, we have to show that the adversary is not able to prevent the honest parties from executing these protocols successfuly. For **KeyGen**, instantiated with the **DKG** protocol, it follows from the proof in [GJKR07]. Basically, the reason is that any party who deviates from the protocol is either disqualified or its correct secret share is reconstructed by the honest majority. For **Sign**, we make use of the fact that the **DKG** protocol delivers for all parties $P_i$ the elements $[x_i]_2$, where $x_i$ is their share of the secret. Functioning as public keys $pk_i$ they allow the other parties to check validity in (4.7) of the submitted partial signatures. Hence, only signatures which are valid for message $M$ will be aggregated. It follows that $k$ parties can neither manipulate nor prevent the key generation or the final signature as long as $n \geq 2k + 1$. □

**Theorem 4.4.** *Let $\mathcal{A}$ be an adaptive adversary that $(t_{\mathcal{A}}, \varepsilon_{\mathcal{A}})$-breaks the EUF-CMA security of $(T, n, k)$-$\Sigma_{FST}$ and let $\Sigma$ be the single user signature scheme from [DN19]. Given $\mathcal{A}$, we can build an adversary $\mathcal{R}$ that $(t_{\mathcal{R}}, \varepsilon_{\mathcal{R}})$-breaks the EUF-CMA forward security of $\Sigma$ for $T$ time periods such that*

$$t_{\mathcal{R}} = O(t_{\mathcal{A}}) \text{ and } \varepsilon_{\mathcal{R}} \geq \binom{n}{k}^{-1} \varepsilon_{\mathcal{A}}.$$

*Proof.* We show how $\mathcal{R}$ simulates $\mathcal{A}$'s EUF-CMA security experiment instantiated with $\Sigma_{FST}$ perfectly and how it uses the challenge provided by $\mathcal{A}$ to win its own EUF-CMA security experiment instantiated with $\Sigma$. Simulator $\mathcal{R}$ receives a public key $y \in \mathbb{G}_2$ at the beginning of it s EUF-CMA forward security experiment. Then, $\mathcal{R}$ guesses the first $k$ parties $\mathcal{A}$ might compromise. Let $\mathcal{B}'$ denote the set of indices of these parties. W.l.o.g. we assume these parties to be $P_1, \dots, P_k$. Further let $\mathcal{G} = \{1, \dots, n\}$ and $\mathcal{B} = \emptyset$ denote the indices of the uncompromised and compromised parties from the perspective of $\mathcal{A}$. Then, $\mathcal{R}$ and $\mathcal{A}$ continue with the key generation procedure **KeyGen**. During this procedure, the adversary is allowed to make $k$ queries to **Break-In**. The procedures for adversary $\mathcal{A}$ are simulated by $\mathcal{R}$ as follows.

- **KeyGen.** To simulate the **KeyGen** protocol for $\mathcal{A}$ the simulator runs the **DKGSim** protocol on input $(y, n, k)$. $\mathcal{A}$ receives all information to compute the initial secret key shares of the compromised paroties as well as the common public key $pk = y$ and the user public keys $pk_i$ for $i = 1, \dots, n$. Additionally to all public keys, $\mathcal{R}$ holds the initial secret keys $sk_{0,i}$ for all $i \in \mathcal{B}'$.

- **KeyUpdate.** Whenever $\mathcal{A}$ asks to execute **KeyUpdate**, then $\mathcal{R}$ asks for **KeyUpdate** in the EUF-CMA forward security game of $\Sigma$.
- **Signing**$(t, M)$**.** Whenever $\mathcal{A}$ sends a signing query for message $M$ at time period $t$, $\mathcal{B}$ forwards this query to its own signing oracle and receives a signature $(t, \sigma_1, \sigma_2)$. Then, $\mathcal{B}$ runs the **SignSim** protocol with input $(t, M, (\sigma_1, \sigma_2), y)$.
- **Break-In**$(j, t)$**.** Adversary $\mathcal{A}$ queries the break-in oracle on input an index $j$ and the current time period $t$. Simulator $\mathcal{R}$ checks if $|\mathcal{B}| < k$. If this holds and $j \notin \mathcal{B}'$, then $\mathcal{R}$ aborts. Else $\mathcal{R}$ removes $j$ out of $\mathcal{G}$ and adds it to $\mathcal{B}$. If $sk_{j,t}$ is already defined, it is delivered to $\mathcal{A}$. If $|\mathcal{B}| = k$ and $j \in \mathcal{G}$, the challenger $\mathcal{R}$ sets $\tilde{t} \leftarrow t$ and computes the secret keys $sk_{t,j}$ for all $j \in \mathcal{G}$ as follows. First, it queries the break-in oracle in the EUF-CMA security experiment of $\Sigma$. It obtains the secret key $sk_t$, which consists of tuples of the form

$$(c, d, e_{s+1}, \dots, e_{\ell+1}) = \left( [r]_2, [hx + (h_0 + \sum_{v=1}^{s} h_v {w_v}^r)]_1, [h_{s+1} r]_1, \dots, [h_{\ell+1} r]_1 \right),$$

which correspond to either internal nodes $\omega = \omega_1 \dots \omega_s$, where $s \leq \ell$ or to the leaf representing time period $t$. It defines similar to **SignSim**:

  - $H_{is} := [ha_{is}]_1$ for all $i \in QUAL \backslash \{n\}$, $s = 0, \dots, k$, where $a_{is}$ are the coefficients of the polynomials computed during **DKGSim**.
  - $H^*_{n0} := \prod_{i \in QUAL \backslash \{n\}} (H_{i0}^{-1}) [hx]_1$.
  - $s^*_{nj} := s_{nj} = a_n(j)$ for $j = 1, \dots, k$.
  - $H^*_{ns} := (H^*_{n0})^{\lambda_{s0}} \cdot \prod_{i=1}^{k} [hs^*_{ni}]_1^{\lambda_{si}}$ for $s = 1, \dots, k$, where $\lambda_{si}$s are the Lagrange interpolation coefficients.
  - $\bar{H}_{n0} := \prod_{i \in QUAL \backslash \{n\}} (H_{i0}^{-1} \cdot d)$ for each tuple $(c, d, \dots)$ separately.

  Then, for all tuples on the stack it computes $d_j$ as:

$$\prod_{i \in QUAL \backslash \{n\}} \prod_{s=0}^{k} (H_{is})^{j^s} \cdot (\bar{H}_{n0}) \cdot \prod_{s=1}^{k} \left( (\bar{H}_{n0})^{\lambda_{s0}} \cdot \prod_{i=1}^{k} [hs^*_{ni}]_1^{\lambda_{si}} \right)^{j^s}.$$

  for all values $x$ from $\{c, e_{i+1}, \dots, e_{\ell+1}\}$ it computes $x_j$ as:

$$x^{\sum_{s=1}^{k} \lambda_{s0} \cdot j^s + 1}$$

  In order to guarantee a perfect simulation $\mathcal{R}$ re-randomizes the secret keys of all parties in the same fashion as the signatures in (4.10). Finally, it outputs $sk_{t,j}$ for all $j \in \mathcal{G}$ as the stack of tuples of the form $(c_j, d_j, e_{j,i+1}, \dots, e_{j,\ell+1})$.

  Analogously to the signature in **SignSim** it holds that

$$d_j = \prod_{i \in QUAL \backslash \{n\}} \prod_{k=0}^{k} (H_{is})^{j^s} \cdot (\bar{H}_{n0}) \cdot \prod_{s=1}^{k} \left( (\bar{H}_{n0})^{\lambda_{s0}} \cdot \prod_{i=1}^{k} [hs^*_{ni}]_1^{\lambda_{si}} \right)^{j^s}$$

$$= [hx_j + (h_0 + \prod_{v=1}^{s} h_v w_v)(r \sum_{s=1}^{k} \lambda_{k0} j^s + r)]_1.$$

Overall, these stack form valid secret keys $sk_{t,j}$ for all $j \in \mathcal{G}$ and their simulation is perfect.

- **Finalize**$(M^*, (t^*, \sigma_1^*, \sigma_2^*))$. At the end, $\mathcal{A}$ submits the challenge $(M^*, (t^*, \sigma_1^*, \sigma_2^*))$. $\mathcal{B}$ forwards this challenge to the **Finalize** procedure in its own EUF-CMA security experiment.

Together with Lemma 4.1 and Lemma 4.2 it follows that the simulation of the security experiment for $\mathcal{A}$ is perfect. Both, $\mathcal{A}$ as well as $\mathcal{R}$ have to provide a valid forgery for the same public key and under the same restriction of the time period $\tilde{t}$ from the Break-In procedure. Hence, $\mathcal{R}$ wins its security game whenever $\mathcal{A}$ provides a valid forgery and $\mathcal{R}$ guesses the corrupted parties correctly, which happens with probability at least $\binom{n}{k}^{-1}$. Note that a forgery $(M^*, (t^*, \sigma_1^*, \sigma_2^*))$ is only valid if no signature for $(t^*, M^*)$ was queried. The running time of $\mathcal{R}$ is the running time of $\mathcal{A}$ plus some small overhead. Overall, we have

$$t_{\mathcal{R}} = O(t_{\mathcal{A}}) \text{ and } \varepsilon_{\mathcal{R}} \geq \binom{n}{k}^{-1} \cdot \varepsilon_{\mathcal{A}}.$$

□

## 4.6 Forward-Secure Threshold PKE Schemes

In this section, we introduce our FST encryption scheme. As in the signature scheme, we will make use of the **DKG** protocol presented in Section 4.2.1. We start with formal definitions of FST-PKE schemes and their security. We adapt the definition of FST-PKE and its security from Libert and Yung [LY13b].

**Definition 4.13. Forward-secure threshold public key encryption scheme (FST-PKE)** A forward-secure threshold public key encryption scheme $(T, n, k)$-$\Pi_{FST}$ for $T$ time periods, $n$ parties, and threshold $k$ is defined via the following components:

- **KeyGen**$(1^\lambda, n, k, T) \to (pk, (pk_i)_{i \in [n]}, (sk_{0,i})_{i \in [n]})$. On input the security parameter in unary, the maximum number of time periods $T$, the maximum number of parties $n$, and a threshold $k$, it outputs a common public key $pk$, user public keys $(pk_i)_{i \in [n]}$, and initial secret key shares , $(sk_{0,i})_{i \in [n]}$.

- **KeyUpdate**$(sk_{t,i}) \to sk_{t+1,i}$. On input a secret key share for a time period $t < T - 1$, it outputs a secret key share for the next time period $t + 1$ and deletes the input from its storage. Else it outputs $\perp$.

- **Enc**$(t, pk, M) \to C$. On input a time period $t$, a common public key $pk$, and a message $M$, it outputs a ciphertext $C$.

- $\mathbf{Dec}(t, C) \to M$. If run by at least $k+1$ honest and uncompromised parties on input a time period $t$ and a ciphertext $C$, it outputs a message $M$.

The key generation procedure can be either a protocol between all parties or executed by a trusted dealer. The key update procedure is assumed to be non-interactive. The decryption procedure is a protocol and contains of various steps: ciphertext-verify, share-decrypt, share-verify, and combine. For simplicity we defined the input only as a time period and a ciphertext and omit the key material held by all participating parties.

**Definition 4.14. Correctness.** Let $(pk, (pk_{0,i})_{i\in[n]}, (sk_{0,i})_{i\in[n]}) \leftarrow$**KeyGen** and $(sk_{t,i}) \leftarrow \mathbf{KeyUpdate}(sk_{j-1,i})$ for $j = 1, \dots, t$ and $i \in [n]$. We call $(T, n, k)$-$\Pi_{FST}$ correct if for all messages $M$, all time periods $t \in \{0, \dots, T-1\}$, and all subsets $\mathcal{U} \subseteq \{sk_{t,1}, \dots sk_{t,n}\}$ of size at least $k+1$ held by uncorrupted parties it holds that

$$\Pr[\mathbf{Dec}(t, \mathbf{Enc}(t, pk, M)) = M] = 1.$$

Note that the secret keys are an implicit input to **Dec**.

We adapt the robustness notion of threshold signature schemes by Gennaro et al.[GJKR07] to FST public key encryption schemes.

**Definition 4.15. Robustness**. A forward-secure threshold PKE $(T, n, k)$-$\Pi_{FST}$ is $(n, k_1, k_2)$-robust if in a group of $n$ parties, even in the presence of an adversary who halts up to $k_1$ and corrupts maliciously $k_2$ parties, **Keygen** and **Dec** complete successfully.

Note that malicious adversaries can either deviate from the protocol in any way and especially halt some parties. Hence, they are stronger than halting adversaries, see [AMN01].

**CCA forward security.** Chosen ciphertext attack (CCA) forward security against adaptive (static)[9] adversaries is defined by the following game between a challenger and an adversary $\mathcal{A}$. Let $\mathcal{B}$ and $\mathcal{G}$ be the sets of indices, which denote the corrupted and uncorrupted parties, respectively. Initially $\mathcal{B}$ is empty and $\mathcal{G} = \{1, \dots, n\}$. The challenger (on behalf of the uncorrupted parties) and the adversary (on behalf of the corrupted parties) run $\mathbf{KeyGen}(1^\lambda, n, k, T)$. The adversary receives the common public key $pk$, all user public keys $(pk_i)_{i\in[n]}$ and the initial user secret key shares $(sk_{i,0})_{i\in\mathcal{B}}$. The adversary has access to the following oracles:

- $\underline{\mathbf{Break\text{-}In}(t', j)}$. On input time period $t'$ and index $j \in \mathcal{G}$, the challenger checks if $|\mathcal{B}| < k$. If this holds, the challenger removes $j$ out of $\mathcal{G}$ and adds it to $\mathcal{B}$. If $sk_{t,j}$ is already defined, i.e. after **KeyGen** had finished, it is delivered to $\mathcal{A}$. If $|\mathcal{B}| = k$, the challenger outputs $sk_{t,j}$ for all $j \in \mathcal{G}$. [10]

- $\underline{\mathbf{Challenge}(t^*, M_0, M_1)}$. The adversary submits a time period $t^*$ and two messages $M_0, M_1$. The challenger picks a bit $b$ uniformly at random and responds with a challenge ciphertext $C^* = \mathbf{Enc}(t^*, pk, m_b)$.

[9]In the static security model the adversary has to submit its choice of $k$ parties it wants to corrupt before receiving the public key. In the adaptive model it can corrupt the parties at any time. For a proper overview see [AMN01].

[10]Note that this case can only occur for time periods $t > 0$, i.e. after **KeyGen** had finished. Otherwise the adversary would have no possibility to win the security game.

- **Dec**$(t, C)$. On input time period $t$ and ciphertext $C$, the challenger (on behalf of the uncorrupted parties) and the adversary $\mathcal{A}$ (on behalf of the corrupted parties) run the decryption protocol **Dec**$(t, C)$. The output of this execution is delivered to $\mathcal{A}$. If **Challenge**$(t^*, M_0, M_1)$ has already been queried and $C^*$ is the response to this query then query **Dec**$(t^*, C^*)$ is disallowed.

- **Guess**$(b')$. The adversary outputs its guess $b' \in \{0, 1\}$. The challenger outputs 1 if $b = b'$, else 0. The game stops.

The adversary is allowed to make $k+1$ queries **Break-In**$(t', j)$ one query **Challenge**$(t^*, m_0, m_1)$, and multiple queries **Dec**$(t, C)$, in any order, subject to $0 \leq t^* < t'_{k+1} < T$, where $t'_{k+1}$ is the time period of the $k+1$-th query to **Break-In**. After **Break-In**$(t'_{k+1}, j)$, **Dec**$(t, C)$ cannot be queried anymore. **Guess**$(b')$ can only be queried after **Challenge**$(t^*, M_0, M_1)$. For all queries the time periods must be in $[0, \ldots, T-1]$.

**Definition 4.16.** Let $\mathcal{A}$ be an adaptive (static) adversary playing the CCA forward-security game for a FST-PKE $(T, n, k)$-$\Pi_{FST}$. It $(t_{\mathcal{A}}, \varepsilon_{\mathcal{A}})$-breaks the CCA forward security of $(T, n, k)$-$\Pi_{FST}$, if it runs in time $t_{\mathcal{A}}$ and

$$|\Pr[\mathbf{Guess}(b') = 1] - 1/2| \geq \varepsilon_{\mathcal{A}}.$$

**Remark 4.3.** *The only difference between the CPA and CCA security game is that the adversary has no access to the decryption oracle in the former game.*

### 4.6.1 New Forward-Secure Threshold PKE Scheme

Here, we introduce our FST encryption scheme. We show how the modified **DKG** protocol from Fig. 4.3 allows to create a FST-PKE scheme, which is much more efficient than the state of the art. Eventually, we prove our FST-PKE scheme CCA forward secure against adaptive and robust against malicious adversaries.

**The scheme.** Again, we assume the common parameters to be given in our scheme. As explained in Section 4.5.1 this does not contradict no need of a trusted dealer. Our FST-PKE scheme $(T, n, k)$-$\Pi_{FST}$ is defined as follows.

- **Common parameters.** Let $\mathcal{H}$ be a familiy of hash functions $H : \{0,1\}^* \to \mathbb{Z}_q$ mapping bits to elements in $\mathbb{Z}_q$. Let $\Sigma = (\mathbf{Sig.KeyGen}, \mathbf{Sig.Sign}, \mathbf{Sig.Verify})$ be a signature scheme. The common parameters consist of $\mathcal{PG}$, the description of a cryptographic Type-3 pairing group, the description of hash function $H$ picked randomly from $\mathcal{H}$, the description of $\Sigma$, the maximum number of time periods $T = 2^\ell$ as well as random group elements $[1]_1, [h]_1, [h_0]_1, \ldots, [h_{\ell+1}]_1 \leftarrow \mathbb{G}_1$, and $[1]_2, [\tilde{h}]_2 \leftarrow \mathbb{G}_2$, where $h, h_0, \ldots, h_{\ell+1}, \tilde{h} \in \mathbb{Z}_q$ and where $[1]_1$ and $[1]_2$ are generators of $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively.

- **KeyGen**$(1^\lambda, n, k, T)$. The $n$ parties run the **DKG**$(n, k)$ protocol from Figure 4.3. Subsequently each party $P_i$ holds an initial secret key share

$$sk_{0,i} := ([r_i]_2, [hx_i]_1[h_0 r_i]_1, [h_1 r_i]_1, \ldots, [h_{\ell+1} r_i]_1) \in \mathbb{G}_2 \times \mathbb{G}_1^{\ell+2}$$

as well as the public keys $pk_j = [x_j]_2$ for all $j \in [n]$. The common public key $pk = [x]_2 \in \mathbb{G}_2$ is published. [11]

- **KeyUpdate**($sk_{t,i}$). We assume the $T$ time periods $0, \ldots, 2^\ell - 1$ as being organized as leaves of a binary tree of depth $\ell$ and sorted in increasing order from left to right. This means, $00\ldots0$ is the first and $11\ldots1$ is the last time period. The path from the root of the tree to a leaf node $t$ equals the bit representation $t_1 \ldots t_\ell$, where we take the left branch for $t_z = 0$ and the right one for $t_z = 1$. Prefixes of time periods correspond to internal nodes $\omega = \omega_1, \ldots, \omega_s$, where $s < \ell$. Let $r'_i \leftarrow \mathbb{Z}_q$ be picked uniformly at random. Then, we associate to each party $P_i, i \in [n]$ and each node $\omega$ a secret key:

$$(c, d, e_{s+1}, \ldots, e_{\ell+1})s = \left([r]_2, [hx + h_0 r + \sum_{v=1}^{s} h_v w_v r]_1, [h_{s+1} r]_1, \ldots, [h_{\ell+1} r]_1\right). \tag{4.11}$$

Given such a secret key, we derive a secret key for a descendant node $\omega' = \omega_1 \ldots \omega_{s'}$, where $s' > s$ as

$$\begin{aligned}
&(c', d', e'_{s+1}, \ldots, e'_{\ell+1}) \\
&= \left(c[1]_2^{r'}, d \prod_{v=s+1}^{s'} e_v^{w_v} \left([h_0]_1 \prod_{v=1}^{s'} [h_v]_1^{w_v}\right)^{r'}, e_{s'+1}[h_{s'+1}]_1^{r'}, \ldots, e_{\ell+1}[h_{\ell+1}]_1^{r'}\right) \\
&= \left([r + r']_2, [hx + h_0 r + \sum_{v=1}^{s} h_v w_v r]_1 [\sum_{v=s+1}^{s'} h_v w_v r]_1 [h_0 r' + \sum_{v=1}^{s'} h_v w_v r']_1,\right. \\
&\left.[h_{s'+1}(r + r')]_1, \ldots, [h_{\ell+1}(r + r')]_1\right) \\
&= \left([r + r']_2, [hx + h_0(r + r') + \sum_{v=1}^{s'} h_v w_v (r + r')]_1,\right. \\
&\left.[h_{s'+1}(r + r')]_1, \ldots, [h_{\ell+1}(r + r')]_1\right),
\end{aligned}$$

where $r'_i \leftarrow \mathbb{Z}_q$ is picked uniformly at random.
We define $C_t$ as the smallest subset of nodes that contains an ancestor or leaf for each time period $t, \ldots, T-1$ , but no nodes of ancestors or leafs for time periods $0, \ldots, t-1$. For time period $t$, we define the secret key $sk_{i,t}$ of party $P_i$ as the set of secret keys associated to all nodes in $C_t$. To update the secret key to time period $t+1$, determine $C_{t+1}$ and compute the secret keys for all nodes in $C_{t+1} \setminus C_t$. Afterwards, delete $sk_{i,t}$ and all used re-randomization exponents $r'_i$.

[11] Note that the secret share $x_i$ is computed commonly by all parties and the randomness $r_i$ is computed locally by party $P_i$ and is not a share of another random value. This approach is more efficient than computing random values commonly, especially to different bases.

- **Encrypt**$(pk, t, M)$. Let $M \in \mathbb{G}_3$ be the message and $t_1 \dots t_\ell$ the bit representation of time period $t$. First, run **Sig.KeyGen**$\rightarrow (vk, signk)$ and compute $H(vk) =: VK$. Then, pick a uniformly random $r \leftarrow \mathbb{Z}_q$ and compute $(C_1, C_2, C_3)$ as

$$\left( e([h]_2, pk)^r \cdot M,\ [1]_2^r,\ [(h_0 + \sum_{v=1}^{\ell} h_v t_v + h_{\ell+1} VK) r]_1 \right) \in \mathbb{G}_3 \times \mathbb{G}_2 \times \mathbb{G}_1$$

  and **Sig.Sign**$(signk, (C_1, C_2, C_3), ) \rightarrow \sigma$. Output the ciphertext

$$C = ((C_1, C_2, C_3), vk, \sigma).$$

- **Decrypt**$(t, (C_1, C_2, C_3), vk, \sigma)$. Let $\mathcal{W}$ be the set of indices of all participating parties. W.l.o.g. we assume that $\mathcal{W}$ contains at least $k + 1$ distinct indices. The participating parties run the decryption protocol from Figure 4.7.[12]

[12] Note that decryption happens with respect to a time period. Since time periods are encoded in full bit length (even if they start with zero) they are low in the binary tree. Hence, the corresponding keys have only left $e_{i,\ell+1}$ as going down one level in depth erases one value $e_{i,x}, x \in \{1, \dots, \ell\}$.

**Dec**$(t, (C_1, C_2, C_3), vk, \sigma, P_1, ..., P_n)$**:**

1. **Ciphertext-Verify.** At decryption request of $((C_1, C_2, C_3), vk, \sigma)$ at time $t = t_1 \dots t_\ell$, each party $P_i, i \in \mathcal{W}$ checks whether **Sig.Verify**$(vk, (C_1, C_2, C_3), \sigma) = 1$ and whether the ciphertext is valid for time period $t$ and for the hashed verification key $VK = H(vk)$. That is, it checks whether the following equation holds.
$$e([h_0 + \sum_{v=1}^{s} h_v t_v + h_{\ell+1} VK]_1, C_2) = e(C_3, [1]_2).$$
If one or both checks fail it aborts.
2. **Share-Decrypt.** Else each party picks a uniformly random $v_i \leftarrow \mathbb{Z}_q$ and uses its secret key $sk_{i,t} = (c_i, d_i, e_{i,\ell+1})$ to compute a decryption share $(c_i', d_i')$, where
$$d_i' := d_i e_{i,\ell+1}^{VK} [(h_0 + \sum_{v=1}^{\ell} h_v t_v + h_{\ell+1} VK) v_i]_1 \quad \text{and} \quad c_i' := c_i [v_i]_2.$$
Afterwards, each party $P_i, \in \mathcal{W}$ sends $(c_i', d_i')$ secretly to all other parties.
3. **Share-Verify.** All parties in $\mathcal{W}$ use the public keys $pk_j, j \in \mathcal{W}$ to check if the contributed decryption shares are valid. That is, if
$$e(d_j', [1]_2) = e([h]_1, pk_j) \cdot e([h_0 + \sum_{v=1}^{\ell} h_v t_v + h_{\ell+1} VK]_1, c_j').$$
4. **Combine.** Let $\mathcal{V} \subseteq \mathcal{W}$ indicate a set of parties sending valid decryption shares. If $\mathcal{V}$ contains at least $k+1$ distinct indices then the decryption key $(c', d')$ is computed as
$$c' = \prod_{i \in \mathcal{V}} c_i'^{L_i} \quad \text{and} \quad d' = \prod_{i \in \mathcal{V}} d_i'^{L_i},$$
where $L_i = \prod_{j \in \mathcal{V}, j \neq i} (-i)/(j-i)$ are the Lagrange coefficients.
5. Finally, the plaintext is computed as
$$C_1 \cdot e(C_3, c') / e(d', C_2) = M. \tag{4.12}$$

Figure 4.7: The **decryption** protocol of our FST-PKE.

**Remark 4.4.** *Note that the subset $\mathcal{V} \subseteq \mathcal{W}$ from the decryption protocol (Fig. 4.7) might be of size greater than $k+1$, while $k+1$ partial decryption shares are sufficient to* *decrypt the ciphertext. For this reason aggregating only $k+1$ decryption shares avoids computational overhead.*

**Remark 4.5.** *The secret keys in our FST-PKE have a binary tree structure in the sense of [CHK03b], except for the lowest level. In Theorem 4 from [CHK03b], it is shown that encryption schemes, which have a binary tree structure on all levels, imply forward security. Although it is possible to remove the lowest level and the strong one-time signature scheme in our construction, doing so would result in a scheme which is only forward secure against CPA instead of CCA.*

### 4.6.2 Proof of Correctness.

We have to prove that $(c', d')$ is a valid decryption key for ciphertext $C = (C_1, C_2, C_3, vk, \sigma)$ under $t.VK$ and the common public key $pk = [x]_2$. To that end we show first that set $\mathcal{V}$, which indicates the correct decryption shares, can be determined. That is, we show that we can check whether the decryption shares $(c'_i, d'_i)$, $i \in \mathcal{W}$ are correct. Let $(c'_i, d'_i)$ be an honestly generated decryption in Step 2 of the protocol. Then,

$$\begin{aligned} d'_i &= d_i \cdot e_{i,\ell+1}^{VK}[(h_0 + \sum_{v=1}^{\ell} h_v t_v + h_{\ell+1} VK) r'_i]_1 \\ &= [h^{x_i} + (h_0 + \sum_{v=1}^{\ell} h_v t_v) r_i]_1 [h_{\ell+1} r_i VK]_1 [(h_0 + \sum_{v=1}^{\ell} h_v t_v + h_{\ell+1} VK) r'_i]_1 \\ &= [h x_i + (h_0 \prod_{v=1}^{\ell} h_v t_v + h_{\ell+1} VK)(r_i + r'_i)]_1, \end{aligned} \tag{4.13}$$

where we used the fact that $e_{i,\ell+1} = [h_{\ell+1} r_i]_1$. Furthermore,

$$c'_i = [r_i + r'_i]_2. \tag{4.14}$$

If a decryption share $(c'_i, d'_i)$ satisfies (4.13) and (4.14) then the validity check in Step 3 is correct, because

$$\begin{aligned} e(d'_i, [1]_2) &= e([h x_i + (h_0 + \sum_{v=1}^{\ell} h_v t_v + h_{\ell+1} VK)(r_i + r'_i)]_1, [1]_2) \\ &= e([h x_i]_1, [1]_2) \cdot e([(h_0 + \sum_{v=1}^{\ell} h_v t_v + h_{\ell+1} VK)(r_i + r'_i)]_1, [1]_2) \\ &= e([h]_1, [x_i]_2) \cdot e([(h_0 + \sum_{v=1}^{\ell} h_v t_v + h_{\ell+1} VK)]_1, [r_i + r'_i]_2) \\ &= e([h]_1, pk_i) \cdot e([h_0 + \sum_{v=1}^{\ell} h_v t_v + h_{\ell+1} VK]_1, c'_i). \end{aligned}$$

For this reason, we can indeed check whether a decryption share $(c'_i, d'_i)$ for message $C$ under $t, VK$ and user public key $[x_i]_2$ is correct and thus include $i$ into set $\mathcal{V}$. It remains to show that all decryption shares $(c'_i, d'_i)$, $i \in \mathcal{V}$ interpolate to a valid decryption key under the common public key $g_2^x$. For this purpose, we set $R := \sum_{i \in \mathcal{V}} L_i(r_i + r'_i)$. Then,

$$\begin{aligned} d' &= \prod_{i \in \mathcal{V}} d_i'^{L_i} = [h \sum_{i \in \mathcal{V}} L_i \cdot x_i]_1 [(h_0 + \sum_{v=1}^{\ell} h_v t_v + h_{\ell+1} VK)(\sum_{i \in \mathcal{V}} L_i(r_i + r'_i))]_1 \\ &= [hx]_1 [(h_0 + \sum_{v=1}^{\ell} h_v t_v + h_{\ell+1} VK) R]_1, \end{aligned}$$

where $\sum_{i \in \mathcal{V}} L_i \cdot x_i = x$ and $\sum_{i \in \mathcal{V}} L_i \cdot (r_i + r'_i) = R$. Furthermore,

$$c' = \prod_{i \in \mathcal{V}} c_i'^{L_i} = [\sum_{i \in \mathcal{V}} L_i(r_i + r'_i)]_2 = [R]_2.$$

Overall, we have for a valid ciphertext

$$\begin{aligned} & C_1 \cdot e(C_3, c')/e(d', C_2) \\ &= M \cdot e([h]_1, pk)^r \, \frac{e([(h_0 + \prod_{v=1}^{\ell} h_v t_v + h_{\ell+1} VK) r], [R])}{e([h^x + (h_0 + \sum_{v=1}^{\ell} h_v t_v + h_{\ell+1} VK) R]_1, [r]_2)} \\ &= M \cdot e([hx]_1, [r]_2) \, \frac{e([(h_0 + \prod_{v=1}^{\ell} h_v t_v + h_{\ell+1} VK) r]_1, [R]_2)}{e([hx + (h_0 \sum_{v=1}^{\ell} h_v t_v + h_{\ell+1} VK) R]_1, [r]_2)} \\ &= M. \end{aligned}$$

### 4.6.3 Proof of Security.

For our FST signature scheme, we showed that it is secure if the forward-secure single user signature scheme of Drijvers and Neven is secure. It is also possible to construct a forward-secure single user encryption scheme in the same fashion as our FST scheme and it is also possible to show that this single user scheme is secure. However, in order to show that our FST encryption scheme is secure we cannot construct a reduction from the single user to the threshold scheme. For the signature scheme this is possible because on query of a signature the reduction forwards this query to its own signing oracle. Then, it uses the received signature to compute the corresponding signature shares. In the encryption scheme however, the reduction would simply receive the plaintext from which it cannot compute valid decryption shares it needs to output to the attacker of the FST encryption scheme. We circumvent this issue by a reduction from the HIBE scheme from Boneh et al. [BBG05]; see Section 4.4.1. We interpret the identities as time periods. Then, on query of a decryption the reduction makes a user secret key query for the "identity" which corresponds to the appropriate time period of the ciphertext. Given this user secret key it can provide valid decryption shares.
Similar to the proof of our FST signature scheme, we need to simulate the key generation protocol **DKG** and the decryption protocol **Decrypt**. We start with the simulation of **Decrypt** in Figure 4.5. For the simulation of **DKG** we refer to Section 4.5.3. According to Definition 4.16, the adversary is allowed to control up to $k$ parties during the key generation and decryption procedure. First, we assume a static adversary as in the

proof in [GJKR07]. In the proof of Theorem 4.7, it is shown how to achieve security against adaptive adversaries. In Section 4.7, it is explained why this approach gives a more efficient scheme than the use of composite order groups as in [LY13b].
W.l.o.g. we assume the corrupted parties to be $P_1, \dots, P_k$. Let $\mathcal{B} := \{1, \dots, k\}$ indicate the set of corrupted parties, controlled by the adversary $\mathcal{A}$, and let $\mathcal{G} := \{k+1, \dots, n\}$ indicate the set of uncorrupted parties, run by the simulator.

**DecSim**$(t, (C_1, C_2, C_3), vk, \sigma), (c, d))$

1. The decryption is requested. Let $t = t_1 \dots t_\ell$. The simulator performs the validity checks from Step 1 of protocol **Dec**. If one of these checks fail it aborts.
2. Else, the simulator uses the polynomials $a_i(Z), b_i(Z)$ for all $i \in [n]$ and the decryption key $(c, d)$ for $t, VK$ to compute valid decryption shares on behalf of all uncorrupted parties: for $j = k+1, \dots, n$ it computes $d''_j$ as

$$\prod_{i \in QUAL \setminus \{n\}} \prod_{s=0}^{k} (H_{is})^{j^s} \hat{H}_{n0} \prod_{s=1}^{k} \left((\hat{H}_{n0})^{\lambda_{s0}} \cdot \prod_{i=1}^{k} [hs^*_{ni}]_1^{\lambda_{si}}\right)^{j^s}.$$

   Then, it picks $w_j \leftarrow \mathbb{Z}_q$ uniformly at random and computes $d'_j$ as

$$d''_j[(h_0 + \sum_{v=1}^{\ell} h_v t_v + h_{\ell+1} VK) w_j]_1.$$

   Afterwards, it computes $c'_j$ as:

$$c^{\sum_{s=1}^{k} \lambda_{s0} \cdot j^s + 1} [w_j]_2.$$

   Then, the simulator sends $(c'_j, d'_j)$ for all $j = k+1, \dots, n$ to the corrupted parties $P_1, \dots, P_k$. The simulator might receive decryption shares on behalf of the corrupted parties.
3. The simulator does nothing.
4. The adversary can use any set $\mathcal{V}$ of at least $k+1$ partial decryption shares to construct the final decryption key:

$$(c', d') = (\prod_{i \in \mathcal{V}} c_i^{L_i}, \prod_{i \in \mathcal{V}} d_i^{L_i}),$$

   where $L_i = \prod_{j \in \mathcal{V}, j \neq i} (-i)/(j-i)$ are Lagrange coefficients. The simulator does nothing.
5. The adversary can use the decryption key $(c, d)$ to decrypt the ciphertext. The simulator does nothing.

Figure 4.8: The simulation of the **decryption** protocol.

Note that during the simulation of the decryption procedure the simulator already executed **DKGSim** (Fig. 4.5). Therefore, it is in possession of the secret shares $x_1, \dots, x_k$ and the polynomials $a_i(Z), b_i(Z)$ for all $i \in [n]$. In the **DKG** protocol (Fig. 4.3) the secret shares for all parties $P_j, j \in [n]$ are defined as $x_j := \sum_{i \in QUAL} s_{ij} \bmod q$. In **DKGSim** however, the shares are defines as $x_j := \sum_{i \in QUAL \setminus \{n\}} s_{ij} + s^*_{nj} \bmod q$, where the values $s^*_{nj}$ for $j = k+1, \ldots, n$, i.e. for $j \in \mathcal{G}$, are not explicitly known. Moreover, in **DKGSim** the public key of user $P_j, j \in [n]$ is

$$[x_j]_2 = \prod_{i \in QUAL \setminus \{n\}} g_2^{s_{ij}} g_2^{s^*_{nj}} = \prod_{i \in QUAL \setminus \{n\}} \prod_{s=0}^{k} (A_{is})^{j^s} \prod_{s=0}^{k} (A^*_{ns})^{j^s},$$

where the values $A^*_{ns}$ include the common public key $y = [x]_2$. Hence, in order to compute the secret share $h^{x_j}$ for $j \in \mathcal{G}$ either the corresponding value $[hx]_1$ or $s^*_{nj}$ is required. Although these values are not known to the simulator it is still able to simulate the role of the uncompromised parties during the decryption of a valid ciphertext $(t, C_1, C_2, C_3, vk, \sigma)$. In order to do so it requires a valid secret key $(c, d)$ for time $t$ together with the hashed verification key $VK = H(vk)$, i.e. for the string $t, VK$. If the simulator is an adversary breaking the CPA security of the HIBE scheme from Section 4.4.1 this key can be requested in its own security experiment. Let $t = t_1 \ldots t_\ell$ and $(c, d) = \Big([r]_2, [hx + (h_0 + \sum_{v=1}^{\ell} h_v t_v + h_{\ell+1} VK) r]_1\Big)$. Define consistently with **DKGSim**:

- $H_{is} := [ha_{is}]_1$ for all $i \in QUAL \setminus \{n\}, s = 0, \ldots, k$
- $H^*_{n0} := \prod_{i \in QUAL \setminus \{n\}} (H_{i0}^{-1}) [hx]_1$
- $s^*_{nj} := s_{nj} = a_n(j)$ for $j = 1, \ldots, k$
- $H^*_{ns} := (H^*_{n0})^{\lambda_{s0}} \prod_{i=1}^{k} [hs^*_{ni}]_1^{\lambda_{si}}$ for $s = 1, \ldots, k$
- $\hat{H}_{n0} := \prod_{i \in QUAL \setminus \{n\}} (H_{i0}^{-1} d)$

Thus, $pk_j = [hx_j]_1, j \in \mathcal{G}$ are defined as

$$\prod_{i \in QUAL \setminus \{n\}} \prod_{s=0}^{k} (H_{is})^{j^s} \prod_{s=0}^{k} (H^*_{ns})^{j^s}.$$

To obtain a valid decryption share for party $P_j, j \in \mathcal{G}$ compute an intermediate $d''_j$ as:

$$\prod_{i \in QUAL \setminus \{n\}} \prod_{s=0}^{k} (H_{is})^{j^s} \hat{H}_{n0} \prod_{s=1}^{k} \left( (\hat{H}_{n0})^{\lambda_{s0}} \prod_{i=1}^{k} ([hs^*_{ni}]_1)^{\lambda_{si}} \right)^{j^s}$$
$$= \prod_{i \in QUAL \setminus \{n\}} \prod_{s=0}^{k} (H_{is})^{j^s} \left( \prod_{i \in QUAL \setminus \{n\}} H_{i0}^{-1} d \right) \prod_{s=1}^{k} \left( ( \prod_{i \in QUAL \setminus \{n\}} H_{i0}^{-1} \sigma_1)^{\lambda_{s0}} \prod_{i=1}^{k} [hs^*_{ni}]_1^{\lambda_{si}} \right)^{j^s}$$
$$= \prod_{i \in QUAL \setminus \{n\}} \prod_{s=0}^{k} (H_{is})^{j^s} \left( \prod_{i \in QUAL \setminus \{n\}} H_{i0}^{-1} [hx + (h_0 + \sum_{v=1}^{\ell} h_v t_v + h_{\ell+1} VK) r]_1 \right)$$
$$\prod_{s=1}^{k} \left( ( \prod_{i \in QUAL \setminus \{n\}} H_{i0}^{-1} [hx + (h_0 + \sum_{v=1}^{\ell} h_v t_v + h_{\ell+1} VK) r]_1)^{\lambda_{s0}} \prod_{i=1}^{k} [hs^*_{ni}]_1^{\lambda_{si}} \right)^{j^s}$$

$$= \prod_{i \in QUAL \setminus \{n\}} \prod_{s=0}^{k} (H_{is})^{j^s} \Big( \prod_{i \in QUAL \setminus \{n\}} H_{i0}^{-1}[hx]_1 \Big) [(h_0 + \sum_{v=1}^{\ell} h_v t_v + h_{\ell+1} VK) r]_1$$

$$\cdot \prod_{s=1}^{k} \Big( ( \prod_{i \in QUAL \setminus \{n\}} H_{i0}^{-1}[hx]_1 )^{\lambda_{s0}} \prod_{i=1}^{k} [hs_{ni}^{*}]_1^{\lambda_{si}} \Big)^{j^s} \cdot \prod_{s=1}^{k} \Big( [(h_0 + \sum_{v=1}^{\ell} h_v t_v + h_{\ell+1} VK) r \lambda_{s0}]_1 \Big)^{j^s}$$

$$= \prod_{i \in QUAL \setminus \{n\}} \prod_{s=0}^{k} (H_{is})^{j^s} H_{n0}^{*} \prod_{s=1}^{k} \left(H_{ns}^{*}\right)^{j^s} [(h_0 + \sum_{j=1}^{\ell} h_j t_j + h_{\ell+1} VK)(r \sum_{s=1}^{k} \lambda_{s0} j^s + r)]_1$$

$$= \prod_{i \in QUAL \setminus \{n\}} \prod_{s=0}^{k} (H_{is})^{j^s} ( \prod_{s=0}^{k} \left(H_{ns}^{*}\right)^{j^s} ) [(h_0 + \sum_{v=1}^{\ell} h_v t_v + h_{\ell+1} VK)(r \sum_{s=1}^{k} \lambda_{s0} j^s + r)]_1$$

$$= [hx_j + (h_0 + \sum_{v=1}^{\ell} h_v t_v + h_{\ell+1} VK)(r \sum_{s=1}^{k} \lambda_{s0} j^s + r)]_1.$$

To re-randomize, pick a uniformly random $w_j \leftarrow \mathbb{Z}_p$ and compute $d'_j$ and $c'_j$ as

$$d''_j[(h_0 + \sum_{v=1}^{\ell} h_v t_v + h_{\ell+1} VK) w_j] \quad \text{and} \quad c^{\sum_{s=1}^{k} \lambda_{s0} j^s + 1}[w_j]_2. \tag{4.15}$$

Again, without re-randomization the computations would all be deterministic. This would allow an adversary two re-compute the original user secret key from two different decryption shares and thus to distinguish the simulated game from the original security game.

**Lemma 4.3.** *The protocols **Dec** and **DecSim** are indistinguishable.*

*Proof.* We compare the information seen by the adversary in each step of both protocols:

1. In both protocols the adversary sees or sends the decryption request for message $(C_1, C_2, C_3, vk, \sigma)$ at time $t$.

2. The adversary receives decryption shares, which are all valid and uniformly random. The adversary is allowed to send valid or invalid decryption shares on behalf of the corrupted parties or to halt these parties. This has no impact as their are sufficient, i.e. $k+1$, uncorrupted parties left to provide valid decryption shares.

3. The adversary can check the validity of the received decryption shares. All the shares sent by the simulator are valid and uniformly random.

4. For every set of at least $k+1$ valid decryption shares the adversary can construct a valid decryption key for the public key $y$. Since decryption shares are valid and uniformly random all possible final decryption keys are valid and uniformly random, as well.

5. The adversary decrypts the ciphertext correctly in both protocols.

We conclude that in each step both protocols have the same probability distribution and that the view of the adversary is identical. Thus, both of them are indistinguishable. □

**Theorem 4.5.** *The scheme $(T, n, k)$-$\Pi_{FST}$ from Section 4.6 is $(n, k_1, k_2)$-robust if $k_1 + k_2 \leq k$ and $n \geq 2k + 1$. In particular, the scheme is $(n, 0, k)$-robust, i.e. robust against malicious adversaries.*

*Proof.* We argue for the strongest case, i.e. $(n, 0, k)$. To show that $(T, n, k)$-$\Pi_{FST}$ is $(n, 0, k)$-robust we analyze all protocols where the adversary on behalf of the uncompromised parties may interact with the honest ones, i.e. **KeyGen** and **Decrypt**. More precisely, we show that the adversary is incapable to prevent the honest parties from executing these protocols successfully. The **KeyGen** protocol is instantiated with the **DKG** protocol from [GJKR07], which was shown to be robust against malicious adversaries. The reason for this is that a party which deviates from the protocol specification is either disqualified or its secret share is reconstructed by the honest parties. In the case of the **Decrypt** protocol the adversary has two options to attempt cheating. One option is to try manipulating the ciphertext. This however, is prevented in Step 1 of the decryption protocol by checking the ciphertext for validity. The second option is to try manipulating or denying decryption shares. This is prevented in Step 3 by using the user public keys $[x_i]_1, i \in [n]$ to check if the decryption shares are valid. Hence only valid decryption shares are aggregated and the message is decrypted correctly. Moreover, the adversary is allowed to control or halt at most $k$ parties. Thus, a valid decryption share can still be computed as long as $n \geq 2k + 1$. □

**Theorem 4.6.** *Let $n \geq 2k + 1$ and let $\mathcal{A}$ be a static adversary that $(t_{\mathcal{A}}, \varepsilon_{\mathcal{A}})$-breaks the CCA forward security of $(T, n, k)$-$\Pi_{FST}$ from Section 4.6. Given $\mathcal{A}$, we can build an adversary $\mathcal{A}'$ that $(t_{\mathcal{A}'}, \varepsilon_{\mathcal{A}'})$-breaks the CPA security of HIBE $\Pi_{HIBE}$ from [BBG05], an adversary $\mathcal{A}''$ that $(t_{\mathcal{A}''}, \varepsilon_{\mathcal{A}''})$-breaks the sEUF-1CMA security of signature scheme $\Sigma$, and an adversary $\mathcal{A}'''$ that $(t_{\mathcal{A}''}, \varepsilon_{\mathcal{A}'''})$-breaks the collision resistance of hash function $H$, such that*

$$t_{\mathcal{A}'''} \approx t_{\mathcal{A}''} \approx t_{\mathcal{A}'} \approx t_{\mathcal{A}} \quad \textit{and} \quad \varepsilon_{\mathcal{A}'''} + \varepsilon_{\mathcal{A}''} + \varepsilon_{\mathcal{A}'} \geqslant \varepsilon_{\mathcal{A}}.$$

*Proof.* Conceptually, we follow the proofs from Sections 4 and 6 in [BCHK07], which were also reproduced in Section 4.1 in [BBH06]. In [BCHK07], a CPA-secure HIBE with $\ell + 1$ levels and identities of length $n + 1$ bits is turned into a CCA-secure HIBE with $\ell$ levels and identities of length $n$ bits. The reason for the shorter identities in the CCA-secure scheme is that this framework uses one bit of the identity as a padding. This padding guarantees that decryption queries do not correspond to prefixes of the challenge identity. In our scheme however, the first $\ell$ levels are single bits and the deepest level has elements in $\mathbb{Z}_q$, which makes it impossible to spend one bit of each identity for the padding. However, in our scheme the adversary is only allowed to make decryption queries with respect to time periods (plus a value in $\mathbb{Z}_q$). Since time periods are always encoded with full length they cannot correspond to prefixes of each other. Thus a padding is not necessary.
We start with describing an adversary $\mathcal{A}'$ playing the CPA security game for HIBE $\Pi_{HIBE}$ and simulating the CCA forward security game for a static adversary $\mathcal{A}$.
At the beginning, $\mathcal{A}$ sends its choice of the $k$ parties it wants to corrupt to $\mathcal{A}'$. Let $\mathcal{B}$ denote the set of the indices of these parties and $\mathcal{G} := \{1, \ldots, n\} \setminus \mathcal{B}$. Adversary $\mathcal{A}'$ runs **Sig.KeyGen** to obtain $(vk^*, signk^*)$. Then it computes $H(vk^*) := VK^*$. Moreover,

it receives a master public key $mpk := [x]_2 \in \mathbb{G}_2$ from its own security experiment. In order to simulate the **KeyGen** procedure adversary $\mathcal{A}'$(on behalf of the uncompromised parties) runs the **DKGSim** protocol on input $([x]_2, n, k)$. Both adversaries receive all information to compute the secret key shares of the compromised parties and the user public keys $pk_i$ for all $i \in [n]$ as well as the common public key $pk = mpk = [x]_2$. Afterwards, $\mathcal{A}$ has access to the following procedures, which are simulated by $\mathcal{A}'$ as follows.

- **Break-In**$(t', j)$.On input a time period $t' = t_1 \dots t_\ell$ adversary $\mathcal{A}'$ queries **Key-Query** on all nodes from the set $C_{t'}$, which was defined in the **KeyUpdate** procedure in 4.6. According to the definition of $C_{t'}$ these are all the nodes which allow the computation of the secret keys for all time periods $t \geqslant t'$ but for no time period $t < t'$. As a response it obtains tuples of the form

$$(c, d, e_{s+1}, \dots, e_{\ell+1}) = \left( [r]_2, [hx + (h_0 + \sum_{v=1}^{s} h_v w_v) r]_1, [h_{s+1} r]_1, \dots, [h_{\ell+1} r]_1 \right),$$

which correspond to internal nodes $\omega = \omega_1 \dots \omega_s$, where $s \le \ell$. In order to compute the secret keys $sk_{t',j}$ for all $j \in \mathcal{G}$ it proceeds as follows. It defines equivalently to **DecSim**:
  - $H_{is} := [ha_{is}]_1$ for all $i \in QUAL \setminus \{n\}, s = 0, \dots, k$
  - $H^*_{n0} := \prod_{i \in QUAL \setminus \{n\}} (H_{i0}^{-1})[hx]_1$
  - $s^*_{nj} := s_{nj} = a_n(j)$ for $j = 1, \dots, k$
  - $H^*_{ns} := (H^*_{n0})^{\lambda_{s0}} \prod_{i=1}^{k} [hs^*_{ni}]_1^{\lambda_{si}}$ for $s = 1, \dots, k$
  - $\bar{H}_{n0} := \prod_{i \in QUAL \setminus \{n\}} (H_{i0}^{-1} d)$ for each tuple $(c, d, \dots)$ separately.

  It computes for all tuples a corresponding value $d_j$ as:

$$\prod_{i \in QUAL \setminus \{n\}} \prod_{s=0}^{k} (H_{is})^{j^s} \bar{H}_{n0} \prod_{s=1}^{k} \left( (\bar{H}_{n0})^{\lambda_{s0}} \prod_{i=1}^{k} [hs^*_{ni}]_1^{\lambda_{si}} \right)^{j^s}$$

  and for all values $\tilde{x}$ from $\{c, e_{i+1}, \dots, e_{\ell+1}\}$ it computes $\tilde{x}_j$ as

$$\tilde{x}^{\sum_{s=1}^{k} \lambda_{s0} \cdot j^s + 1}.$$

  In order to guarantee a perfect simulation, $\mathcal{A}'$ re-randomizes the secret keys of all parties in the same fashion as the decryption shares in (4.15). Finally, it outputs $sk_{t',j}$ for all $j \in \mathcal{G}$ as the stack of tuples of the form $(c_j, d_j, e_{j,i+1}, \dots, e_{j,\ell+1})$. Analogously to the decryption in **DecSim** we have

$$d_j = [hx_j + (h_0 + \sum_{v=1}^{s} h_v w_v)(r' \sum_{s=1}^{k} \lambda_{k0} j^s + r')]_1.$$

  Overall, these stacks form valid secret keys $sk_{t',j}$ for all $j \in \mathcal{G}$ and their simulation is perfect. If **Challenge**$(t^*, M_0, M_1)$ was already queried then all break-in queries with $t' \leqslant t^*$ are invalid.

- **Challenge**($t^*, M_0, M_1$). Adversary $\mathcal{A}$ submits two messages $M_0, M_1$ and challenge time period $t^*$. Adversary $\mathcal{A}'$ forwards $(t^*.VK^*, M_0, M_1)$ to **Challenge** in its own security game and receives a ciphertext $(C_1, C_2, C_3)$ which equals

$$\left(e([h]_1, pk)^r \cdot M_b,\ [r]_2,\ [(h_0 + \sum_{v=1}^{\ell} h_v t_v + h_{\ell+1} VK^*)r]_1\right) \in \mathbb{G}_3 \times \mathbb{G}_2 \times \mathbb{G}_1,$$

  where $b$ is a uniformly random bit. Afterwards, it computes a signature $\sigma^* \leftarrow$ **Sig.Sign**($signk^*, (C_1, C_2, C_3)$) and outputs the challenge ciphertext

$$C^* = (C_1, C_2, C_3, vk^*, \sigma^*).$$

  If **Break-In** on input $t' \leqslant t^*$ was previously queried then the challenge query is invalid.

- **Dec**($t, C_1, C_2, C_3, vk, \sigma$). Whenever $\mathcal{A}$ asks for a decryption then $\mathcal{A}'$ proceeds as follows. First, it checks if $vk = vk^*$ and **Sig.Verify**($vk, (C_1, C_2, C_3), \sigma$) $= 1$ or if $vk \neq vk^*$ and $H(vk) = VK^*$. If one of these conditions is true then $\mathcal{A}'$ aborts and outputs a uniformly random bit to **Guess** in its own security game. Else it queries **KeyQuery**($t, VK$) to obtain the decryption key $sk_{t,VK} = (c, d)$. Then, it simulates the decryption procedure by running **DecSim**($t, (C_1, C_2, C_3), vk, \sigma, (c, d)$). Since $t, VK$ is unequal to and no prefix of $t^*, VK^*$ the query to **KeyQuery** is valid. **Dec** cannot be queried on input $(t^*, C^*)$.

- **Guess**($b'$). Adversary $\mathcal{A}$ outputs its guess $b' \in \{0, 1\}$, which $\mathcal{A}'$ forwards to **Guess** in its own experiment.

We denote **Forge** the event that $\mathcal{A}'$ aborts during a decryption query because of the first condition and **Coll** that it aborts because of the second condition. Together with Lemma 4.1 and Lemma 4.3 it can be seen that adversary $\mathcal{A}'$ provides a perfect simulation to $\mathcal{A}$ as long as any of these two events do *not* happen. Thus,

$$|\varepsilon_{\mathcal{A}} - \varepsilon_{\mathcal{A}'}| \leqslant \Pr[\textbf{Forge} \cup \textbf{Coll}] = \Pr[\textbf{Forge}] + \Pr[\textbf{Coll}]. \tag{4.16}$$

In order to determine Pr[**Forge**] note that if **Forge** occurs then $\mathcal{A}$ has submitted a valid ciphertext $(C_1, C_2, C_3, vk^*, \sigma^*)$, which means that $\sigma^*$ is a valid signature for message $(C_1, C_2, C_3)$ under verification key $vk^*$. We show how to build an adversary $\mathcal{A}''$ that breaks the sEUF-1CMA security of $\Sigma$ using $\mathcal{A}$.
Adversary $\mathcal{A}''$ plays the sEUF-1CMA security game with respect to $\Sigma$. At the beginning, it receives a verficiation key $vk^*$ from its challenger. After $\mathcal{A}$ has submitted its choice of corrupted parties adversary $\mathcal{A}''$ picks a uniformly random $x \leftarrow \mathbb{Z}_q$ and executes **DKGSim**($[x]_2, n, k$) on behalf of the uncorrupted parties. Since $\mathcal{A}''$ is in possession of $x$ it is able to simulate all secret keys queried to **Break-In**. If $\mathcal{A}$ makes a valid query **Dec**($t, C_1, C_2, C_3, vk^*, \sigma^*$) then $\mathcal{A}''$ outputs $(C_1, C_2, C_3, \sigma^*)$ as a forgery to its own security experiment. If $\mathcal{A}$ makes a query **Challenge**($t^*, M_0, M_1$) then $\mathcal{A}''$ picks a bit $b$ uniformly at random and computes **Encrypt**($pk, t^*, M_b$) $\rightarrow$ $(C_1, C_2, C_3)$. Afterwards, it queries the signing oracle in its own security experiment on input $(C_1, C_2, C_3)$. It receives a signature $\sigma$ and returns $(t^*, C_1, C_2, C_3, vk^*, \sigma)$ to

$\mathcal{A}$. If $\mathcal{A}$ happens to query $\mathbf{Dec}(C_1, C_2, C_3, vk^*, \sigma^*)$ then, $\mathcal{A}''$ submits $((C_1, C_2, C_3), \sigma^*)$ as its forgery. Note that if the challenge oracle was already queried we still have $((C_1, C_2, C_3), \sigma) \neq ((C_1, C_2, C_3), \sigma^*)$. It follows that

$$\Pr[\mathbf{Forge}] = \varepsilon_{\mathcal{A}''}. \tag{4.17}$$

It remains to determine $\Pr[\mathbf{Coll}]$ for $H$ by building an adversary $\mathcal{A}'''$ that breaks the collision resistance of $\mathcal{H}$. It is eadsy to see that adversary $\mathcal{A}'''$ can simulate the CCA forward security game for $\mathcal{A}$ perfectly by running **KeyGen** and **Sig.KeyGen**. Whenever a collision occurs it forwards the corresponding inputs to $H$ to its own challenger. It follows that

$$\Pr[\mathbf{Coll}] = \varepsilon_{\mathcal{A}'''}. \tag{4.18}$$

Putting (4.17) and (4.18) in (4.16) gives us $\varepsilon_{\mathcal{A}} \leqslant \varepsilon_{\mathcal{A}'} + \varepsilon_{\mathcal{A}''} + \varepsilon_{\mathcal{A}'''}$.
It is easy to see that all algorithms run in approximately the same time. This completes the proof. □

**Theorem 4.7.** *Let $n \geq 2k+1$ and let $\mathcal{A}$ be an adaptive adversary that $(t_{\mathcal{A}}, \varepsilon_{\mathcal{A}})$-breaks the CCA forward security of $(T, n, k)$-$\Pi_{FST}$ from Section 4.6. Given $\mathcal{A}$ we can build an adversary $\mathcal{A}'$ that $(t_{\mathcal{A}'}, \varepsilon_{\mathcal{A}'})$-breaks the CPA security of HIBE $\Pi_{HIBE}$ from [BBG05], an adversary $\mathcal{A}''$ that $(t_{\mathcal{A}''}, \varepsilon_{\mathcal{A}''})$-breaks the sEUF-1CMA security of a signature scheme $\Sigma$, and an adversary $\mathcal{A}'''$ that $(t_{\mathcal{A}''}, \varepsilon_{\mathcal{A}'''})$-breaks the collision resistance of $\mathcal{H}$, such that*

$$t_{\mathcal{A}'''} \approx t_{\mathcal{A}''} \approx t_{\mathcal{A}'} \approx t_{\mathcal{A}} \quad \textit{and} \quad \varepsilon_{\mathcal{A}'''} + \varepsilon_{\mathcal{A}''} + \binom{n}{k}\varepsilon_{\mathcal{A}'} \geqslant \varepsilon_{\mathcal{A}}.$$

*Proof.* Adversary $\mathcal{A}'$ proceeds as in the proof of Theorem 4.6. The only difference is that it guesses in advance of step 1 of **DKGSim** which parties the adversary is going to corrupt. Whenever the adversary corrupts a party $P_j$, $j \in \mathcal{B}$ then it takes over the role of this party and receives all values computed and stored on behalf of this party by $\mathcal{A}'$. If at the end $\mathcal{A}$ outputs a bit but has not corrupted at least $k$ parties then $\mathcal{A}'$ adds some artificial corruptions to the set $\mathcal{B}$ uniformly at random such that it has exactly $k$ corrupted parties. Adversary $\mathcal{A}'$ aborts the simulation and outputs uniformly random bit if a guess was wrong (either of $\mathcal{A}$ or $\mathcal{A}'$). The simulation is successful with probability $1/\binom{n}{k}$.

□

## 4.7 Discussions

**From static to adaptive adversaries.** To protect against adaptive adversaries we used complexity leveraging. This approach results in an additional security loss of $\binom{n}{k}$, where $n$ denotes the number of parties and $k$ a threshold. Although, this loss seems to be quite big, in practice $n$ is relatively small. For instance for a threshold scheme with $10, 20$ or $30$ parties the maximum loss is $2^8, 2^{18}$ and $2^{28}$, respectively. Libert and Yung [LY13b] also achieve security against adaptive adversaries for their FST-PKE. Even their suggestions for FST signature schemes [LY13a] would achieve security against adaptive adversaries.

However, in order to circumvent complexity leveraging they make use of bilinear pairings of composite order. This approach is known as the dual system approach and prior to their work it was only used to achieve full security for (Hierarchical-)IBE and attribute-based encrpytion schemes [Wat09, LOS$^+$10, LW10]. Although the dual system approach is a very powerful tool to obtain full security or security against adaptive adversaries, it lacks efficiency when implemented. The reason for this is that groups of composite order require a much bigger modulus to guarantee the same level of security than elliptic curves on groups of prime order.[13] It can be seen that the security loss in our scheme can be compensated by a slightly bigger modulus. This modulus remains much smaller compared to one of composite order and thus results in a much more efficient scheme. Finally, it should be mentioned that there exist several techniques to transfer the dual system approach to prime order groups [Wat09, Fre10, LW10]. However, they result in larger ciphertexts and seem also to be less efficient in terms of communication rounds for decryption. Moreover, it is not clear whether they can be instantiated without a trusted dealer. We leave it as an open problem to use these techniques to achieve the same efficiency and advantages as our forward-secure threshold schemes, i.e. a *non-interactive* key update and signing(decryption) procedure, *no trusted dealer*, and the possibility to implement the scheme on *standardized elliptic curves*.

**Sizes of keys, signatures, and ciphertexts.** In [DN19], Drijvers and Neven analyzed the signature and key sizes in the context of their forward-secure single user scheme. The structure of the signatures and keys in our threshold scheme is the same structure as of the signatures and keys in their single user scheme. The only difference is that the common parameters in the threshold version have one additional element in $\mathbb{G}_2$. As pointed out in [DN19], a signature consists of one element in $\mathbb{G}_1$, one element in $\mathbb{G}_2$ and one bit string of length $\ell$, where $T = 2^\ell$ is the maximum number of time periods. For all time periods, the secret key contains at most one node key at every level $d = 1, \ldots, \ell$. For level $d$ the node key consists of 1 element in $\mathbb{G}_2$ and $\ell - d + 2$ elements in $\mathbb{G}_1$, which leads to at most $\ell$ elements in $\mathbb{G}_2$ and $\ell^2/2 + 3\ell/2$ elements in $\mathbb{G}_1$. The public key and the common parameters consist of $\ell + 4$ elements in $\mathbb{G}_1$, 3 elements in $\mathbb{G}_2$, and a description of the hash function and the pairing group. This includes the additional element in $\mathbb{G}_2$. For a BLS12-381 curve one element in $\mathbb{G}_1$ requires 32 bits and one element in $\mathbb{G}_2$ requires 40 bits. Assuming $T = 2^{30}$, this yields a signature size of 84 bytes and a secret key shares of size of at most 16800 bytes. The elements from the public key and the common parameters require 1208 bytes. For the FST-PKE, we also need in the common parameters a description of a strong one-time secure signature scheme. The key sizes for the encryption scheme are the same as for the signature scheme. A ciphertext consists of one element from $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_3$ plus the signature and verification key of the strong one-time secure signature scheme.

**The (im-)possibility of mobile adversaries and non-interactive key update.** A mobile adversary can switch between the parties it corrupts. We want to emphasize that in general, it is not possible to tolerate such adversaries while having non-interactive key update. The reason is that an adversary could for instance compromise all parties but only one party per time period. Thus, it could update the secret key shares to the time

[13]For comparison of concrete sizes see the common recommendations: https://www.keylength.com/.

period of the last compromise and reconstruct the secret key without ever exceeding the threshold during one time period, which by security definition basically means that the adversary broke the scheme without having $k+1$ key shares for one time period. This attack is even possible for interactive key updates that require interaction of fewer parties than the threshold. However, prevention against this attack can be given by proactive security. In a nutshell, proactive security means that we can refresh the secret key shares in a way such that all shares which were *not* refreshed cannot be used to reconstruct the secret key anymore (as long as the amount of these shares do not exceed the threshold). Unfortunately, this refreshing requires interaction between the parties or between each party and a trusted dealer, such that allowing mobile adversaries or having non-interactive key updates can be seen as a trade-off. Indeed, our FST schemes can be proactivized.

**On the distinction between proactive and forward security.** Prevention against mobile adversary can be given by proactive security. In a nutshell, proactive security means that we can refresh the secret key shares in a way such that all shares which were *not* refreshed cannot be used to reconstruct the secret key anymore (as long as the amount of these shares do not exceed the threshold). Unfortunately, this refreshing requires interaction between the parties or between each party and a trusted dealer, such that allowing mobile adversaries or having non-interactive key updates can be seen as a trade-off. We explain the differences between proactive and forward security in more details below. In forward security, the public key is fixed but the secret key changes in regard to the time period. The time period is also embedded in the secret key. If the secret key of time period $t$ gets exposed, it is possible to sign signatures for all time periods $t' \geqslant t$, but it is still infeasible to do so for time periods $t' < t$. Proactive security is an additional *interactive* security mechanism for secret sharing. This mechanism changes the secret shares, but not the secret itself and as in forward security, the public key remains unchanged. Since proactive security does not embed the time period it can be seen as a parallel time line when combined with forward security. For instance, let $k_1$ be shares from a time period $t$ and $k_2$ be shares from the same time period $t$ but after proactivization. Further, let $k_1 + k_2 \geqslant k+1$, but $k_1, k_2 < k+1$, where $k$ denotes the threshold. Then, it is *not* possible to reconstruct the secret from the shares $k_1$ and $k_2$, although they belong to the same time period. Overall, proactivization can provide additional security but it must be executed carefully since every set of players of size at most $k$ which did not participate in proactivizing is forever excluded from signing.

**Tolerating mobile adversaries.** In order to proactivize the key material in our schemes, the users had to run the **DKG** protocol where each party $P_i$ sets the constant term of polynomial $a_i$ to 0. Then, the resulting share held by each party after this **DKG** execution is multiplied to all terms $d_i'$ in its secret key shares. It is also worth mentioning that in order to tolerate mobile adversaries, having a non-interactive key update procedure and a separate protocol for proactivization is still preferable to having an interactive key update procedure. The reason is that the former allows different levels of granularity. For instance, key update could happen every hour and proactivization once per day. Proactive security was introduced by Ostrovsky and Yung [OY91]. Herzberg et al. proposed techniques to achieve robust proactivization for polynomial secret sharing [HJKY95, HJJ$^+$97].

**Conclusion.** We proposed a *forward-secure threshold schemes* signature and encrption scheme. Bot of them improve the state of the art in many aspects. The main advantages of our schemes are the *non-interactive key update and signing(decryption) procedure*, the fact that it can be implemented on *standardized pairing-friendly curves* and, that *no trusted dealer* is required. Our signature, ciphertext and key sizes are also quite short. We proved our scheme secure against adaptive and robust against malicious adversaries.Morever, it is possible to add a proactivization procedure to our schemes in order to tolerate mobile adversaries.

CHAPTER 5

# Efficient Adaptively-Secure Cryptosystems via Near-Collision Resistance

*We construct adaptively-secure variants of the selectively-secure pairing-based IB-KEM and digital signature of Boneh and Boyen [BB04a, BB04c]. Both are proven secure in the standard model, based on q-type assumptions, and have public key size $O(\lambda)$, where $\lambda$ denotes the security parameter. The IB-KEM and the digital signature scheme have only a single group element as ciphertext and signature, respectively. Moreover, the security reductions are quadratically tighter than in the corresponding schemes by Jager and myself [JK18, Asiacrypt 2018]. As a technical contribution we introduce blockwise partitioning, which leverages the assumption that a cryptographic hash function is Near-Collision Resistant to prove full adaptive security of cryptosystems.*
*The results in this chapter are based on collaborations with Tibor Jager and David Niehues. The concept of Near-Collision Resistance is due to David Niehues and inspired by Truncation Collision Resistance from Tibor Jager and myself [JK18, Asiacrypt 2018]. The construction of IB-KEM and digital signatures are mainly due to myself. The results are part of an ongoing submission.*

## 5.1 Introduction

In the random oracle model (ROM) [BR93a] a cryptographic hash function is modeled as an oracle that implements a truly random function. This approach provides a very powerful tool to prove security of cryptosystems. For example, it enables to adaptively "program" a hash function to map certain input values to specific output values in the security proof. However, the random oracle is a hypothetical concept and in practice it requires implementation with a standard cryptographic hash function, like SHA-3. This approach incurs the assumption that the implemented hash function is "secure enough" for the given application, but does not require precise security properties. In addition to these missing security properties, there are also well-known difficulties of instantiating random oracles up to the point that a scheme can be proven secure in the ROM, but being insecure when implemented[CGH98]. A fundamental question is to which extent the ROM is indeed necessary to obtain practical constructions of cryptosystems. By advancing our proof techniques for cryptographic schemes, we may eventually be able to construct secure schemes in the standard model with the same efficiency as corresponding schemes with security proofs in the ROM. It is known that for some primitives a programmable random oracle is indeed inherently necessary [Nie02, FLR$^+$10, HMS12, FF13]. But for *adaptively-secure* identity-based encryption schemes and standard signatures schemes, there are no such impossibility results.

**Techniques to achieve adaptive security.** In the context of identity-based encryption, the established standard security notion is called *adaptive* security [BF01]. This notion considers an adversary that is able to choose the identities for which it requests secret keys or a challenge ciphertext *adaptively* in the security experiment. A weaker notion is given by so-called *selective* security [BB04a], where the adversary has to announce the "target identity" associated with a challenge ciphertext at the beginning of the security experiment, even before seeing the public parameters. There exists also a distinction between adaptive and selective security for signature schemes [BLS04, BB04c]. In the case of signature schemes, the adversary can either query signatures adaptively or it has to announce for which messages it wants to receive signatures before seeing the public parameters. The advantage of "selective" security is that it is much easier to achieve and therefore yields more efficient constructions. The ROM can then be used to convert a selectively-secure scheme into an adaptively-secure one with negligible performance overhead, and thus yields an efficient and adaptively-secure construction. This generic transformation is based on "programming" the random oracle, which essentially means that it is possible to adaptively modify the mapping of function inputs to outputs in a way that is convenient for the security proof. Although this is very useful to achieve efficient and adaptively-secure constructions, programming the random oracle is considered as particularly unnatural, because no fixed function can be as freely programmed as a random oracle. There exist techniques to achieve adaptive security in the standard model by realizing certain properties of a random oracle with a concrete construction (i. e., in the standard model). This includes *admissible hash functions* [BB04b], *programmable hash functions* [Wat05, HK08, HJK11, FHPS13, CFN15], and *extremely lossy functions* [Zha16]. However, these typically yield significantly less efficient constructions and are therefore less interesting for practical applications than corresponding contructions in the random oracle model.

**Truncation Collision Resistance by Jager and Kurek.** A different approach to obtain adaptive security is to use *Truncation Collision Resistance* [JK18] of a cryptographic hash function to achieve adaptive security. In contrast to the aforementioned approaches, this does not introduce a new "algebraic" construction of a hash function. Instead, the idea is to formulate a concrete hardness assumption that on the one hand is "weak enough" to appear reasonable for *standard* cryptographic hash functions, such as SHA-3, but which at the same time is "strong enough" to be used to achieve adaptive security. It is shown that this indeed yields very efficient and adaptively-secure constructions, such as identity-based encryption with a single group element overhead and digital signatures that consist of a single group element. However, the main disadvantages of the constructions in [JK18] are that very strong computational hardness assumptions (so-called $q$-type assumptions with very large $q$) are required, and that the reductions are extremely non-tight.

**Our contributions.** We introduce *blockwise partitioning* as an approach to leverage the assumption that a cryptographic hash function is *near-collision resistant.* Near-collision resistance is inspired by the notion of Truncation Collision Resistance by Jager and

| Scheme | $\lvert\mathsf{mpk}\rvert$ | $\lvert C\rvert$ | Security | Assumption | ROM | Security Loss |
|---|---|---|---|---|---|---|
| [BF01] | 2 | 1 | adap. | DBDH | Yes | $O(q_{key})$ |
| [Wat05] | $n+3$ | 2 | adap. | DBDH | No | $O(t^2 + (n \cdot q_{key} \cdot \varepsilon^{-1})^2)$ |
| [Wat09] | 13 | 10 | adap. | DLIN+DBDH | No | $O(q_{key})$ |
| [Lew12] | 25 | 6 | adap. | DLIN | No | $O(q_{key})$ |
| [CLL+13] | 9 | 4 | adap. | SXDH | No | $O(q_{key})$ |
| [AHY15] | $O(\lambda)$ | 8 | adap. | DLIN | No | $O(\log(\lambda))$ |
| [BB04a] | 4 | 2 | selec. | qDBDHI | No | $O(1)$ |
| [JK18] | $O(\lambda)$ | 1 | adap. | qDBDHI | No | $O(t_{\mathcal{A}}^7/\varepsilon_{\mathcal{A}}^4)$ |
| Ours | $O(\lambda)$ | 1 | adap. | qDBDHI | No | $O(t_{\mathcal{A}}^3/\varepsilon_{\mathcal{A}}^2)$ |

Figure 5.1: Comparison of IB-KEMs based on pairings with prime order groups and short ciphertexts. $\lvert\mathsf{mpk}\rvert$ is the number of group elements in public keys (descriptions of groups and hash functions not included), $n$ the length of identities, $\lambda$ the security parameter. All public keys include at least one element from the target group of the pairing, except for [BF01]. $\lvert C\rvert$ is the number of group elements in the ciphertexts when viewed as a KEM, respectively. "Adap." means adaptive IND-ID-CPA security as defined below, "selec." is selective security in the sense of [BB04a]. The security loss is defined as the value $L$ that satisfies $t_{\mathcal{B}}/\varepsilon_{\mathcal{B}} = L \cdot t_{\mathcal{A}}/\varepsilon_{\mathcal{A}}$, where $t_{\mathcal{A}}$,$\varepsilon_{\mathcal{A}}$ and $t_{\mathcal{B}}$,$\varepsilon_{\mathcal{B}}$ are the running time and advantage probability of the adversary and reduction, respectively. $q_{key}$ is the number of identity key queries

*Kurek*[JK18]. It essentially captures the same intuition and therefore can be considered equally reasonable. We show that Near-Collision Resistance yields more efficient cryptographic schemes. More precisely, we construct a new variant of the pairing-based identity-based encryption schemes of Boneh and Boyen [BB04a]. In comparison to [BB04a], we achieve adaptive security instead of selective security and do not require the random oracle model. Our scheme is based on a $q$-type assumption and a ciphertext consists only of a *single* group element. In comparison to the corresponding construction from Jager and Kurek [JK18], the $q$ of the required $q$-type assumption is reduced *quadratically*, while the tightness of the reduction is improved *quadratically*, as well. We also construct a signature scheme with adaptive security in the standard model, where a signature is only a single element from a prime-order group, which achieves the same quadratic improvement over a construction from [JK18]. Again, we do not require the random oracle model. We compare existing identiy-based encryption schemes and signatures in Fig. 5.1 and Fig. 5.2, respectively. We compare Truncation Collision Resistance with near-collision resistance in Section 5.5.

## 5.2 Blockwise Partitioning via Near-Collision Resistance

*Confined guessing* [BHJ+13, BHJ+15] is a technique to construct efficient and adaptively-secure digital signature schemes. It has been used for instance in [DM14, Alp15]. Unfortunately, it is only applicable to signatures, but not to identity-based schemes such

| **Scheme** | $\lvert pk\rvert$ | $\lvert\sigma\rvert$ | **Security** | **Assumption** | **ROM** | **Security Loss** |
|---|---|---|---|---|---|---|
| [BLS04] | 2 | 1 | adap. | CDH | Yes | $O(q_{Sig})$ |
| [BB04c] | 5 | 2 | selec. | qDH | No | $O(1)$ |
| [Wat05] | $n+3$ | 2 | adap. | CDH | No | $O(n \cdot q_{Sig})$ |
| [HJK11] | $n+\lambda+3$ | 2 | adap. | qDH | No | $O(n^2 \cdot q_{Sig})$ |
| [BHJ+13] | $O(\log\lambda)$ | 3 | adap. | CDH | No | $O((\varepsilon^{-1} \cdot q_{Sig}^{m+1})^{c/m})$ |
| [JK18] | $O(\lambda)$ | 1 | adap. | qDH | No | $O(t_{\mathcal{A}}^7/\varepsilon_{\mathcal{A}}^4)$ |
| Ours | $O(\lambda)$ | 1 | adap. | qDH | No | $O(t_{\mathcal{A}}^3/\varepsilon_{\mathcal{A}}^2)$ |

Figure 5.2: Comparison of short signature schemes based on pairing with prime order groups. The column $|\mathbb{G}|$ refers to the order of the underlying groups (prime or composite), $|pk|$ is the number of group elements in public keys, $n$ is the length of messages, and $\lambda$ the security parameter. All public keys include one element from the target group of the pairing, except for [BLS04, HJK11, BHJ+13]. The column $|\sigma|$ refers to the number of group elements in the signature. "Adaptive" security means EUF-CMA security, "selec." security is from [BB04c]. The remaining columns state the assumption the proof is based on, whether the Random Oracle Model is used, and the security loss of the reduction, where $q_{Sig}$ is the number of signing queries, $t_{\mathcal{A}}$ and $\varepsilon_{\mathcal{A}}$ the running time and advantage of the adversary, and the loss is computed as explained in Figure 5.1. The values $m$ and $c$ are system parameters influencing keys and signature sizes. Note that [HJK11] present also other trade-offs with larger public keys consisting and shorter signatures, but always strictly larger than one group element.

as identity-based key encpsulation mechanisms (IB-KEMs). We propose *blockwise partitioning* as a similarly technique, and show how it can be used to construct efficient IB-KEMs and signature schemes with adaptive security. It is based on the *near-collision resistance* of a cryptographic hash function, which is inspired by the closely related notion of *truncation collision resistance*by Jager and *Kurek* [JK18]. The main advantage of near-collision resistance is that it enables a more fine-grained guessing compared to truncation collision resistance, which results in more efficient constructions due to tighter security reductions and weaker hardness assumptions. Additionally, near-collision resistance enables the use of hash functions with smaller output length.The constructions in [JK18] require a hash function output length of $4(\lambda + 1)$, where $\lambda$ is the security parameter. Here, a hash function output length of $2\lambda + 3$ is sufficient. Due to the fact that in practice the common output length for collision-resistant hash functions is approximately $2\lambda$ this is basically optimal. In particular, for many practical construction collision resistant hash function are used to map long idetities to short strings anyway. In this section we describe the framework and assumptions for blockwise partitioning and give some more technical intuition. Moreover, we state and prove a technical lemma that provides useful properties of blockwise partitioning for security proofs.

### 5.2.1 Blockwise Partitioning.

Let $H : \{0,1\}^* \to \{0,1\}^n$ be a hash function. W.l.o.g. we assume that $n = \sum_{i=0}^{\ell} 2^i$. One can generalize this to arbitrary $n$, but this would make the notation rather cumbersome without providing additional insight or clarity. Then we can view the output space $\{0,1\}^n$ of the hash function as a direct product of sets of exponentially-increasing size

$$\{0,1\}^n = \{0,1\}^{2^0} \times \cdots \times \{0,1\}^{2^\ell}.$$

For a hash function $H$ we define functions $H_1, \ldots, H_\ell$ such that

$$H_i : \{0,1\}^* \to \{0,1\}^{2^i} \qquad \text{and} \qquad H(x) = H_0(x)||\cdots||H_\ell(x).$$

One can consider each $H_i(x)$ as one “block” of $H(x)$. Note that blocks have exponentially increasing size and there are $\lfloor \log n \rfloor + 1$ blocks in total.

**Remark 5.1.** *Note that in this chapter the notation $H_i$ has a different meaning than in Chapter 3.*

**Using blockwise partitioning.** Let $t = t(\lambda)$ be a polynomial and let $\varepsilon = \varepsilon(\lambda)$ be a non-negligible function such that $\varepsilon > 0$ and $t/\varepsilon < 2^\lambda$ for all $\lambda \in \mathbb{N}$. In the security proofs of our cryptosystems, $t$ and $\varepsilon$ are (approximations of) the running time and success probability of the adversary. We define an integer $n'$ depending on $(t, \varepsilon)$ as

$$n' := \lceil \log(4t \cdot (2t-1)/\varepsilon) \rceil. \tag{5.1}$$

Note that if $n \geq 2\lambda + 3$, then we have $0 \leq n' \leq n$ as we show in Lemma 5.1 below.

The value $n'$ uniquely determines an index set $\mathcal{I} = \{i_1, \ldots, i_\omega\} \subseteq \{0, \ldots, \ell\}$ such that $n' = \sum_{i \in \mathcal{I}} 2^i$, where $\ell := \lfloor \log n \rfloor$. The key point in defining $n'$ as in Eq. (5.1) is that it provides the following two properties simultaneously:

**Guessing from polynomially-bounded range.** In order to transform a cryptographic scheme from selective to adaptive security the reductions has to "predict" a certain hash values $H(x^*)$. Think of $x^*$ as the challenge identity in an IB-KEM security experiment, or the message from the forgery in a signature security experiment. Blockwise partitioning enables to do this from a polynomially bounded range as as follows.

Consider the following probabilistic algorithm $\mathsf{BPSmp}$ On input $\lambda$, $t$, and $\varepsilon$ it computes $n'$ as in Eq. (5.1). Then it chooses $K_i \leftarrow \{0,1\}^{2^i}$ uniformly random for $i \in \mathcal{I}$ and defines $K_i = \perp$ for all $i \notin \mathcal{I}$. Finally it outputs

$$(K_0, \ldots, K_\ell) \leftarrow \mathsf{BPSmp}(1^\lambda, t, \varepsilon).$$

The joint range of all hash functions $H_i$ with $i \in \mathcal{I}$ is $\{0,1\}^{2^{i_1}} \times \cdots \times \{0,1\}^{2^{i_\omega}}$, has size

$$2^{n'} = 2^{\sum_{i \in \mathcal{I}} 2^i}.$$

Hence, we have

$$\Pr\left[H_i(x^*) = K_i \text{ for all } i \in \mathcal{I}\right] = 2^{-n'}.$$

Note that $2^{n'}$ is *polynomially bounded*, due to the definition of $n'$ in Eq. 5.1.

**Upper bound on the collision probability.** In Lemma 5.1 below we will show that near-collision resistance of $H$ guarantees that the probability that an adversary running in time $t$ outputs any two values $x \neq x'$ such that

$$H_i(x) = H_i(x') \qquad \text{for all } i \in \mathcal{I} \tag{5.2}$$

is at most $\varepsilon/2$. In the context of IB-KEMs $x$ and $x'$ could be adaptively chosen identities. In the context of digital signature schemes, $x$ and $x'$ could be adaptively chosen messages. Note that this does *not* mean that the collision probability is negligible. In fact, this is not even possible due to the polynomially-bounded space. However, there will be no collision with probability at least $\varepsilon/2$, which means that collisions are sufficiently unlikely: an adversary that runs in some time $t$ and has some advantage $\varepsilon$ will be often enough successful *without* finding a collision.

## 5.2.2 Blockwise Partitioning Via Near-Collision Resistance.

We give a formal definition of near-collision resistance and then provide a technical lemma which will be useful for security proofs based on blockwise partioning of hash function outputs.

**Definition 5.1. Near-collision resistance** Let $\mathcal{H} = \{H : \{0,1\}^* \to \{0,1\}^n\}$ be a family of hash functions. For $n' \in \{1, \ldots, n\}$, we say that an adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ $n'$-breaks the near-collision resistance of $\mathcal{H}$, if it runs in time $t_\mathcal{A}$, $\mathcal{A}_1$ on input $n'$ outputs an index set $I \subseteq \{1, ..., n\}$ with $|I| = n'$ and

$$\Pr_{H \leftarrow \mathcal{H}}\left[\begin{array}{l}(x_0, \ldots x_Q) \leftarrow \mathcal{A}_2(H) : \\ \exists u, v \text{ s.t. } H(x_u)[i] = H(x_v)[i] \; for \; all \; i \in I\end{array}\right] > \frac{t_\mathcal{A}(t_\mathcal{A} - 1)}{2^{n'+1}},$$

where $H(\cdot)[i]$ denotes the $i$-th bit of $H$. We say that $\mathcal{H}$ is *near-collision resistant*, if there exists no adversary $\mathcal{A}$ $n'$-breaking the near-collision resistance of $H$ for any $n' \in \{1, \dots, n\}$.

The following lemma will be useful to apply blockwise partitioning in security proofs.

**Lemma 5.1.** *Let $H : \{0,1\}^* \to \{0,1\}^n$ be a hash function, $t$ be a polynomial, and let $\varepsilon$ be a non-negligible function such that $\varepsilon > 0$ and $t/\varepsilon < 2^\lambda$ for all $\lambda$. Let $n' := \lceil \log(4t \cdot (2t-1)/\varepsilon) \rceil$ as in Eq. 5.1 and define set $\mathcal{I}$ such that $n' = \sum_{i \in \mathcal{I}} 2^i$. Let $\mathcal{A}$ be an algorithm that outputs $(X^{(1)}, \dots, X^{(Q)}, X^*)$ and runs in time $t$ and let*

$$(K_0, \dots, K_\ell) \leftarrow \mathsf{BPSmp}(1^\lambda, t, \varepsilon),$$

*where $\mathsf{BPSmp}$ is the algorithm described above.*

1. *Let $\mathsf{coll}$ be the event that there exists $x, x' \in \{X^{(1)}, \dots, X^{(Q)}, X^*\}$ such that*

$$H_i(x) = H_i(x') \text{ for all } i \in \mathcal{I}. \tag{5.3}$$

   *Let $\mathsf{badChal}$ be the event that there exists $i \in \mathcal{I}$ such that $\Pr[H_i(X^*) \neq K_i]$. If $H$ is drawn uniformly at random from a family of near-collision resistant hash functions in the sense of Definition 5.1, then we have*

$$(\varepsilon - \Pr[\mathsf{coll}]) \cdot \Pr[\neg\mathsf{badChal}] \geq \varepsilon^2/(32t^2 - 16t).$$

   *Moreover, $\mathsf{coll}$ and $\mathsf{badChal}$ are independent of each other.*

2. *Let $\mathsf{badEval}$ be the event that there exists $x \in \{X^{(1)}, \dots, X^{(Q)}\}$ with $x \neq X^*$ such that $H_i(x) = K_i$ for all $i \in \mathcal{I}$. Then we have*

$$\mathsf{badEval} \implies \mathsf{coll} \vee \mathsf{badChal}.$$

*Proof.* The proof uses the following inequalities and identities from [JK18, JN19].

$$n' \in \{1, \dots, 2k+3\}, \qquad \frac{2t(2t-1)}{2^{n'}} \leq \frac{\varepsilon}{2}, \text{ and } \qquad \frac{1}{2^{n'}} \geq \frac{\varepsilon}{16t^2 - 8t} \tag{5.4}$$

For a complete overview we start by rephrasing the proof of these identites and inequalities from [JK18, JN19]. We start by proving $n' \in \{1, \dots, 2\lambda + 3\}$.

$$\begin{aligned} n' = \lceil \log(4t(2t-1)/\varepsilon) \rceil &\leq \left\lceil \log\left(4 \cdot 2^\lambda (2t-1)\right) \right\rceil \\ &\leq \left\lceil \log\left(8 \cdot 2^\lambda t\right) \right\rceil \leq \left\lceil \log\left(2^\lambda 2^{\lambda+3}\right) \right\rceil = 2\lambda + 3 \end{aligned}$$

Since $4t(2t-1) = 8t^2 - 4t > 1$ for all $t \in \mathbb{N}$ and $\varepsilon \in (0,1]$, we have $\log(4t(2t-1)/\varepsilon) > 0$ and therefore $j \geq 1$.

We proceed to prove $2t(2t-1)/2^{n'} \leq \varepsilon/2$.

$$\frac{2t(2t-1)}{2^{n'}} = \frac{2t(2t-1)}{2^{\lceil \log(4t(2t-1)/\varepsilon) \rceil}} \leq \frac{\varepsilon 2t(2t-1)}{4t(2t-1)} = \frac{\varepsilon}{2}$$

Finally, we have

$$\frac{1}{2^{n'}} = \frac{1}{2^{\lceil \log(4t(2t-1)/\varepsilon) \rceil}} \geq \frac{1}{2} \cdot \frac{\varepsilon}{4t(2t-1)} = \frac{\varepsilon}{16t^2 - 8t}.$$

Now we are ready to continue with proving Property 1 by showing $\Pr[\mathsf{coll}] < \varepsilon/2$. Assume an algorithm $\mathcal{A}$ running in time $t_{\mathcal{A}}$ and outputting $(X^{(1)}, \ldots, X^{(Q)}, X^*)$ such that there exist $x, x' \in \{X^{(1)}, \ldots, X^{(Q)}, X^*\}$ such that Eq. (5.3) holds with probability at least $\varepsilon/2$. By the definition of $\mathcal{I}$ and the functions $H_i$, this gives us that $H(x)$ and $H(x')$ agree on at least $n'$ positions. From $\mathcal{A}$, we can construct an algorithm $\mathcal{B}$ that $n'$-breaks the near-collision resistance of $\mathcal{H}$. In order to do so, $\mathcal{B}$ simply outputs $(X^{(1)}, \ldots, X^{(Q)}, X^*)$ provided by $\mathcal{A}$. The running time $t_{\mathcal{B}}$ of $\mathcal{B}$ is at most $2t_{\mathcal{A}}$, since $\mathcal{B}$ does nothing more than executing $\mathcal{A}$ and transmitting its outputs. Therefore, we get

$$\Pr[\mathsf{coll}] > \varepsilon_{\mathcal{A}}/2 \geq \frac{2t_{\mathcal{A}}(2t_{\mathcal{A}} - 1)}{2^{n'}} \geq \frac{t_{\mathcal{B}}(t_{\mathcal{B}} - 1)}{2^{n'+1}},$$

where the second inequality follows from Eq. (5.4). This contradicts the assumptions that $\mathcal{H}$ is near-collision resistant. Next, we determine $\Pr[\neg\mathsf{badChal}]$. We have that the events $\mathsf{coll}$ and $\mathsf{badChal}$ are independent of each other because $(K_0, \ldots, K_\ell)$ is chosen independently from $(X^{(1)}, \ldots, X^{(Q)}, X^*)$. Moreover, each $K_i$ with $i \in \mathcal{I}$ is chosen uniformly at random from $\{0,1\}^{2^{2^i}}$ and thus we have

$$\Pr[\neg\mathsf{badChal}] = \Pr[H_i(X^*) = K_i \text{ for all } i \in \mathcal{I}] = \frac{1}{2^{\sum_{i \in \mathcal{I}} 2^i}} = 2^{-n'},$$

where the last equation follows by definition of $n'$. Finally, to prove Property 1, we calculate

$$(\varepsilon_{\mathcal{A}} - \Pr[\mathsf{coll}])2^{-n'} \geq \left(\varepsilon_{\mathcal{A}} - \frac{\varepsilon_{\mathcal{A}}}{2}\right) \frac{\varepsilon_{\mathcal{A}}}{16t_{\mathcal{A}}^2 - 8t_{\mathcal{A}}} = \frac{\varepsilon_{\mathcal{A}}^2}{32t_{\mathcal{A}}^2 - 16t_{\mathcal{A}}},$$

where the first inequality follows from Eq. (5.4). To show Property 2, we explain that if $\mathsf{badEval}$ occurs, then either $\mathsf{badChal}$ or $\mathsf{coll}$ *must occur*. This is because if there exists $x \in \{X^{(1)}, \ldots, X^{(Q)}\}$ with $x \neq X^*$ with $H_i(x) = K_i$ for all $i \in \mathcal{I}$, then we have either that also $H_i(X^*) = K_i$ for all $i \in \mathcal{I}$ and then $\mathsf{coll}$ occurs or we have that there exists an index $i \in \mathcal{I}$ such that $H_i(X^*) \neq K_i$ and then $\mathsf{badChal}$ occurs. This concludes the proof. □

## 5.3 Adaptively Secure IB-KEM with Short Ciphertexts

In this section, we present a new IB-KEM that is adaptively secure and where the ciphertext consists of only a *single* element. Compared to the only other construction with these properties[JK18], the $\bar{q}$ of the required $\bar{q}$-type assumption is reduced *quadratically*, while the tightness of the reduction is improved *quadratically*, as well. Due to near-collision resistance, we are also able to reduce the output length of the hash function to roughly half of the output length required in [JK18], which reduces computational costs while guaranteeing the same level of security.

**The construction.** Let $\mathcal{H} = \{H : \{0,1\}^* \to \{0,1\}^{2\lambda+3}\}$ be a family of hash functions, let $\ell = \lfloor \log(2\lambda + 3) \rfloor$, and let $\mathcal{PG}$ be the description of a cryptographic Type-1 pairing group. We construct the IB-KEM scheme $\Pi = (\mathbf{Setup}, \mathbf{KeyGen}, \mathbf{Encap}, \mathbf{Decap})$ as follows.

- $\mathbf{Setup}(1^\lambda)$. Choose a random hash function $H \leftarrow \mathcal{H}$, a random generator $[1]_1 \in \mathbb{G}_1$, and random elements $x_0, \ldots, x_\ell \in \mathbb{Z}_q^*$. Define the master secret key $\mathsf{msk}$ as

  $$\mathsf{msk} = (x_0, \ldots, x_\ell) \in \mathbb{Z}_q^{\ell+1}.$$

  For $i \in \mathbb{N}$ and $m \in \mathbb{N}_0$ define $b_i(m)$ as the function that, on input of integer $m$, outputs the $i$-th bit of the binary representation of $m$. For $\mathsf{msk} = (x_0, \ldots, x_\ell)$ and $m = 0, \ldots, 2^{\ell+1} - 1$ define

  $$F(\mathsf{msk}, m) := \prod_{i=0}^{\ell} x_i^{b_i(m)}. \tag{5.5}$$

  The public parameters are defined as

  $$\mathsf{mpk} = (\mathcal{PG}, H, [F(\mathsf{msk}, 0)]_1, \ldots, [F(\mathsf{msk}, 2^{\ell+1} - 1]_1).$$

- $\mathbf{KeyGen}(\mathsf{msk}, id)$. Let

  $$u(id) = \prod_{i=0}^{\ell} (H_i(id) + x_i) \in \mathbb{Z}_q. \tag{5.6}$$

  Then the private key for identity $id$ is computed as

  $$USK_{id} = [1/u(id)]_1.$$

- $\mathbf{Encap}(\mathsf{mpk}, id)$. Observe that

  $$u(id) = \prod_{i=0}^{\ell} (H_i(id) + x_i) = d_0 + \sum_{m=1}^{2^\ell - 1} \left( d_m \prod_{i=0}^{\ell} x_i^{b_i(n)} \right),$$

  where the constants $d_i$ are efficiently computable from $H(id)$. Using $H(id)$ and $\mathsf{mpk}$ first $[u(id)]_1$ is computed as

  $$[u(id)]_1 = \left[ d_0 + \sum_{m=1}^{2^\ell - 1} \left( d_m \prod_{i=0}^{\ell} x_i^{b_i(n)} \right) \right]_1 = [d_0]_1 \cdot \prod_{m=1}^{2^\ell - 1} [F(\mathsf{msk}, m)]_1^{d_m}.$$

  Note that this does not require knowledge of $x_0, \ldots, x_\ell$ explicitly.

  Finally, the ciphertext and key are computed as

  $$(C, K) = ([u(id)]_1^r, e([1]_1, [1]_1)^r) \in \mathbb{G}_1 \times \mathbb{G}_T.$$

for a uniformly random $r \leftarrow \mathbb{Z}_q$.

- **Decap**$(USK_{id}, C, id)$. To recover $K$ from a ciphertext $C$ for identity $id$ and a matching user secret key $[1/(u(id))]_1$, compute and output $e(C, USK_{id})$.

### 5.3.1 Proof of Correctness

Let $C$ be a ciphertext for identity $id$ computed as above. Let $[1/(u(id))]_1$ be a user secret key for identity $id$. Then we have

$$e(C, USK_{id}) = e([u(id)]_1^r, [1/(u(id))]_1) = e([1]_1, [1]_1)^r = K.$$

### 5.3.2 Proof of Security

The security of this construction is based on the $\bar{q}$-DBDHI assumption for Type-1 pairing groups [BB04a, BBG05]. It is the same assumption as for the scheme of [BB04a]. To simplify notation in the security proof, we re-randomize the group elements with an element $y \leftarrow \mathbb{Z}_q^*$. This has no impact on the hardness of the assumption because $[1]_1 \leftarrow \mathbb{G}_1$ is picked uniformly random anyway.

**Definition 5.2. $\bar{q}$-BDHI$_1$ Assumption.**
Let $\mathcal{PG} = (\mathbb{G}_1, \mathbb{G}_T, e, q)$ be the description of a cryptographic Type-1 pairing group and let $[1]_1$ be a random generator of $\mathbb{G}_1$. Let $\mathcal{A}$ be an adversary. We say that it $(t_\mathcal{A}, \varepsilon_\mathcal{A})$-breaks the $\bar{q}$-DBDHI$_1$ assumption, if it runs in time $t_\mathcal{A}$ and

$$\Big| \Pr\left[\mathcal{A}\left(\mathcal{PG}, [y]_1, [y\alpha]_1, [y\alpha^2]_1, \ldots, [y\alpha^{\bar{q}}]_1, V_0\right) = 1\right] - \Pr\left[\mathcal{A}\left(\mathcal{PG}, [y]_1, [y\alpha]_1, [y\alpha^2]_1, \ldots, [y\alpha^{\bar{q}}]_1, V_1\right) = 1\right] \Big| \geq \varepsilon_\mathcal{A}$$

where $y, \alpha \leftarrow \mathbb{Z}_q^*, V_0 = e([y]_1, [y]_1)^{1/\alpha}$ and $V_1 \leftarrow \mathbb{G}_T$.

**Remark 5.2.** *In [BBG05], it is shown that the $\bar{q}$-DBDHI assumption and the $\bar{q}$-DBDHI* assumption from Definition 4.8 are equivalent. However, note that here Definition 4.8 refers to Type-3 pairing groups, whereas Definition 5.2 refers to Type-1 pairings. In order to show equivalence, the same type of pairing is required.*

We start by defining the strength of the $\bar{q}$-DBDHI assumption. We set

$$\bar{q} := 4\lambda + 7 + j + 2 \sum_{\substack{i \in [\lfloor \log(2\lambda+3) \rfloor]_0 \\ K_i \neq \perp}} \left(2^{2^i} - 1\right).$$

Using $j \leq \lfloor \log(2\lambda + 3) \rfloor + 1$ and the following lemma, we obtain

$$\bar{q} \leq 4\lambda + 8 + \lfloor \log(2\lambda + 3) \rfloor + 32 t_\mathcal{A}^2 / \varepsilon_\mathcal{A}.$$

**Lemma 5.2.** *Let $\mathcal{I} = \{i : K_i \neq \perp\}$ be as above, then*

$$2 \cdot \sum_{\substack{i \in [\lfloor \log(2\lambda+3) \rfloor]_0 \\ i \in \mathcal{I}}} \left(2^{2^i} - 1\right) \leq \frac{32t_{\mathcal{A}}^2}{\varepsilon_{\mathcal{A}}}.$$

*Proof.*

$$\begin{aligned} 2 \cdot \sum_{\substack{i \in [\lfloor \log(2\lambda+3) \rfloor]_0 \\ K_i \neq \perp}} \left(2^{(2^i)} - 1\right) &< 2 \cdot \sum_{\substack{i \in [\lfloor \log(2\lambda+3) \rfloor]_0 \\ K_i \neq \perp}} 2^{(2^i)} < \prod_{\substack{i \in [\lfloor \log(2\lambda+3) \rfloor]_0 \\ K_i \neq \perp}} 2^{(2^i)} \qquad (5.7) \\ &= 2 \cdot 2^{\sum_{\substack{i \in [\lfloor \log(2\lambda+3) \rfloor]_0 \\ K_i \neq \perp}} (2^i)} = 2^{\ell} \\ &= 2 \cdot 2^{\lceil \log(4t_{\mathcal{A}}(2t_{\mathcal{A}}-1)/\varepsilon_{\mathcal{A}}) \rceil} \leq \frac{8t_{\mathcal{A}}(2t_{\mathcal{A}}-1)}{\varepsilon_{\mathcal{A}}} \\ &\geq 2 \cdot \frac{16t_{\mathcal{A}}^2}{\varepsilon_{\mathcal{A}}} = \frac{32t_{\mathcal{A}}^2}{\varepsilon_{\mathcal{A}}}, \end{aligned}$$

where Inequality (5.7) holds, because $a + b < ab$ for all $a, b \geq 2$. This concludes the proof. □

**Theorem 5.1.** *Let $\Pi$ from Section 5.3 be instantiated with a family $\mathcal{H}$ of near-collision resistant hash functions in the sense of Definition 5.1. Let $\mathcal{A}$ be an adversary that $(t_{\mathcal{A}}, \varepsilon_{\mathcal{A}})$-breaks the IND-ID-CPA security of $\Pi$. Given $\mathcal{A}$, we can build an adversary $\mathcal{B}$ that, given (sufficiently close approximations of) $t_{\mathcal{A}}$ and $\varepsilon_{\mathcal{A}}$, $(t_{\mathcal{B}}, \varepsilon_{\mathcal{B}})$-breaks the $\bar{q}$-*DBDHI *assumption with $\bar{q} \leq 4\lambda + 9 + \lfloor \log(2\lambda+3) \rfloor + 32t_{\mathcal{A}}^2/\varepsilon_{\mathcal{A}}$ such that*

$$t_{\mathcal{B}} = O(32t_{\mathcal{A}}^2/\varepsilon_{\mathcal{A}}) \quad \textit{and} \quad \varepsilon_{\mathcal{B}} \geq \varepsilon_{\mathcal{A}}^2/(32t_{\mathcal{A}}^2 - 16t_{\mathcal{A}}) - \mathsf{negl}(\lambda),$$

*for some negligible term* $\mathsf{negl}$.

*Proof.* Consider the following sequence of games. We denote with $G_i$ the event that Game $i$ outputs 1 and with $E_i := \Pr[1 \leftarrow G_i] - 1/2$ the advantage of $\mathcal{A}$ in Game $i$.

**Game 0.** This is the original $\mathsf{IND\text{-}ID\text{-}CPA}_{\mathcal{A}}^{\Pi}(\lambda)$ security experiment. By definition, we have

$$E_0 = \Pr[\mathsf{IND\text{-}ID\text{-}CPA}_{\mathcal{A}}\Pi(\lambda) = 1] - 1/2 = \varepsilon_{\mathcal{A}}.$$

**Game 1.** This game is identical to Game 0, except that the challenger runs $\bar{K} = (K_0, \ldots, K_\ell) \leftarrow \mathsf{BPSmp}(t_{\mathcal{A}}, \varepsilon_{\mathcal{A}})$ from Theorem 5.1. Furthermore, it defines $\mathcal{I} := \{i : K_i \neq \perp\}$. Let $\mathcal{Q}$ be the set of all queries that the adversary queries to $\mathsf{KeyGen}(\mathsf{mpk}, \mathsf{msk}, \cdot)$, and let $\mathcal{Q}^* := \mathcal{Q} \cup \{id^*\}$, where $id^*$ is the challenge query. Additionally, the challenger raises event $\mathsf{coll}$, aborts and outputs a random bit if there exist $id, id' \in \mathcal{Q}$ such that $id \neq id'$, but $H_i(id) = H_i(id')$ for all $i \in \mathcal{I}$. Since $\mathsf{coll}$ is defined exactly as in Lemma 5.1 we have

$$E_1 \geq E_0 - \Pr[\mathsf{coll}] = \varepsilon_{\mathcal{A}} - \Pr[\mathsf{coll}]$$

**Game 2.** In this game, the challenger raises event $\mathsf{badChal}$ which occurs if there exists an index $i \in \mathcal{I}$ such that $H_i(id^*) \neq K_i$ and it raises event $\mathsf{badEval}$ if there exists $id \in \mathcal{Q}$ such that $H_i(id) = K_i$ for all $i \in \mathcal{I}$. If either $\mathsf{badChal}$ or $\mathsf{badEval}$ occur it aborts and outputs a random bit. By Property 2 of Lemma 5.1 we have $\mathsf{badEval} \implies \mathsf{coll} \vee \mathsf{badChal}$ and by Property 1 we have

$$E_2 = E_1 \cdot \Pr[\neg\mathsf{badChal}] = (\varepsilon_{\mathcal{A}} - \Pr[\mathsf{coll}]) \cdot \Pr[\neg\mathsf{badChal}] \geq \frac{\varepsilon_{\mathcal{A}}^2}{32t_{\mathcal{A}}^2 - 16t_{\mathcal{A}}}$$

**Game 3.** In this game, we change the way $\mathsf{msk}$ and $\mathsf{mpk}$ are generated by the challenger. It samples $\alpha \leftarrow \mathbb{Z}_q^*$, $\tilde{x}_i \leftarrow \mathbb{Z}_q^*$ and sets for all $i = 0, \dots, \lfloor \log(n) \rfloor$

$$x_i := \begin{cases} \tilde{x}_i \cdot \alpha - K_i & \text{if } K_i \neq \perp \\ \tilde{x}_i & \text{otherwise.} \end{cases}$$

If $x_i = 0$ for any $0 \leq i \leq \lfloor \log(n) \rfloor$, then the challenger aborts and outputs a random bit. This happens *iff* $\tilde{x}_i\alpha = K_i$. Since $\tilde{x}_i\alpha$ is distributed uniformly at random in $\mathbb{Z}_q^*$, the probability that this happens for any of the $O(\log \lambda)$ many $x_i$ is negligible and we therefore have

$$E_3 = E_2 - \mathsf{negl}(\lambda).$$

**Game 4.** In this game, the way the challenger chooses $[1]_1$ is changed. Let $j = |\mathcal{I}|$ be the number of non-wildcard positions in $\bar{K}$. The challenger then first defines the polynomial $W(\mathsf{Z}) \in \mathbb{Z}_q[\mathsf{Z}]$ as

$$W(\mathsf{Z}) := \mathsf{Z}^{j-1} \prod_{\substack{i=0 \\ K_i \neq \perp}}^{\lfloor \log(n) \rfloor} \prod_{\substack{-2^{2^i}+1 \leq k \leq 2^{2^i}-1 \\ k \neq 0}} (\tilde{x}_i \mathsf{Z} + k)\,. \tag{5.8}$$

The challenger samples $[y]_1 \leftarrow \mathbb{G}_1$ and sets $[1]_1 := [y]_1^{W(\alpha)}$. If $[1]_1 = 1_{\mathbb{G}_1}$, which happens *iff* $W(\alpha) \equiv 0 \bmod p$, the challenger outputs a random bit and aborts. It can be seen that the distribution of $[1]$ changes only if $W(\alpha) \equiv 0 \bmod p$. By Lemma 5.2, we have $\mathsf{deg}(W) \leq \lfloor \log(n) \rfloor + 8 + 32t_{\mathcal{A}}^2/\varepsilon_{\mathcal{A}}$. Since $\alpha$ is uniformly random in $\mathbb{Z}_q^*$, we have by the Schwartz-Zippel Lemma $\Pr[W(\alpha) = 0] \leq \frac{\deg(W)}{q-1}$. Due to the fact that $\mathsf{deg}(W)$ is polynomial in $\lambda$ and $q \in 2^{O(\lambda)}$ by the properties of $\mathsf{GrpGen}$, we have

$$E_4 = E_3 - \mathsf{negl}(\lambda).$$

The previous steps now enable us to construct a an adversary $\mathcal{B}$ against the $\bar{q}$-DBDHI that perfectly simulates Game 4. $\mathcal{B}$ operates as follows.

**Initialization of $\mathcal{B}$.** $\mathcal{B}$ runs $\bar{K} = (K_0, \ldots, K_\ell) \leftarrow \mathsf{BPSmp}(t_\mathcal{A}, \varepsilon_\mathcal{A})$ at the beginning of the experiment and by doing so it also determines the index set $\mathcal{I} := \{i : K_i \neq \bot\}$ according to Lemma 5.1. Moreover, it receives a $\bar{q}$-DBDHI instance $(\mathcal{PG}, [y]_1, [y\alpha]_1, \ldots, [y\alpha^{\bar{q}}]_1, V)$, where $\bar{q} = 4\lambda + 6 + j + 2\sum_{\substack{i \in [\lfloor \log(n) \rfloor]_0 \\ K_i \neq \bot}} \left(2^{2^i} - 1\right)$, where $j = |\mathcal{I}|$ is the number of non-wildcard positions in $\bar{K}$, and where either $V = e([y]_1, [y]_1)^{1/\alpha}$ or $V \leftarrow \mathbb{G}_T$. In order to define $[1]_1$, $\mathcal{B}$ samples $\tilde{x}_i \leftarrow \mathbb{Z}_q^*$ for all $i = 0, \ldots, \lfloor \log(n) \rfloor$. Then it (implicitly) sets

$$x_{i'} := \begin{cases} \tilde{x}_{i'} \cdot Z - K_{i'} & \text{if } K_i \neq \bot \\ \tilde{x}_{i'} & \text{otherwise} \end{cases}.$$

It continues with compututing coefficients $\varphi_0, \ldots \varphi_{\bar{q}-4\lambda-8} \in \mathbb{Z}_q$ such that

$$W(\mathsf{Z}) = \mathsf{Z}^{j-1} \prod_{\substack{i=0 \\ K_i \neq \bot}}^{\lfloor \log(n) \rfloor} \prod_{\substack{-2^{2^i}+1 \leq k \leq 2^{2^i}-1 \\ k \neq 0}} (\tilde{x}_i \mathsf{Z} + k) = \sum_{k=0}^{\bar{q}-4\lambda-8} \varphi_k \mathsf{Z}^k.$$

$\mathcal{B}$ defines

$$[1]_1 := \prod_{k=0}^{\bar{q}-8\lambda-8} [y\alpha^k]_1^{\varphi_k} = [y \sum_{k=0}^{\bar{q}-4\lambda-8} \varphi_k \alpha^k]_1 = [yW(\alpha)]_1.$$

If $[1]_1 = 1_\mathbb{G}$, meaning $W(\alpha) \equiv 0 \bmod q$, $\mathcal{B}$ aborts and outputs a random bit. Afterwards $\mathcal{B}$ proceeds with computing $[F(\mathsf{msk}, m)]_1$ for $m = 0, ..., 2^{\ell+1} - 1$, where $\ell = \lfloor \log(n) \rfloor$ and $F(\mathsf{msk}, m) = \prod_{i'=0}^{\ell} x_{i'}^{b_{i'}(m)}$ as in Equation (5.5). That is, it computes $\psi_{m,0}, \ldots, \psi_{m,\bar{q}} \in \mathbb{Z}_q$ for all $m = 0, ..., 2^{\ell+1} - 1$ such that

$$W(Z) \cdot \prod_{i'=0}^{\ell} x_{i'}^{b_{i'}(m)} = \sum_{k=0}^{\bar{q}} \psi_{m,k} Z^k.$$

For all $m = 0, \ldots, 2^{\ell+1} - 1$ it sets

$$[F(\mathsf{msk}, m)]_1 := \prod_{k=0}^{\bar{q}} [y\alpha^k]_1^{\psi_{m,k}} = [yW(\alpha) \cdot \prod_{i'=1}^{\ell} x_{i'}^{b_{i'}(m)}]_1.$$

Finally $\mathcal{B}$ outputs $\mathsf{mpk} = (\mathcal{PG}, H, [F(\mathsf{msk}, 0)]_1, \ldots, [F(\mathsf{msk}, 2^{\ell+1} - 1]_1)$ to $\mathcal{A}$.

In order to respond to **KeyGen** queries and the challenge query for $id^*$, $\mathcal{B}$ defines a polynomial $P_{id}(\mathsf{Z}) \in \mathbb{Z}_q[\mathsf{Z}]$, which will assume the role of $u(id)$. Let $x_{i'} = \tilde{x}_{i'} \cdot Z - K_{i'}$ if $K_i \neq \bot$ and $x_{i'} = \tilde{x}_{i'}$ else. Then

$$P_{id}(\mathsf{Z}) := \prod_{i'=1}^{\lfloor \log(n) \rfloor} (x_{i'} + H_{i'}(id)).$$

Note that the values $x_{i'}$ may contain $\mathsf{Z}$. We require the following lemma by Yamada [Yam17] to proceed with the description of $\mathcal{B}$.

**Lemma 5.3** (Lemma 14 in [Yam17]). *Let $id \in \{0,1\}^*$ then there exist $\zeta_{id} \in \mathbb{Z}_q^*$ and $R_{id}(\mathsf{Z}) \in \mathbb{Z}_q[\mathsf{Z}]$ such that*

$$\frac{W(\mathsf{Z})}{P_{id}(\mathsf{Z})} = \begin{cases} \frac{\zeta_{id}}{\mathsf{Z}} + R_{id}(\mathsf{Z}) & \text{if } H_i(id) = K_i \text{ for all } i \in \mathcal{I} \\ R_{id}(\mathsf{Z}) & \text{else} \end{cases}.$$

**Answering key queries.** When $\mathcal{B}$ receives a key query for $id \in \{0,1\}^*$ from $\mathcal{A}$, it checks whether $H_i(id) = K_i$ for all $i \in \mathcal{I}$ and if so aborts and outputs a random bit. If there is an index $i \in \mathcal{I}$ such that $H_i(id) \neq K_i$ let $R_{id}(\mathsf{Z}) = \sum_{k=0}^{\bar{q}} \mathsf{Z}^k \rho_{id,k} \in \mathbb{Z}_q[\mathsf{Z}]$ such that $R_{id}(\mathsf{Z}) = W(\mathsf{Z})/P_{id}(\mathsf{Z})$, which is guaranteed to exist by Lemma 5.3. First $\mathcal{B}$ computes the coefficients $\rho_{id,k} \in \mathbb{Z}_q$ for all $k$. Then it computes and returns

$$[1/u(id)]_1 := \prod_{k=0}^{\bar{q}} [y\alpha^k]_1^{\rho_{id,k}} = [y \sum_{k=0}^{\bar{q}} \alpha^k \rho_{id,k}]_1 = [yR_{id}(\alpha)]_1 = [yW(\alpha)/P_{id}(\alpha)]_1.$$

**Answering challenge.** When $\mathcal{B}$ receives the challenge $id^* \in \{0,1\}^*$, it checks whether there exists $i \in \mathcal{I}$ such that $H_i(id) \neq K_i$. If this is true then it aborts and outputs a random bit. Else, let $\zeta := \zeta_{id^*} \in \mathbb{Z}_q$ and $R(\mathsf{Z}) := R_{id^*}(\mathsf{Z}) = \sum_{k=0}^{\bar{q}} \mathsf{Z}^k \gamma_k \in \mathbb{Z}_q[\mathsf{Z}]$ such that $W(\mathsf{Z})/P_{id^*}(\mathsf{Z}) = \zeta/\mathsf{Z} + R(\mathsf{Z})$ as guaranteed by Theorem 5.3. $\mathcal{B}$ then computes coefficients $\gamma_k \in \mathbb{Z}_q$ of $R$.
Using that $W(\mathsf{Z}) = \sum_{k=0}^{\bar{q}-4\lambda-8} \varphi_k \mathsf{Z}^k$ it also computes

$$\tilde{K} := V^{\zeta \cdot \varphi_0} e\left([y], \prod_{k=1}^{\bar{q}-4\lambda-8} [y\alpha^{k-1}]^{\zeta\varphi_k}\right) e\left(\prod_{k=0}^{\bar{q}} [y\alpha^k]^{\gamma_k}, \prod_{k=0}^{\bar{q}-4\lambda-8} [y\alpha^k]_1^{\varphi_k}\right) \tag{5.9}$$

Afterwards it samples $L \leftarrow \mathbb{Z}_q$ and computes and outputs

$$K := \tilde{K}^L \text{ and } C := [1]^L.$$

Finally, when $\mathcal{A}$ outputs its guess $b'$ to $\mathcal{B}$, then $\mathcal{B}$ also outputs $b'$ as solution to the $\bar{q}$-$\mathsf{DBDHI}$ instance.

**Analysis of $\mathcal{B}$.** Recapping the construction of $\mathcal{B}$, we observe that $\mathsf{mpk}$ and all answers of $\mathcal{B}$ are distributed identically to the challenger's interactions with $\mathcal{A}$ in Game 4. Analyzing $\mathcal{B}$'s response to the challenge $id^*$, we distinguish the following two cases depending on the $\bar{q}$-DBDHI instance.
Let $V = e([y]_1, [y]_1)^{1/\alpha}$. Then we have for the first part of (5.9)

$$V^{\zeta \cdot \varphi_0} e\left([y]_1, \prod_{k=1}^{\bar{q}-4\lambda-8} [y\alpha^{k-1}]_1^{\zeta\varphi_k}\right) = e([y]_1, [y]_1)^{\zeta \cdot \varphi_0/\alpha} e\left([y], \prod_{k=1}^{\bar{q}-4\lambda-7} [y\alpha^{k-1}]^{\zeta\varphi_k}\right)$$
$$= e\left([y]_1, [y \sum_{k=0}^{\bar{q}-4\lambda-8} \zeta\varphi_k \alpha^{k-1}]_1\right) = e\left([y]_1, [y]_1\right)^{\zeta W(\alpha)/\alpha}$$

and for the second part

$$e\left(\prod_{k=0}^{\bar{q}}[y\alpha^k]_1^{\gamma_k}, \prod_{k=0}^{\bar{q}-4\lambda-8}[y\alpha^k]_1^{\varphi_k}\right) = e\left([yR(\alpha)]_1, [yW(\alpha)]_1\right) = e\left([y]_1, [y]_1\right)^{R(\alpha)W(\alpha)}.$$

Together we have that

$$\tilde{K} = e\left([y]_1, [y]_1\right)^{\zeta \cdot W(\alpha)/\alpha + R(\alpha)\cdot W(\alpha)} = e\left([y]_1, [y]_1\right)^{W(\alpha)^2/P_{id^*}(\alpha)} = e\left([1]_1, [1]_1\right)^{1/P_{id^*}(\alpha)},$$

where we use $[yW(\alpha)]_1 = [1]_1$ and Lemma 5.3 multiplied by $W(\mathsf{Z})$ on both sides. Then if we implicitly set $s := L/P_{id^*}(\alpha)$ we have

$$K = \tilde{K}^L = e\left([1]_1, [1]_1\right)^{L/P_{id^*}(\alpha)} = e\left([1]_1, [1]_1\right)^s \text{ and } C = [1]_1^L = [1]_1^{P_{id^*}(\alpha)\cdot s}.$$

Since $P_{id^*}(\alpha) = u(id^*)$ we have that $C$ is a correct encapsulation of $K$.

In case that $V \leftarrow \mathbb{G}_T$ we have that $\tilde{K}^L$ is uniformly random in $\mathbb{G}_T$. Thus $K = \tilde{K}$ and $C$ do not match. Overall it can be seen that $\mathcal{B}$ simulates Game 4 perfectly. Plugging the game sequence and the reduction together gives us

$$\varepsilon_{\mathcal{B}} \geq \varepsilon_{\mathcal{A}}^2/(32t_{\mathcal{A}}^2 - 16t_{\mathcal{A}}) - \mathsf{negl}(\lambda).$$

**Running time of $\mathcal{B}$.** The running time $t_{\mathcal{B}}$ of $\mathcal{B}$ consists of the running time $t_{\mathcal{A}}$ of $\mathcal{A}$plus the time required to compute a valid public key $\mathsf{mpk}$ and the time to respond to oracle queries by $\mathcal{A}$. For computing $\mathsf{mpk}$ and these responses the most time consuming operations are exponentiation. Each computation needs at most $\bar{q}$ exponentiations. By Lemma 5.2 we have $\bar{q} \leq 4\lambda + 8 + \lfloor \log(n) \rfloor + 32t_{\mathcal{A}}^2/\varepsilon_{\mathcal{A}} = O(t_{\mathcal{A}}^2/\varepsilon_{\mathcal{A}})$ and thus

$$t_{\mathcal{B}} = t_{\mathcal{A}} + O(\bar{q}) = O(t_{\mathcal{A}}^2/\varepsilon_{\mathcal{A}}).$$

This completes the proof. □

## 5.4 A Digital Signature Scheme

In this section we describe a new digital signature scheme that is adaptively secure and where the signature consist of a *single* group element. Again, compared to the only other construction with those properties [JK18] the $\bar{q}$ of the required $\bar{q}$-type assumption is reduced *quadratically*, while the tightness of the reduction is improved *quadratically*, as well. Due to near-collision resistance, we are also able to reduce the output length of the hash function to roughly half of the output length required in [JK18], which reduces computational costs while guaranteeing the same level of security.

**The construction.** Let $\mathcal{H} = \{H : \{0,1\}^* \to \{0,1\}^{2\lambda+3}\}$ be a family of hash functions, let $\ell = \lfloor \log(2\lambda + 3) \rfloor$, and let $\mathcal{PG}$ be the description of a cryptographic Type-1 pairing group. We construct the digital signature scheme $\Sigma = (\mathbf{KeyGen}, \mathbf{Sign}, \mathbf{Verify})$ as follows.

- $\mathbf{KeyGen}(1^\lambda)$. Choose a random hash function $H \leftarrow \mathcal{H}$, a random generator $[1] \in \mathbb{G}_1$, and random elements $x_0, \ldots, x_\ell \in \mathbb{Z}_q^*$. Define the secret signing key *signk* as

$$signk = (x_0, \ldots, x_\ell) \in \mathbb{Z}_q^{\ell+1}.$$

For $i \in \mathbb{N}$ and $m \in \mathbb{N}_0$ define $b_i(m)$ as the function that, on input of integer $m$, outputs the $i$-th bit of the binary representation of $m$. For $signk = (x_0, \ldots, x_\ell)$ and $m = 0, \ldots, 2^{\ell+1} - 1$ define

$$F(signk, m) := \prod_{i=0}^{\ell} x_i^{b_i(m)}. \tag{5.10}$$

The verification key is defined as

$$vk = (\mathcal{PG}, H, [F(signk, 0)]_1, \ldots, [F(signk, 2^{\ell+1} - 1]_1).$$

- **Sign**$(signk, M)$. Let

$$u(M) = \prod_{i=0}^{\ell} (H_i(M) + x_i) \in \mathbb{Z}_q. \tag{5.11}$$

Then the signature for message $M$ is computed as

$$\sigma = [1/u(M)]_1.$$

- **Verify**$(vk, M, \sigma)$. Observe that

$$u(M) = \prod_{i=0}^{\ell} (H_i(M) + x_i) = d_0 + \sum_{m=1}^{2^\ell - 1} \left(d_m \prod_{i=0}^{\ell} x_i^{b_i(n)}\right),$$

where the constants $d_i$ are efficiently computable from $H(M)$. Using $H(M)$ and $\mathsf{vk}$ first $[u(M)]_1$ is computed as

$$[u(M)]_1 = \left[d_0 + \sum_{m=1}^{2^\ell - 1} \left(d_m \prod_{i=0}^{\ell} x_i^{b_i(n)}\right)\right]_1 = [d_0]_1 \cdot \prod_{m=1}^{2^\ell - 1} [F(\mathsf{msk}, m)]_1^{d_m}.$$

Note that this does not require knowledge of $x_0, \ldots, x_\ell$ explicitly.

Finally, the algorithm output 1 if

$$e([u(id)]_1, \sigma) = e([1]_1, [1]_1)$$

and else 0.

### 5.4.1 Proof of Correctness

Let $\sigma$ be a signature for message $M$ computed as above. Then we have

$$e([u(M)]_1, \sigma) = e([u(M)]_1, [1/(u(M))]_1) = e([1]_1, [1]_1).$$

### 5.4.2 Proof of Security

The security of this construction is based on the $\bar{q}$-BDHI assumption for Type-1 pairing groups [BB04a]. For consistency with the $\bar{q}$-dBDHI assumptuion, we re-randomize the group elements with an element $y \leftarrow \mathbb{Z}_q^*$. This has no impact on the hardness of the assumption because $[1]_1 \leftarrow \mathbb{G}_1$ is picked uniformly random anyway.

**Definition 5.3. $\bar{q}$-BDHI$_1$ Assumption.**
Let $\mathcal{PG} = (\mathbb{G}_1, \mathbb{G}_T, e, q)$ be the description of a cryptographic Type-1 pairing group and let $[1]_1$ be a random generator of $\mathbb{G}_1$. Let $\mathcal{A}$ be an adversary. We say that it $(t_\mathcal{A}, \varepsilon_\mathcal{A})$-breaks the $\bar{q}$-BDHI$_1$ assumption, if it runs in time $t_\mathcal{A}$ and

$$\left| \Pr \left[ \mathcal{A} \left( \mathcal{PG}, [y]_1, [y\alpha]_1, [y\alpha^2]_1, \ldots, [y\alpha^{\bar{q}}]_1 \right) = e([y]_1, [y]_1)^{1/\alpha} \right] \right| \geq \varepsilon_\mathcal{A}$$

where $y, \alpha \leftarrow \mathbb{Z}_q^*$.

Due to the fact that security of this signature scheme can be shown analogously to the one for IBKEM we leave the following theorem without a proof.

**Theorem 5.2.** *Let $\Sigma$ from Section 5.4 be instantiated with a family $\mathcal{H}$ of near-collision resistant hash functions in the sense of Definition 5.1. Let $\mathcal{A}$ be an adversary that $(t_\mathcal{A}, \varepsilon_\mathcal{A})$-breaks the EUF-CMA security of $\Sigma$. Given $\mathcal{A}$, we can build an adversary $\mathcal{B}$ that, given (sufficiently close approximations of) $t_\mathcal{A}$ and $\varepsilon_\mathcal{A}$, $(t_\mathcal{B}, \varepsilon_\mathcal{B})$-breaks the $\bar{q}$-BDHI$_1$ assumption with $\bar{q} \leq 4\lambda + 9 + \lfloor \log(2\lambda + 3) \rfloor + 32t_\mathcal{A}^2/\varepsilon_\mathcal{A}$ such that*

$$t_\mathcal{B} = O(32t_\mathcal{A}^2/\varepsilon_\mathcal{A}) \quad \textit{and} \quad \varepsilon_\mathcal{B} \geq \varepsilon_\mathcal{A}^2/(32t_\mathcal{A}^2 - 16t_\mathcal{A}) - \mathsf{negl}(\lambda),$$

*for some negligible term* $\mathsf{negl}$.

## 5.5 Discussions

**Comparison to confined guessing and Truncation Collision Resistance.** In order to generically construct adaptively secure cryptosystems from selectively secure ones, all techniques aim at reducing the size of an exponential-sized target space to a polynomial-sized one. Both, near-collision resistance as well as Truncation Collision Resistance [JK18], provide that on the one hand it is sufficiently easy to guess some amount of bits of a hash value, but on the other hand, collisions on this bits are sufficiently unlikely. This approach is similar in spirit to extremely lossy functions [Zha16]. In the case of near-collision resistance, the index set $\mathcal{I}$ as defined in Section 5.2.1, may contain *multiple* indices. This is a major difference of our approach to confined guessing and Truncation Collision Resistance, where always only *single* blocks are guessed, because it enables to define $n'$ in a much more fine-grained way, as *any* integer between 0 and $n$. In contrast, [BHJ+13, BHJ+15] and [JK18] were only able to pick values $n'$ of exponentially increasing size, such that $n' = 2^{2^j}$ for some $j$, which is the reason why our reductions can improve tightness and the strength of the required assumptions quadratically.

**Conclusion.** We introduced and used the concept of blockwise partitioning via near-collision resistance in order to construct efficient and adaptively-secure cryptosystems. This concept enables to reduce the target space, e.g. a identity or message space, to polynomial size and thus to improve the tightness of a cryptographic reduciton. We construct an adaptively-secure IBKEM, where ciphertexts consist of a single element, as well as adaptively-secure digital signature schemes, where signatures consist of a single element as well. The only schemes with those properties are due to Jager and Kurek [JK18]. Compared to [JK18], our schemes even more efficient. The reason is that we we improve the tightness of the cryptographic reductions quadratically and are able to use a cryptographic hash function with roughly half of the output size. Both leads to smaller cryptographic groups without loss of security and hence to more efficient schemes.

# BIBLIOGRAPHY

[ABD$^+$15] David Adrian, Karthikeyan Bhargavan, Zakir Durumeric, Pierrick Gaudry, Matthew Green, J. Alex Halderman, Nadia Heninger, Drew Springall, Emmanuel Thomé, Luke Valenta, Benjamin VanderSloot, Eric Wustrow, Santiago Zanella-Béguelin, and Paul Zimmermann. Imperfect forward secrecy: How Diffie-Hellman fails in practice. In *22nd ACM Conference on Computer and Communications Security*, October 2015.

[AHY15] Nuttapong Attrapadung, Goichiro Hanaoka, and Shota Yamada. A framework for identity-based encryption with almost tight security. In Tetsu Iwata and Jung Hee Cheon, editors, *ASIACRYPT 2015, Part I*, volume 9452 of *LNCS*, pages 521–549. Springer, Heidelberg, November / December 2015.

[Alp15] Jacob Alperin-Sheriff. Short signatures with short public keys from homomorphic trapdoor functions. In Jonathan Katz, editor, *PKC 2015*, volume 9020 of *LNCS*, pages 236–255. Springer, Heidelberg, March / April 2015.

[AMN01] Michel Abdalla, Sara K. Miner, and Chanathip Namprempre. Forward-secure threshold signature schemes. In *CT-RSA*, volume 2020 of *Lecture Notes in Computer Science*, pages 441–456. Springer, 2001.

[AR00] Michel Abdalla and Leonid Reyzin. A new forward-secure digital signature scheme. In Tatsuaki Okamoto, editor, *ASIACRYPT 2000*, volume 1976 of *LNCS*, pages 116–129. Springer, Heidelberg, December 2000.

[BB04a] Dan Boneh and Xavier Boyen. Efficient selective-ID secure identity based encryption without random oracles. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 223–238. Springer, Heidelberg, May 2004.

[BB04b] Dan Boneh and Xavier Boyen. Secure identity based encryption without random oracles. In Matthew Franklin, editor, *CRYPTO 2004*, volume 3152 of *LNCS*, pages 443–459. Springer, Heidelberg, August 2004.

[BB04c] Dan Boneh and Xavier Boyen. Short signatures without random oracles. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 56–73. Springer, Heidelberg, May 2004.

[BBG05] Dan Boneh, Xavier Boyen, and Eu-Jin Goh. Hierarchical identity based encryption with constant size ciphertext. In Ronald Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 440–456. Springer, Heidelberg, May 2005.

[BBH06] Dan Boneh, Xavier Boyen, and Shai Halevi. Chosen ciphertext secure public key threshold encryption without random oracles. In David Pointcheval, editor, *Topics in Cryptology – CT-RSA 2006*, pages 226–243, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[BCHK07] Dan Boneh, Ran Canetti, Shai Halevi, and Jonathan Katz. Chosen-ciphertext security from identity-based encryption. *SIAM Journal on Computing*, 36(5):1301–1328, 2007.

[Bel06] Mihir Bellare. New proofs for NMAC and HMAC: Security without collision-resistance. In Cynthia Dwork, editor, *CRYPTO 2006*, volume 4117 of *LNCS*, pages 602–619. Springer, Heidelberg, August 2006.

[BF01] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the Weil pairing. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 213–229. Springer, Heidelberg, August 2001.

[BFMLS05] K. Bentahar, P. Farshim, J. Malone-Lee, and N.P. Smart. Generic constructions of identity-based and certificateless kems. Cryptology ePrint Archive, Report 2005/058, 2005. `https://eprint.iacr.org/2005/058`.

[BFMS08] Kamel Bentahar, Pooya Farshim, John Malone-Lee, and Nigel P. Smart. Generic constructions of identity-based and certificateless KEMs. *Journal of Cryptology*, 21(2):178–199, April 2008.

[BG90] Mihir Bellare and Shafi Goldwasser. New paradigms for digital signatures and message authentication based on non-interative zero knowledge proofs. In Gilles Brassard, editor, *CRYPTO'89*, volume 435 of *LNCS*, pages 194–211. Springer, Heidelberg, August 1990.

[BH12] Itay Berman and Iftach Haitner. From non-adaptive to adaptive pseudorandom functions. In Ronald Cramer, editor, *TCC 2012*, volume 7194 of *LNCS*, pages 357–368. Springer, Heidelberg, March 2012.

[BHJ$^+$13] Florian Böhl, Dennis Hofheinz, Tibor Jager, Jessica Koch, Jae Hong Seo, and Christoph Striecks. Practical signatures from standard assumptions. In Thomas Johansson and Phong Q. Nguyen, editors, *EUROCRYPT 2013*, volume 7881 of *LNCS*, pages 461–485. Springer, Heidelberg, May 2013.

[BHJ$^+$15] Florian Böhl, Dennis Hofheinz, Tibor Jager, Jessica Koch, and Christoph Striecks. Confined guessing: New signatures from standard assumptions. *Journal of Cryptology*, 28(1):176–208, January 2015.

[BHKN13] Itay Berman, Iftach Haitner, Ilan Komargodski, and Moni Naor. Hardness preserving reductions via Cuckoo hashing. In Amit Sahai, editor, *TCC 2013*, volume 7785 of *LNCS*, pages 40–59. Springer, Heidelberg, March 2013.

[BJLS16] Christoph Bader, Tibor Jager, Yong Li, and Sven Schäge. On the impossibility of tight cryptographic reductions. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 273–304. Springer, Heidelberg, May 2016.

[Ble98] Daniel Bleichenbacher. Chosen ciphertext attacks against protocols based on the RSA encryption standard PKCS #1. In Hugo Krawczyk, editor, *CRYPTO'98*, volume 1462 of *LNCS*, pages 1–12. Springer, Heidelberg, August 1998.

[BLMR13] Dan Boneh, Kevin Lewi, Hart William Montgomery, and Ananth Raghunathan. Key homomorphic PRFs and their applications. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part I*, volume 8042 of *LNCS*, pages 410–428. Springer, Heidelberg, August 2013.

[BLS01] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the Weil pairing. In Colin Boyd, editor, *ASIACRYPT 2001*, volume 2248 of *LNCS*, pages 514–532. Springer, Heidelberg, December 2001.

[BLS04] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the Weil pairing. *Journal of Cryptology*, 17(4):297–319, September 2004.

[BM99] Mihir Bellare and Sara K. Miner. A forward-secure digital signature scheme. In Michael J. Wiener, editor, *CRYPTO'99*, volume 1666 of *LNCS*, pages 431–448. Springer, Heidelberg, August 1999.

[BMR10] Dan Boneh, Hart William Montgomery, and Ananth Raghunathan. Algebraic pseudorandom functions with improved efficiency from the augmented cascade. In Ehab Al-Shaer, Angelos D. Keromytis, and Vitaly Shmatikov, editors, *ACM CCS 2010*, pages 131–140. ACM Press, October 2010.

[BOGW88] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In *STOC*, 1988.

[BOY86] C. BOYD. Digital multisignatures. *Cryptography and Coding*, pages 241–246, 1986.

[BP14] Abhishek Banerjee and Chris Peikert. New and improved key-homomorphic pseudorandom functions. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 353–370. Springer, Heidelberg, August 2014.

[BPR12] Abhishek Banerjee, Chris Peikert, and Alon Rosen. Pseudorandom functions and lattices. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 719–737. Springer, Heidelberg, April 2012.

[BR93a] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In Dorothy E. Denning, Raymond Pyle, Ravi Ganesan, Ravi S. Sandhu, and Victoria Ashby, editors, *ACM CCS 93*, pages 62–73. ACM Press, November 1993.

[BR93b] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *Proceedings of the 1st ACM Conference on Computer and Communications Security*, CCS '93, pages 62–73, New York, NY, USA, 1993. ACM.

[BR95] Mihir Bellare and Phillip Rogaway. Optimal asymmetric encryption. In Alfredo De Santis, editor, *EUROCRYPT'94*, volume 950 of *LNCS*, pages 92–111. Springer, Heidelberg, May 1995.

[CFN15] Dario Catalano, Dario Fiore, and Luca Nizzardo. Programmable hash functions go private: Constructions and applications to (homomorphic) signatures with shorter public keys. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part II*, volume 9216 of *LNCS*, pages 254–274. Springer, Heidelberg, August 2015.

[CG14] Nishanth Chandran and Sanjam Garg. Balancing output length and query bound in hardness preserving constructions of pseudorandom functions. In Willi Meier and Debdeep Mukhopadhyay, editors, *INDOCRYPT 2014*, volume 8885 of *LNCS*, pages 89–103. Springer, Heidelberg, December 2014.

[CGH98] Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited (preliminary version). In *30th ACM STOC*, pages 209–218. ACM Press, May 1998.

[CGHY08] Sherman S. M. Chow, H. W. Go, Lucas Chi Kwong Hui, and Siu-Ming Yiu. Multiplicative forward-secure threshold signature scheme. *I. J. Network Security*, 7:397–403, 2008.

[CHK03a] Ran Canetti, Shai Halevi, and Jonathan Katz. A forward-secure public-key encryption scheme. In Eli Biham, editor, *EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 255–271. Springer, Heidelberg, May 2003.

[CHK03b] Ran Canetti, Shai Halevi, and Jonathan Katz. A forward-secure public-key encryption scheme. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 255–271. Springer, 2003.

[CHM10] Sanjit Chatterjee, Darrel Hankerson, and Alfred Menezes. On the efficiency and security of pairing-based protocols in the type 1 and type 4 settings. volume 2010, page 388, 08 2010.

[CLL$^+$13] Jie Chen, Hoon Wei Lim, San Ling, Huaxiong Wang, and Hoeteck Wee. Shorter ibe and signatures via asymmetric pairings. In Michel Abdalla and Tanja Lange, editors, *Pairing-Based Cryptography – Pairing 2012*, pages 122–140, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[Cor02] Jean-Sébastien Coron. Optimal security proofs for PSS and other signature schemes. In Lars R. Knudsen, editor, *EUROCRYPT 2002*, volume 2332 of *LNCS*, pages 272–287. Springer, Heidelberg, April / May 2002.

[CW79] Larry Carter and Mark N. Wegman. Universal classes of hash functions. *J. Comput. Syst. Sci.*, 18(2):143–154, 1979.

[CW13] Jie Chen and Hoeteck Wee. Fully, (almost) tightly secure IBE and dual system groups. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 435–460. Springer, Heidelberg, August 2013.

[DF90] Yvo Desmedt and Yair Frankel. Threshold cryptosystems. In Gilles Brassard, editor, *CRYPTO'89*, volume 435 of *LNCS*, pages 307–315. Springer, Heidelberg, August 1990.

[DH76] W. Diffie and M. Hellman. New directions in cryptography. *IEEE Trans. Inf. Theory*, 22:644–654, 1976.

[DHKP97] Martin Dietzfelbinger, Torben Hagerup, Jyrki Katajainen, and Martti Penttonen. A reliable randomized algorithm for the closest-pair problem. *Journal of Algorithms*, 25(1):19–51, 1997.

[DM14] Léo Ducas and Daniele Micciancio. Improved short lattice signatures in the standard model. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 335–352. Springer, Heidelberg, August 2014.

[DN19] Manu Drijvers and Gregory Neven. Forward-secure multi-signatures. Cryptology ePrint Archive, Report 2019/261, 2019. `http://eprint.iacr.org/2019/261`.

[DS15] Nico Döttling and Dominique Schröder. Efficient pseudorandom functions via on-the-fly adaptation. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part I*, volume 9215 of *LNCS*, pages 329–350. Springer, Heidelberg, August 2015.

[EHK$^+$13a] Alex Escala, Gottfried Herold, Eike Kiltz, Carla Rafols, and Jorge Villar. An algebraic framework for diffie-hellman assumptions. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology – CRYPTO 2013*, pages 129–147, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[EHK$^+$13b] Alex Escala, Gottfried Herold, Eike Kiltz, Carla Ràfols, and Jorge Villar. An algebraic framework for Diffie-Hellman assumptions. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 129–147. Springer, Heidelberg, August 2013.

[FF13] Marc Fischlin and Nils Fleischhacker. Limitations of the meta-reduction technique: The case of Schnorr signatures. In Thomas Johansson and Phong Q. Nguyen, editors, *EUROCRYPT 2013*, volume 7881 of *LNCS*, pages 444–460. Springer, Heidelberg, May 2013.

[FHPS13] Eduarda S. V. Freire, Dennis Hofheinz, Kenneth G. Paterson, and Christoph Striecks. Programmable hash functions in the multilinear setting. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part I*, volume 8042 of *LNCS*, pages 513–530. Springer, Heidelberg, August 2013.

[FLR$^+$10] Marc Fischlin, Anja Lehmann, Thomas Ristenpart, Thomas Shrimpton, Martijn Stam, and Stefano Tessaro. Random oracles with(out) programmability. In Masayuki Abe, editor, *ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 303–320. Springer, Heidelberg, December 2010.

[FOPS01] Eiichiro Fujisaki, Tatsuaki Okamoto, David Pointcheval, and Jacques Stern. Rsa-oaep is secure under the rsa assumption. In Joe Kilian, editor, *Advances in Cryptology — CRYPTO 2001*, pages 260–274, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.

[Fre10] David Mandell Freeman. Converting pairing-based cryptosystems from composite-order groups to prime-order groups. In Henri Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 44–61. Springer, Heidelberg, May / June 2010.

[GGM84] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. On the cryptographic applications of random functions. In G. R. Blakley and David Chaum, editors, *CRYPTO'84*, volume 196 of *LNCS*, pages 276–288. Springer, Heidelberg, August 1984.

[GGM86] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *Journal of the ACM*, 33(4):792–807, October 1986.

[GHKW16] Romain Gay, Dennis Hofheinz, Eike Kiltz, and Hoeteck Wee. Tightly CCA-secure encryption without pairings. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part I*, volume 9665 of *LNCS*, pages 1–27. Springer, Heidelberg, May 2016.

[GJKR07] Rosario Gennaro, Stanislaw Jarecki, Hugo Krawczyk, and Tal Rabin. Secure distributed key generation for discrete-log based cryptosystems. *Journal of Cryptology*, 20:51–83, 05 2007.

[GM84] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270 – 299, 1984.

[GMR88] Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM J. Comput.*, 17(2):281–308, April 1988.

[Gol01] Oded Goldreich. *Foundations of Cryptography: Basic Tools*, volume 1. Cambridge University Press, Cambridge, UK, 2001.

[GQ90] Louis C. Guillou and Jean-Jacques Quisquater. A "paradoxical" indentity-based signature scheme resulting from zero-knowledge. In Shafi Goldwasser, editor, *CRYPTO'88*, volume 403 of *LNCS*, pages 216–231. Springer, Heidelberg, August 1990.

[HJJ$^+$97] Amir Herzberg, Markus Jakobsson, Staniss Jarecki, Hugo Krawczyk, and Moti Yung. Proactive public key and signature systems. *Proceedings of the ACM Conference on Computer and Communications Security*, 01 1997.

[HJK11] Dennis Hofheinz, Tibor Jager, and Eike Kiltz. Short signatures from weaker assumptions. In Dong Hoon Lee and Xiaoyun Wang, editors, *ASIACRYPT 2011*, volume 7073 of *LNCS*, pages 647–666. Springer, Heidelberg, December 2011.

[HJK12] Dennis Hofheinz, Tibor Jager, and Edward Knapp. Waters signatures with optimal security reduction. In Marc Fischlin, Johannes Buchmann, and Mark Manulis, editors, *PKC 2012*, volume 7293 of *LNCS*, pages 66–83. Springer, Heidelberg, May 2012.

[HJKY95] Amir Herzberg, Stanislaw Jarecki, Hugo Krawczyk, and Moti Yung. Proactive secret sharing or: How to cope with perpetual leakage. In Don Coppersmith, editor, *CRYPTO'95*, volume 963 of *LNCS*, pages 339–352. Springer, Heidelberg, August 1995.

[HK08] Dennis Hofheinz and Eike Kiltz. Programmable hash functions and their applications. In David Wagner, editor, *CRYPTO 2008*, volume 5157 of *LNCS*, pages 21–38. Springer, Heidelberg, August 2008.

[HMS12] Goichiro Hanaoka, Takahiro Matsuda, and Jacob C. N. Schuldt. On the impossibility of constructing efficient key encapsulation and programmable hash functions in prime order groups. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 812–831. Springer, Heidelberg, August 2012.

[IKOS08] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Cryptography with constant computational overhead. In Richard E. Ladner and Cynthia Dwork, editors, *40th ACM STOC*, pages 433–442. ACM Press, May 2008.

[JK18] Tibor Jager and Rafael Kurek. Short digital signatures and ID-KEMs via truncation collision resistance. In Thomas Peyrin and Steven Galbraith, editors, *ASIACRYPT 2018, Part II*, volume 11273 of *LNCS*, pages 221–250. Springer, Heidelberg, December 2018.

[JKP18] Tibor Jager, Rafael Kurek, and Jiaxin Pan. Simple and more efficient prfs with tight security from lwe and matrix-ddh. In Thomas Peyrin and Steven Galbraith, editors, *Advances in Cryptology – ASIACRYPT 2018*, pages 490–518, Cham, 2018. Springer International Publishing.

[JN19] Tibor Jager and David Niehues. On the real-world instantiability of admissible hash functions and efficient verifiable random functions. In Kenneth G. Paterson and Douglas Stebila, editors, *SAC 2019*, volume 11959 of *LNCS*, pages 303–332. Springer, Heidelberg, August 2019.

[JPT12] Abhishek Jain, Krzysztof Pietrzak, and Aris Tentes. Hardness preserving constructions of pseudorandom functions. In Ronald Cramer, editor, *TCC 2012*, volume 7194 of *LNCS*, pages 369–382. Springer, Heidelberg, March 2012.

[KK12] Saqib A. Kakvi and Eike Kiltz. Optimal security proofs for full domain hash, revisited. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 537–553. Springer, Heidelberg, April 2012.

[KL14] Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography, Second Edition.* CRC Press, 2014.

[KOS10] Eike Kiltz, Adam O'Neill, and Adam Smith. Instantiability of rsa-oaep under chosen-plaintext attack. In Tal Rabin, editor, *Advances in Cryptology*

– *CRYPTO 2010*, pages 295–313, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[Kra00] Hugo Krawczyk. Simple forward-secure signatures from any signature scheme. In *Proceedings of the 7th ACM Conference on Computer and Communications Security*, CCS '00, pages 108–115, New York, NY, USA, 2000. ACM.

[Kra10] Hugo Krawczyk. Cryptographic extraction and key derivation: The HKDF scheme. In Tal Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 631–648. Springer, Heidelberg, August 2010.

[Kur20a] Rafael Kurek. Efficient forward-secure threshold public key encryption. In Joseph K. Liu and Hui Cui, editors, *Information Security and Privacy, The 25th Australasian Conference on Information Security and Privacy, ACISP2020.* Springer Nature, 2020.

[Kur20b] Rafael Kurek. Efficient forward-secure threshold signatures. In Aoki Kazumaro and Akira Kanaoka, editors, *Advances in Information and Computer Security,15th International Workshop on Security, IWSEC 2020.* Springer Nature, 2020.

[LCT03] Li-Shan Liu, Cheng-Kang Chu, and Wen-Guey Tzeng. A threshold GQ signature scheme. In Jianying Zhou, Moti Yung, and Yongfei Han, editors, *ACNS 03*, volume 2846 of *LNCS*, pages 137–150. Springer, Heidelberg, October 2003.

[Lev87] Leonid A. Levin. One way functions and pseudorandom generators. *Combinatorica*, 7(4):357–363, Dec 1987.

[Lew12] Allison B. Lewko. Tools for simulating features of composite order bilinear groups in the prime order setting. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 318–335. Springer, Heidelberg, April 2012.

[LJY16] Benoît Libert, Marc Joye, and Moti Yung. Born and raised distributively: Fully distributed non-interactive adaptively-secure threshold signatures with short shares. *Theor. Comput. Sci.*, 645:1–24, 2016.

[LOS$^+$10] Allison B. Lewko, Tatsuaki Okamoto, Amit Sahai, Katsuyuki Takashima, and Brent Waters. Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption. In Henri Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 62–91. Springer, Heidelberg, May / June 2010.

[LW09] Allison B. Lewko and Brent Waters. Efficient pseudorandom functions from the decisional linear assumption and weaker variants. In Ehab Al-Shaer, Somesh Jha, and Angelos D. Keromytis, editors, *ACM CCS 2009*, pages 112–120. ACM Press, November 2009.

[LW10] Allison B. Lewko and Brent Waters. New techniques for dual system encryption and fully secure HIBE with short ciphertexts. In Daniele Micciancio, editor, *TCC 2010*, volume 5978 of *LNCS*, pages 455–479. Springer, Heidelberg, February 2010.

[LW14] Allison B. Lewko and Brent Waters. Why proving HIBE systems secure is difficult. In Phong Q. Nguyen and Elisabeth Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 58–76. Springer, Heidelberg, May 2014.

[LY13a] Benoît Libert and Moti Yung. Adaptively secure non-interactive threshold cryptosystems. *Theoretical Computer Science*, 478:76 – 100, 2013.

[LY13b] Benoît Libert and Moti Yung. Adaptively secure non-interactive threshold cryptosystems. *Theoretical Computer Science*, 478:76 – 100, 2013.

[Nie02] Jesper Buus Nielsen. Separating random oracle proofs from complexity theoretic proofs: The non-committing encryption case. In Moti Yung, editor, *CRYPTO 2002*, volume 2442 of *LNCS*, pages 111–126. Springer, Heidelberg, August 2002.

[NR97] Moni Naor and Omer Reingold. Number-theoretic constructions of efficient pseudo-random functions. In *38th FOCS*, pages 458–467. IEEE Computer Society Press, October 1997.

[OY91] Rafail Ostrovsky and Moti Yung. How to withstand mobile virus attacks (extended abstract). In *Proceedings of the Tenth Annual ACM Symposium on Principles of Distributed Computing*, PODC '91, pages 51–59, New York, NY, USA, 1991. ACM.

[Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Harold N. Gabow and Ronald Fagin, editors, *37th ACM STOC*, pages 84–93. ACM Press, May 2005.

[Sha79] Adi Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, November 1979.

[Sha84] Adi Shamir. Identity-based cryptosystems and signature schemes. In G. R. Blakley and David Chaum, editors, *CRYPTO'84*, volume 196 of *LNCS*, pages 47–53. Springer, Heidelberg, August 1984.

[Sho00] Victor Shoup. Practical threshold signatures. In Bart Preneel, editor, *EUROCRYPT 2000*, volume 1807 of *LNCS*, pages 207–220. Springer, Heidelberg, May 2000.

[Sho01] Victor Shoup. Oaep reconsidered. In *Proceedings of the 21st Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '01, page 239–259, Berlin, Heidelberg, 2001. Springer-Verlag.

[TT01] Wen-Guey Tzeng and Zhi-Jia Tzeng. Robust forward-secure signature schemes with proactive security. In Kwangjo Kim, editor, *PKC 2001*, volume 1992 of *LNCS*, pages 264–276. Springer, Heidelberg, February 2001.

[Wat05] Brent R. Waters. Efficient identity-based encryption without random oracles. In Ronald Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 114–127. Springer, Heidelberg, May 2005.

[Wat09] Brent Waters. Dual system encryption: Realizing fully secure IBE and HIBE under simple assumptions. In Shai Halevi, editor, *CRYPTO 2009*, volume 5677 of *LNCS*, pages 619–636. Springer, Heidelberg, August 2009.

[WQFX06] Hong Wang, Gang Qiu, Dengguo Feng, and Guo-Zhen Xiao. Cryptanalysis of tzeng-tzeng forward-secure signature schemes. *IEICE Transactions*, 89-A:822–825, 2006.

[Yam17] Shota Yamada. Asymptotically compact adaptively secure lattice IBEs and verifiable random functions via generalized partitioning techniques. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part III*, volume 10403 of *LNCS*, pages 161–193. Springer, Heidelberg, August 2017.

[YK07] Jia Yu and Fanyu Kong. Forward secure threshold signature scheme from bilinear pairings. In Yuping Wang, Yiu-ming Cheung, and Hailin Liu, editors, *Computational Intelligence and Security*, pages 587–597, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[Zha16] Mark Zhandry. The magic of ELFs. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part I*, volume 9814 of *LNCS*, pages 479–508. Springer, Heidelberg, August 2016.

[ZXZ13] Xiujie Zhang, Chunxiang Xu, and Wenzheng Zhang. Efficient chosen ciphertext secure threshold public-key encryption with forward security. In *Proceedings of the 2013 Fourth International Conference on Emerging Intelligent Data and Web Technologies*, EIDWT '13, page 407–413, USA, 2013. IEEE Computer Society.