Updatable Signatures and Message Authentication Codes[⋆]

Valerio Cini[1], Sebastian Ramacher[1], Daniel Slamanig[1], Christoph Striecks[1], and Erkan Tairi[2]

[1] AIT Austrian Institute of Technology, Vienna, Austria
{firstname.lastname}@ait.ac.at
[2] TU Wien, Vienna, Austria
erkan.tairi@tuwien.ac.at

Abstract. Cryptographic objects with updating capabilities have been proposed by Bellare, Goldreich and Goldwasser (CRYPTO'94) under the umbrella of incremental cryptography. They have recently seen increased interest, motivated by theoretical questions (Ananth et al., EC'17) as well as concrete practical motivations (Lehmann et al., EC'18; Groth et al. CRYPTO'18; Klooß et al., EC'19). In this work, the form of updatability we are particularly interested in is that primitives are key-updatable *and* allow to update "old" cryptographic objects, e.g., signatures or message authentication codes, from the "old" key to the updated key at the same time without requiring full access to the new key (i.e., only via a so-called update token).

Inspired by the rigorous study of updatable encryption by Lehmann and Tackmann (EC'18) and Boyd et al. (CRYPTO'20), we introduce a definitional framework for updatable signatures (USs) and message authentication codes (UMACs). We discuss several applications demonstrating that such primitives can be useful in practical applications, especially around key rotation in various domains, as well as serve as building blocks in other cryptographic schemes. We then turn to constructions and our focus there is on ones that are secure and practically efficient. In particular, we provide generic constructions from key-homomorphic primitives (signatures and PRFs) as well as direct constructions. This allows us to instantiate these primitives from various assumptions such as DDH or CDH (latter in bilinear groups), or the (R)LWE and the SIS assumptions. As an example, we obtain highly practical US schemes from BLS signatures or UMAC schemes from the Naor-Pinkas-Reingold PRF.

1 Introduction

Updatable cryptographic primitives, initially introduced as incremental cryptography [BGG94, BGG95], support the transformation of one cryptographic object

[⋆] This is the full version of a paper which appears in the proceedings of the 24th International Conference on Practice and Theory of Public-Key Cryptography - PKC 2021, LNCS, Springer.

to a related one without recomputing it entirely and have been widely studied (cf. [ABK20] for an overview). Recently, Ananth et al. in [ACJ17] studied a unified approach towards adding updatability features to many cryptographic primitives such as attribute-based encryption, functional encryption or more generally cryptographic circuit compilers. Moreover, they study the updatability for classical protocols such as zero-knowledge proofs and secure multi-party computation. Their constructions thereby rely on a novel updatable version of randomized encodings [IK00, AIK06]. Besides exploring such updatable primitives from a rather theoretical perspective, there have been various interesting lines of work on specific updatable primitives inspired by concrete practical applications. For instance, Groth et al. in [GKM+18] introduce the notion of an updatable common reference string (CRS) and apply it to zk-SNARKs (used within many real world protocols in the cryptocurrency and distributed ledger domain) to reduce the trust in the generator of the CRS and cope with malicious CRS generators. Later in [Lip20] Lipmaa studied quasi-adaptive NIZK (QA-NIZK) proofs in an updatable CRS setting where in addition to CRS updates "old" valid proofs can be updated to be still valid under an updated CRS. Another such primitive that is strongly motivated by practical applications and has recently been studied quite intensively is updatable encryption (UE) [BLMR13, EPRS17, LT18, KLR19, JKR19, BDGJ20, BEKS20]. An UE scheme is a symmetric encryption scheme that allows the key holder to update keys and to compute an update token, which can be given to a party storing ciphertexts, and can be used to update existing ciphertexts to ones under the new key. UE is motivated by the fact that it is a good key management practice to change encryption keys periodically and it avoids the cumbersome requirement to download, decrypt and re-encrypt and upload all the data again.

Motivation. Our work is now essentially motivated by this previous work on UE and the observation that it is equally important in context of signatures and message authentication codes (MACs) to follow good key management practices and to periodically switch keys. For instance, any kind of software distribution channels including App stores or operating system updates rely on signatures to ensure the authenticity of the software they distribute. Moreover, file systems or (outsourced) databases usually require signatures or MACs to ensure integrity of stored data (we discuss this in more detail later in this section). What we envision therefore are signatures and MACs that are updatable in a sense that, similar to UE, holders of a secret key can compute a token that allows some third party to update existing signatures and MACs to ones valid under the new key. Thereby, we want to guarantee unforgeability even if the adversary can see lots of different keys and tokens with the restriction that we exclude trivial forgeries (we discuss this possible leakage in more detail later).

Related and previous work. While there are notions of signatures that support updating keys or even guarantee unforgeability when allowing queries under (adversarially) updated keys, none of them rigorously covers what we have sketched above and in particular guaranteeing security even if signatures can be updated between different keys. Closest are updatable signatures by

Klooß et al. [KLR19] implicitly used in one of their UE constructions. But they do not treat their security in the updatable setting and rather sketch how they can be obtained from unforgeable signatures in combination with generic properties of the token generation. Key-updatable signatures by Jaeger and Stepanovs [JS18] or key-updating signatures by Jost et al. [JMM19] proposed in context of secure messaging allow to update keys and obtain signatures under updated keys, but do not consider signature updates. Similarly, signatures with re-randomizable keys by Fleischhacker et al. [FKM+16] consider adversarially chosen updates of the secret key (and access to a signing oracle under updated keys), but do not consider updating existing signatures. Somewhat orthogonal, key-homomorphic signatures by Derler and Slamanig [DS19] consider updating keys as well as updating existing signatures (this concept is similar to key-homomorphic PRFs [BLMR13]). But they only study the required properties functional-wise and do not consider an unforgeability notion (rather they implicitly prove them for their respective applications). Nevertheless, as we will see these key-homomorphic signatures and key-homomorphic PRFs [BLMR13, BP14, Kim20] can be used as the basis for some constructions of US and UMACs, respectively. Finally, there is a recent notion of updatable signatures by Abdolmaleki et al. [ARS20], which however focuses on key-update tokens that serve as a proof of correct update and allow extractability of update keys in order to be used within zk-SNARKs with updatable CRS.

Our framework for updatable signatures and MACs. Since none of the existing works cover updatable signatures with strong security guarantees (and there is to the best of our knowledge no work related to updatable MACs), our goal is to design a comprehensive framework and security model. Therefore, similar to models for UE [LT18, BDGJ20], we use the concept of epochs, where each epoch e has an associated key-pair (sk_e, pk_e) of a signature scheme starting with an initial key-pair in epoch 1 (all the discussion below analogously applies to UMACs). An US scheme then provides the functionality that in epoch e given (sk_e, pk_e) we can compute a key-pair (sk_{e+1}, pk_{e+1}) for the next epoch together with an update token Δ_{e+1} that is capable of updating signatures under a key from epoch e to $e+1$. Our focus is on schemes where these update tokens are independent of the signature and so Δ_{e+1} can be used for any signature from epoch e. We want that the schemes support an arbitrary number of epochs (any polynomial in the security parameters) but to support schemes from lattice assumptions we also consider a bounded number of epochs (bounded US) with a concrete bound T that usually depends on some parameters of the scheme. The goal is now to achieve strong security guarantees, in particular, the US scheme stays secure even after signing keys are compromised (a feature which is called post-compromise secrecy) and also before signing keys get corrupted (a feature called forward secrecy). Furthermore, outside of our model, we consider as an additional practical feature of US and UMAC the so called message-independence. This means that the update functionality only requires the update token and a signature, but does not need to access the respective message.

For unforgeability, we allow the adversary to trigger arbitrary signature computations, computations of next keys and updates adaptively and also adaptively compromise tokens and signing keys. We thereby use the concept of leakage profiles originally defined in [LT18] and also used in [BDGJ20] to capture key, token, and signature "leakage" that cannot be captured by the oracles in the security experiment. The reason is that due to the nature of updates, US schemes inherently allow for information leakage of updated message-signature pairs, keys, and tokens besides what is modeled in the security experiment. For instance, if the adversary compromises a secret key sk_e and a token Δ_{e+1} it might be possible to derive sk_{e+1} (or sk_e from key sk_{e+1} and token Δ_{e+1}). Also, a token Δ_e besides allowing to update signatures into the next epoch may also allow to switch signatures back to previous epochs. However, we stress that in contrast to UE, where no-directional UE schemes are highly desirable, for US it does not seem to be that useful. The reason is that upgrading old signatures is covered by correctness (and thus cannot be prevented) and preventing switching keys or signatures back to previous epochs is only required if old public keys are not considered revoked (and we currently do not see such applications).

In addition to the unforgeability notion, we also provide an unlinkability notion that essentially says that updated signatures cannot be distinguished from fresh signatures. More precisely, we require that an adversary even when given all signing keys, tokens as well as signatures is not able to distinguish a fresh signature from an updated version of the signature that it already holds. While this property does not seem essential to the practical applications discussed below, we discuss cryptographic applications where this notion is important.

Exploring applications. We will now discuss practical as well as cryptographic applications of US and UMACs.

Key rotation in software distributions with US. Software distribution channels including App stores such as Google Play [Goo19, Ele14, WLX+19], Apple's App Store [App20], or Microsoft's Windows Apps [Mic20] and Windows Updates, or Linux distributions such as Debian [Kra05], Ubuntu, Red Hat [Red18] and Arch Linux [Arc20] rely on signatures to ensure the authenticity of their software packages. In one way or another, they either sign indices including hashes of the software packages or sign the software packages directly. For the latter, packages are often signed by individual developers whose keys are either signed by some central party like the app store provider or are shipped to the user directly via keyring packages containing all trusted keys. In this setting, key rotation of individual developer keys becomes an issue, since, if keys are rotated, all software packages signed by the old key have to be re-signed with the new one. The same issue can also be observed in the context of signed boot loaders and kernels for secure boot [LSW10]. When relying on US, key rotation of developer keys becomes less of a burden on the developer. Indeed, the developer would update the key and produce an update token which is then used to update all signatures from this developer. Thus, instead of the developer having to re-sign all their packages, the signature adaption can be outsourced to a service run by the app store. Note that the effort for certifying new or updated keys would be the same in both settings.

File system and (outsourced) database integrity with UMACs. Modern file systems including zfs [ZRAA10] ensure the on-disk data integrity by storing hashes of the data. Additionally, when replicating data from one storage pool to another, the digests ensure integrity during transport. Similarly, databases support integrity checks which are helpful for replication and backups. Especially interesting is the application to outsourced databases [MNT06, WG18], where in case of key-rotation the use of an UMAC enables these updates to be performed without re-computing all the authentication tags from scratch and without giving the actual key to the third party hosting the database.

Malleable signatures and revocation in privacy protocols. Redactable signatures (RS) [SBZ02, JMSW02] and sanitizable signatures (SS) [ACdMT05, BFF$^+$09] are malleable signatures that allow to remove parts from signed messages or replace designated parts of signed messages by designated parties without invalidating the signatures. They have numerous applications, but due to their selective disclosure functionality are especially attractive to protect privacy in medical documents when shared with other parties. Replacing the conventional EUF-CMA secure signature scheme that typically serves as a building block in such schemes with an US can, similar to the above applications, help to reduce re-signing effort in case of a key-rotation. This is particularly interesting when large amounts of signed medical documents are involved. US providing unlinkability can firstly help if one requires unlinkability from the respective RS or SS scheme (cf. [BFLS10] for a discussion of the linkability problem when joining different versions of a document to derive additional information) even over different versions of one document under different (updated) keys. Secondly, unlinkable RS [CDHK15, San20] serve as a building block to construct anonymous credentials (ACs). In this context, unlinkable US allow to realize credential revocation in the following way: the issuer can provide an update token to a service that receives signatures from users and only updates credentials of non-revoked users to the current issuer key. Unlinkability of the US thereby in particular guarantees that it is hard to distinguish between credentials of non-revoked users from that of newly joined users. The same revocation idea can also be applied to replace re-issuing based revocation [BCC$^+$16] in group signatures that follow the sign-randomize-prove paradigm [BCN$^+$10, PS16, DS18]. Moreover, UMACs seem to be suitable for the same purpose in keyed-verification ACs [CMZ14], i.e., ACs where issuer and verifier are the same entity, typically constructed from algebraic MACs instead of signatures. An in-depth study of these cryptographic applications of US and UMACs is considered as future work.

CCA-secure UE with ciphertext integrity using UMACs. Klooß et al. [KLR19] showed how to achieve CCA security and ciphertext integrity for UE using the Encrypt-and-MAC transformation. Their transformation requires encryption and MAC schemes that support key-rotation. In [KLR19] the key-rotatable MAC was instantiated using the DDH-based PRF (NPR) [NPR99] using a key-switching akin to the proxy re-encryption approach due to Blaze et al. [BBS98]. We note that a UMAC satisfies the key-rotatable property, and hence, can be directly plugged into the described transformation to obtain CCA-secure UE with ciphertext integrity by using a suitable encryption scheme and any UMAC.

Further contributions. Besides the already discussed comprehensive framework for US and UMACs, we provide the following contributions:
US and UMAC from KH primitives. We construct US from key-homomorphic (KH) signatures [DS19], which satisfy some additional requirement that all natural schemes provide. Due to the properties of KH signatures, they are unlinkable. With respect to provable security, we use a proof-technique that is inspired by the key-insulation technique due to Klooß et al. [KLR19], where we essentially can use the unlinkability property of the underlying KH signature. Similarly, we obtain fixed-length UMACs from key-homomorphic PRFs [BLMR13], which we generically turn into variable-input UMACs and our proof technique essentially follows the ones for signatures. This allows us to instantiate US and UMACs from various assumptions such as DDH or CDH (latter in bilinear groups). For instance, we achieve instantiations for highly practical US schemes from BLS signatures [BLS01] or UMAC schemes from the Naor-Pinkas-Reingold PRF [NPR99]. Interestingly, by using some tricks, we can show how to generically construct UMACs from "almost" key-homomorphic PRFs [BLMR13] leading to constructions from the (R)LWE assumption, and thus, post-quantum UMACs. Unfortunately, there are no known key-homomorphic signatures with the required properties from post-quantum assumptions. Consequently, we investigate direct constructions of US from lattices. On the positive side we are able to provide a US construction based on the probabilistic GPV scheme [GPV08] under the SIS assumption. On the negative side, we therefore have to weaken the adversarial capabilities to prove it secure. While we provide formal evidence that this does not seem to be too problematic in practice, we consider it as a challenging open issue to construct US schemes from post-quantum assumptions being provable secure without any such restrictions.
Message-independent US and UMAC. We further discuss message-independent constructions of US and UMAC from the BLS signature scheme [BLS01], from the Pointcheval-Sanders signature scheme [PS16], and from the Naor-Pinkas-Reingold PRF [NPR99]. These overcome the limitations of the respective constructions directly obtained from them viewed as KH-signatures and -PRFs, which are message-dependent. Message-independence can be a desirable property in practical applications, as access to the message is not required for updating signatures and MACs, i.e., if they are verified and then stored and at a later point updated (in a batch) one does not need to access the respective messages and improves update performance.

2 Preliminaries

Notation. For $n \in \mathbb{N}$, let $[n] := \{1, \ldots, n\}$, and let $\lambda \in \mathbb{N}$ be the security parameter. For a finite set $\mathcal{S}$, we denote by $s \leftarrow \mathcal{S}$ the process of sampling

s uniformly from $\mathcal{S}$. For an algorithm A, let $y \leftarrow A(\lambda, x)$ be the process of running A on input (λ, x) with access to uniformly random coins and assigning the result to y (we may omit to mention the λ-input explicitly and assume that all algorithms take λ as input). To make the random coins r explicit, we write $A(\lambda, x; r)$. We use $\perp$ to indicate that an algorithm terminates with an error and A^B when A has oracle access to B, where B may return $\top$ as a distinguished special symbol. We say an algorithm A is probabilistic polynomial time (PPT) if the running time of A is polynomial in λ. Given $\mathbf{x} \in \mathbb{Z}^n$, we denote by $||\mathbf{x}||$ its infinity norm, i.e., for $\mathbf{x} = (x_1, x_2, \ldots, x_n)$, we have $||\mathbf{x}|| := \max(|x_1|, \ldots, |x_n|)$. A function f is negligible if its absolute value is smaller than the inverse of any polynomial (i.e., if $\forall c \exists k_0 \forall \lambda \geq k_0 : |f(\lambda)| < 1/\lambda^c$). We may write $q = q(\lambda)$ if we mean that the value q depends polynomially on λ.

Basic primitives. We recall some basic primitives.

Definition 1 (Pseudorandom Function [GGM84]). *Let $F\colon \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ be a keyed function. We say F is secure pseudorandom function (PRF) if for all efficient adversaries A, we have*

$$\left|\Pr\left[A^{F(k,\cdot)}(1^\lambda) = 1 \colon k \leftarrow \mathcal{K}\right] - \Pr\left[A^{f(\cdot)}(1^\lambda) = 1 \colon f \leftarrow \mathsf{Funs}[\mathcal{X}, \mathcal{Y}]\right]\right| = \varepsilon(\lambda),$$

where $\mathsf{Funs}[\mathcal{X}, \mathcal{Y}]$ denotes the set of all functions with domain $\mathcal{X}$ and range $\mathcal{Y}$ and $\varepsilon(\lambda) = \mathsf{negl}(\lambda)$.

Definition 2 (Signature scheme). *A signature scheme $\Sigma = (\mathsf{Gen}, \mathsf{Sig}, \mathsf{Ver})$ consists of the following PPT algorithms:*

$\mathsf{Gen}(\lambda)$: *On input security parameter λ, it outputs a signing key sk and a verification key pk with associated message space $\mathcal{M}$.*

$\mathsf{Sig}(sk, M)$: *On input a secret key sk and a message $M \in \mathcal{M}$, it outputs a signature σ.*

$\mathsf{Ver}(pk, M, \sigma)$: *On input a public key pk, a message $M \in \mathcal{M}$ and a signature σ, it outputs a bit $b \in \{0, 1\}$.*

A digital signature scheme Σ needs to provide (perfect) correctness as well as a notion of unforgeability. We first present the correctness definition.

A digital signature scheme Σ is correct, if for all security parameters $\lambda \in \mathbb{N}$, for all $(sk, pk) \leftarrow \mathsf{Gen}(\lambda)$, for all $M \in \mathcal{M}$, we have that

$$\Pr\left[\mathsf{Ver}(pk, M, \mathsf{Sig}(sk, M)) = 0\right] \leq \varepsilon(\lambda),$$

where $\varepsilon(\lambda) = \mathsf{negl}(\lambda)$, and we call it perfectly correct if $\varepsilon(\lambda) = 0$. We also require existential unforgeability under adaptively chosen message attacks (EUF-CMA security) notion. For EUF-CMA security the adversary has access to an oracle $\mathsf{Sig}'(sk, \cdot)$, which A can adaptively query on any message M of its choice and it returns $\mathsf{Sig}(sk, M)$. An adversary is valid if it does not query M^* to Sig'.

Definition 3 (EUF-CMA security of Σ). *A signature scheme Σ is EUF-CMA-secure iff for any valid PPT adversary A the advantage function*

$$\mathsf{Adv}^{\mathsf{euf\text{-}cma}}_{\Sigma,A}(\lambda) := \Pr\left[\mathsf{Exp}^{\mathsf{euf\text{-}cma}}_{\Sigma,A}(\lambda) = 1\right],$$

is negligible in λ where $\mathsf{Exp}^{\mathsf{euf\text{-}cma}}_{\Sigma,A}(\lambda)$ is defined in Figure 1.

```
Experiment $\mathsf{Exp}^{\mathsf{euf\text{-}cma}}_{\Sigma,A}(\lambda)$
  $(pk, sk) \leftarrow \mathsf{Gen}(\lambda)$
  $(M^*, \sigma^*) \leftarrow A^{\mathsf{Sig}'(sk,\cdot)}(pk)$
  if $A$ is valid and $\mathsf{Ver}(pk, M^*, \sigma^*) = 1$ then return 1 else return 0
```

Fig. 1. The EUF-CMA security notion for signatures.

Definition 4 (Message Authentication Code (MAC)). *A message authentication code $\Pi = (\mathsf{Gen}, \mathsf{Sig}, \mathsf{Ver})$ consists of the following PPT algorithms:*

$\mathsf{Gen}(\lambda)$: *On input security parameter λ it outputs a key sk with associated message space $\mathcal{M}$.*

$\mathsf{Sig}(sk, M)$: *On input a key sk and a message $M \in \mathcal{M}$ it outputs a tag (signature) σ.*

$\mathsf{Ver}(sk, M, \sigma)$: *On input a key pk, a message $M \in \mathcal{M}$ and a tag (signature) σ it outputs a bit $b \in \{0, 1\}$.*

We omit correctness, as it is defined analogously to signatures. We require EUF-CMA security of Π with verification queries.[3] Here, the adversary has access to oracles $\mathsf{Sig}'(sk, \cdot)$ and $\mathsf{Ver}'(sk, \cdot, \cdot)$ which A can adaptively query on any message M and message M and tag σ, respectively, and it returns $\mathsf{Sig}(sk, M)$ and $\mathsf{Ver}(sk, M, \sigma)$, respectively. An adversary is valid if it does not query Sig' on M^*.

Definition 5 (EUF-CMA security of Π). *A MAC Π is EUF-CMA-secure iff for any valid PPT adversary A the advantage function*

$$\mathsf{Adv}^{\mathsf{euf\text{-}cma}}_{\Pi,A}(\lambda) := \Pr\left[\mathsf{Exp}^{\mathsf{euf\text{-}cma}}_{\Pi,A}(\lambda) = 1\right],$$

is negligible in λ where $\mathsf{Exp}^{\mathsf{euf\text{-}cma}}_{\Pi,A}(\lambda)$ is defined in Figure 2.

Key-homomorphic signatures. We recall relevant parts of the definitional framework of key-homomorphic signatures as introduced in [DS19, DS16]. Let $\Sigma = (\mathsf{KGen}, \mathsf{Sign}, \mathsf{Verify})$ be a signature scheme and the secret and public key elements live in groups $(\mathbb{H}, +)$ and $(\mathbb{E}, \cdot)$, respectively. For these two groups it is required that group operations, inversions, membership testing as well as sampling from the uniform distribution are efficient.

[3] We note that this definition is equivalent to one without verification queries, but this makes our proofs more compact and natural.

```
Experiment $\mathsf{Exp}^{\mathsf{euf\text{-}cma}}_{\Pi,A}(\lambda)$
  $sk \leftarrow \mathsf{Gen}(\lambda)$
  $(M^*, \sigma^*) \leftarrow A^{\mathsf{Sig}'(sk,\cdot),\mathsf{Ver}'(sk,\cdot,\cdot)}(\lambda)$
  if $A$ is valid and $\mathsf{Ver}(sk, M^*, \sigma^*) = 1$ then return 1 else return 0
```

Fig. 2. The EUF-CMA security notion for MACs.

Definition 6 (Secret Key to Public Key Homomorphism [DS19]). *A signature scheme Σ provides a secret key to public key homomorphism, if there exists an efficiently computable map $\mu : \mathbb{H} \to \mathbb{E}$ such that for all $sk, sk' \in \mathbb{H}$ it holds that $\mu(sk + sk') = \mu(sk) \cdot \mu(sk')$, and for all $(sk, pk) \leftarrow \mathsf{Gen}(\lambda)$, it holds that $pk = \mu(sk)$.*

In the discrete logarithm setting, it is usually the case $sk \leftarrow \mathbb{Z}_p$ and $pk = g^{sk}$ with g being the generator of some group $\mathbb{G}$ of prime order p.

Definition 7 (Key-Homomorphic Signatures [DS16]). *A signature scheme is called key-homomorphic, if it provides a secret key to public key homomorphism and an additional PPT algorithm* Adapt*, defined as:*

$\mathsf{Adapt}(pk, M, \sigma, \Delta)$: *Given a public key pk, a message M, a signature σ, and a shift amount Δ outputs a public key pk' and a signature σ',*

such that for all $\Delta \in \mathbb{H}$ and all $(pk, sk) \leftarrow \mathsf{Gen}(\lambda)$, all messages $M \in \mathcal{M}$ and all σ with $\mathsf{Ver}(pk, M, \sigma) = 1$ and $(pk', \sigma') \leftarrow \mathsf{Adapt}(pk, M, \sigma, \Delta)$ it holds that

$$\Pr[\mathsf{Ver}(pk', M, \sigma') = 1] = 1 \quad \wedge \quad pk' = \mu(\Delta) \cdot pk.$$

The following notion covers whether adapted signatures look like freshly generated ones, *even* if the initial signature used in Adapt *is known.*

Definition 8 (Perfect Adaption [DS19]). *A key-homomorphic signature scheme provides perfect adaption, if for every $\kappa \in \mathbb{N}$, every message $M \in \mathcal{M}$, it holds that*

$$[\sigma, (sk, pk), \mathsf{Adapt}(pk, M, \sigma, \Delta)],$$

where $(sk, pk) \leftarrow \mathsf{Gen}(\lambda)$, $\sigma \leftarrow \mathsf{Sign}(sk, M)$, $\Delta \leftarrow \mathbb{H}$, and

$$[\sigma, (sk, \mu(sk)), (\mu(sk) \cdot \mu(\Delta), \mathsf{Sign}(sk + \Delta, M))],$$

where $sk \leftarrow \mathbb{H}$, $\sigma \leftarrow \mathsf{Sign}(sk, M)$, $\Delta \leftarrow \mathbb{H}$, are identically distributed.

Key-homomorphic PRFs. Key-homomorphic PRFs (KH-PRFs) are PRFs which satisfy additional algebraic properties. More precisely, the key space $\mathcal{K}$ and the range $\mathcal{Y}$ of the PRF exhibit certain group structures such that the evaluation of the PRF on any fixed input $x \in \mathcal{X}$ is homomorphic with the respect to these group structures. More precisely:

Definition 9 (Key-Homomorphic PRFs [NPR99, BLMR13]). *Let $(\mathcal{K}, \oplus)$, $(\mathcal{Y}, +)$ be groups. Then, a keyed function $F : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ is a key-homomorphic PRF (KH-PRF) if F is a secure PRF and for every key $k_1, k_2 \in \mathcal{K}$ and every input $x \in \mathcal{X}$, we have*

$$F(k_1, x) + F(k_2, x) = F(k_1 \oplus k_2, x).$$

We note that KH-PRFs constructed from assumptions such as Learning with Errors (LWE) as proposed in [BLMR13, Kim20, BEKS20] do not achieve the perfect homomorphism as described in the definition above, but are only "almost" key-homomorphic in that $F(k_1,x)+F(k_2,x) = F(k_1 \oplus k_2, x)+e$, where e is a small error term. For them one needs to bound the number of successive applications and provide T-time correctness for a pre-specified $T \geq 1$ (cf. [BLMR13, Kim20] for a comprehensive treatment). Note also, that only achieving an "almost" key-homomorphic property allows to distinguish fresh evaluations of the PRF from ones obtained via the key-homomorphic property.

3 Updatable MACs and Signatures

In this section, we present our definitional framework of updatable MACs and signatures. In order to make the illustration compact and avoid redundancy, we try to unify the notation as much as possible and will, whenever necessary, point to the differences between the two primitives.

3.1 Updatable MACs

We define updatable message authentication codes (UMACs) next and their security model in Section 3.3. An UMAC scheme UMAC with message space $\mathcal{M}$ is a tuple of the PPT algorithms $(\mathsf{Setup}, \mathsf{Next}, \mathsf{Sig}, \mathsf{Update}, \mathsf{Ver})$:

$\mathsf{Setup}(\lambda, n)$: on input security parameter $\lambda \in \mathbb{N}$ and the maximum number of epochs $n \in O(2^\lambda)$, the setup algorithm outputs a (secret) key k_1.[4]

$\mathsf{Next}(k_e)$: on input key k_e for epoch $e \in [n-1]$, the key-update algorithm outputs an updated key k_{e+1} together with an update token Δ_{e+1}.

$\mathsf{Sig}(k_e, M)$: on input key k_e for epoch $e \in [n]$ and a message $M \in \mathcal{M}$, the signing algorithm outputs a tag σ_e.[5]

$\mathsf{Update}(\Delta_{e+1}, M, \sigma_e)$: on input an update token Δ_{e+1}, a message M, and a tag σ_e for epoch $e < n$, the update algorithm outputs an updated message-tag pair (M, σ_{e+1}) or $\perp$.

$\mathsf{Ver}(k_e, M, \sigma_e)$: on input key k_e, a message M, and a tag σ_e for epoch $e \in [n]$, the verification algorithm outputs a verdict $b \in \{0,1\}$.

Correctness of UMAC. Correctness ensures that an update of a valid tag σ_e (via Δ_{e+1}) from epoch e to $e+1$ yields a valid tag σ_{e+1} that can be verified under the epoch key k_{e+1} which is derived from k_e. More formally, we require that for all $\lambda, n \in \mathbb{N}$, for all $k_1 \leftarrow \mathsf{Setup}(\lambda, n)$, for all $e \in [n-1]$, for all $(k_{e+1}, \Delta_{e+1}) \leftarrow \mathsf{Next}(k_e)$, for all $M \in \mathcal{M}$, for all σ_e with $\mathsf{Ver}(k_e, M, \sigma_e) = 1$, for all $(M, \sigma_{e+1}) \leftarrow \mathsf{Update}(\Delta_{e+1}, M, \sigma_e)$, we have that $\Pr\left[\mathsf{Ver}(k_{e'}, M, \sigma_{e'}) \neq 1\right] \leq \varepsilon(\lambda)$ holds, for all $e' \in [n]$, where $\varepsilon(\lambda) = \mathsf{negl}(\lambda)$, and we call it perfectly correct if $\varepsilon(\lambda) = 0$.

[4] See that such large values of n allow for virtually unbounded number of epochs.

[5] We assume that from keys, tokens, and tags, the associated epoch is efficiently extractable.

3.2 Updatable Signatures

We define updatable signatures (US) next and their security model in Section 3.3. An US scheme US with message space $\mathcal{M}$ is a tuple of the PPT algorithms $(\mathsf{Setup}, \mathsf{Next}, \mathsf{Sig}, \mathsf{Update}, \mathsf{Ver})$:

$\mathsf{Setup}(\lambda, n)$: on input security parameter λ and the maximum number of epochs $n \in O(2^\lambda)$, the setup algorithm outputs a public and secret key pair (pk_1, sk_1).[6]

$\mathsf{Next}(pk_e, sk_e)$: on input a public key pk_e and secret key sk_e for epoch $e \in [n-1]$, the key-update algorithm outputs an updated public key pk_{e+1}, an updated secret key sk_{e+1} and an update token Δ_{e+1}.

$\mathsf{Sig}(sk_e, M)$: on input secret key sk_e for epoch $e \in [n]$ and a message $M \in \mathcal{M}$, the signing algorithm outputs a signature σ_e.

$\mathsf{Update}(\Delta_{e+1}, M, \sigma_e)$: on input an update token Δ_{e+1}, a message M, and a signature σ_e for epoch $e < n$, the update algorithm outputs an updated message-signature pair (M, σ_{e+1}) or $\perp$.

$\mathsf{Ver}(pk_e, M, \sigma_e)$: on input public key pk_e, a message M, and a signature σ_e for epoch $e \in [n]$, the verification algorithm outputs a verdict $b \in \{0, 1\}$.

Correctness of US. For all $\lambda, n \in \mathbb{N}$, for all $(pk_1, sk_1) \leftarrow \mathsf{Setup}(\lambda, n)$, for all $e \in [n-1]$, for all $(pk_{e+1}, sk_{e+1}, \Delta_{e+1}) \leftarrow \mathsf{Next}(pk_e, sk_e)$, for all $M \in \mathcal{M}$, for all σ_e with $\mathsf{Ver}(pk_e, M, \sigma_e) = 1$, for all $(M, \sigma_{e+1}) \leftarrow \mathsf{Update}(\Delta_{e+1}, M, \sigma_e)$, we have that $\Pr\left[\mathsf{Ver}(pk_{e'}, M, \sigma_{e'}) \neq 1\right] \leq \varepsilon(\lambda)$ holds, for all $e' \in [n]$, where $\varepsilon(\lambda) = \mathsf{negl}(\lambda)$, and we call it perfectly correct if $\varepsilon(\lambda) = 0$.

3.3 Security of UMAC and US

We are now ready to define the security notions of UMAC and US where we will use UX with $\mathsf{X} \in \{\text{MAC, S}\}$ to distinguish between those two primitives. In order to make the description as compact as possible, we will use pk_e and sk_e, for $e \in [n]$, as handles to the public and secret key, respectively; where for UMACs we have $pk_e := \perp$ and $sk_e := k_e$. Moreover, we will also call the tags in UMACs signatures henceforth.

We introduce security definitions for existential unforgeability under chosen-message attack (UX-EUF-CMA) and unlinkable updates under chosen-message attack (UX-UU-CMA). Loosely speaking, the UX-EUF-CMA notion ensures that signatures cannot be forged even when the PPT adversary sees many signatures of chosen messages while the UX-UU-CMA notion guarantees that signatures derived from Update are unlinkable even when the PPT adversary sees many (updated) signatures of chosen messages.

In our security experiments, let $q \in \mathbb{N}$ be the number of signature queries and e the current epoch. Furthermore, we introduce a global state $\mathbf{S} = (\mathcal{I}, \mathcal{K}, \mathcal{T}, \mathcal{S})$:

When the experiment is initialized, we set $\mathcal{I} = \{((pk_1, sk_1), \Delta_1)\}$, for $(pk_1, sk_1) \leftarrow \mathsf{Setup}(\lambda, n)$ and $\Delta_1 := \perp$, and let $\mathcal{S}$, $\mathcal{K}$, and $\mathcal{T}$ be initially empty sets. Additionally, we require the following oracles which are eligible to change sets in $\mathbf{S}$ for any epoch $e' \in [e]$:

[6] As in UMACs, such large values of n allow for virtually unbounded number of epochs.

$\mathcal{I} = \{((pk_{e'}, sk_{e'}), \Delta_{e'})_{e' \in [e]}\}$: all keys and update tokens.
$\mathcal{K} = \{e' \in [e]\}$: all epochs where the adversary queried $\mathsf{Corrupt}(\mathsf{key}, e')$.
$\mathcal{T} = \{e' \in [e]\}$: all epochs where the adversary queried $\mathsf{Corrupt}(\mathsf{token}, e')$.
$\mathcal{S} = \{(e', M, \sigma_{e'})_{e' \in [e]}\}$: all tuples where the adversary queried $\mathsf{Sig}'(M, e')$ in epoch e' or $\mathsf{Update}'(M, \cdot)$ in epoch $e' - 1$;

$\mathsf{Sig}'(M, e')$: on input message M and epoch $e' \in [e]$, compute signature $\sigma_{e'} \leftarrow \mathsf{Sig}(sk_{e'}, M)$, set $\mathcal{S} := \mathcal{S} \cup \{(e', M, \sigma_{e'})\}$, and return $\sigma_{e'}$. Else, return $\perp$.
Next' : find $(pk_e, sk_e) \in \mathcal{I}$, compute $(pk_{e+1}, sk_{e+1}, \Delta_{e+1}) \leftarrow \mathsf{Next}(pk_e, sk_e)$, update $\mathcal{I} := \mathcal{I} \cup \{((pk_{e+1}, sk_{e+1}), \Delta_{e+1})\}$, return pk_{e+1} and set $e := e + 1$.
$\mathsf{Update}'(M, \sigma_{e'})$: on input a message-signature pair $(M, \sigma_{e'})$, return $\perp$ if $\mathsf{Ver}'(M, \sigma_{e'}) \neq 1$; else, compute $(M, \sigma_{e'+1}) \leftarrow \mathsf{Update}(\Delta_{e'+1}, M, \sigma_{e'})$, set $\mathcal{S} := \mathcal{S} \cup \{(e' + 1, M, \sigma_{e'+1})\}$ and return $\sigma_{e'+1}$.
$\mathsf{Corrupt}(\{\mathsf{token}, \mathsf{key}\}, e')$: on input handles token or key, and epoch $e' \in [e]$,
- return $\Delta_{e'+1}$ and set $\mathcal{T} := \mathcal{T} \cup \{e'\}$, if called with token and $e' < e$,
- return $sk_{e'}$ and set $\mathcal{K} := \mathcal{K} \cup \{e'\}$, if called with key. Else, return $\perp$.

$\mathsf{Ver}'(M, \sigma_{e'})$: on input a message-signature pair $(M, \sigma_{e'})$,
- return $b \leftarrow \mathsf{Ver}(sk_{e'}, M, \sigma_{e'})$ in the UMAC case,
- return $b \leftarrow \mathsf{Ver}(pk_{e'}, M, \sigma_{e'})$ in the US case.

Leakage profile $(\mathcal{K}^*, \mathcal{T}^*, \mathcal{S}^*)$**.** We use the concept of a leakage profile originally defined in [LT18] to capture key, token, and signature "leakage" that cannot be directly captured via oracles. The reason is that due to the nature of signature updates, UX schemes inherently allow for information leakage of updated message-signature pairs, keys, and tokens besides what is modeled via the global state $\mathbf{S}$. For example, one token $\Delta_{e'+1}$ alone in such schemes is capable of updating polynomially many message-signature pairs $((M_1, \sigma_{1,e'}), \ldots, (M_\ell, \sigma_{\ell,e'}))$, for all $\ell = \ell(\lambda)$ and for each epoch $e' \in [n-1]$. As this is required by the correctness of the scheme, we cannot capture which particular signature σ' the adversary retrieves (via an update token) and, hence, cannot include it into $\mathcal{S}$.

Furthermore, signatures, keys, and tokens cannot only be "upgraded" but also potentially "downgraded", e.g., a token $\Delta_{e'}$ and key $sk_{e'}$ or a token $\Delta_{e'}$ and a message-signature pair $(M, \sigma_{e'})$ for epoch $e' \in [n]$ might be used to derive a key $sk_{e'-1}$ or message-signature pair $(M, \sigma_{e'-1})$ of the previous epoch $e' - 1$, respectively. Hence, we cannot capture which particular key sk' or signature σ' the adversary retrieves (via an update token) and, hence, cannot include those as well into $\mathcal{K}$ or $\mathcal{S}$, respectively.

We want to emphasize that the directionality of updates, i.e., either bidirectional or unidirectional, is subject to discussion in updatable encryption [Jia20]. In context of US or UMACs, we observe that due to correctness one always can upgrade signatures (so leakage in this direction does not add anything), and stronger schemes could only prevent to derive keys or signatures "into the past". This, however, seems of limited interest in authentication primitives, where old keys are typically assumed to be invalidated. Consequently, we opted for the arguably simpler bidirectional setting.

When looking ahead, we will construct US from key-homomorphic (KH) signatures. Now, in [DS19], the authors also provide a number of constructions of KH signatures that provide a property being weaker than the one we are using and which is called adaption of signatures. Now, one could wonder why we do not support such schemes. Firstly, it would only allow to achieve a very weak notion of unlinkability. Secondly, and more importantly, all known KH signatures with this "weak" adaption (e.g., Schnorr or Guillou-Quisquater signatures) have the property that a signature and its updated version leak the update token. Consequently, an adversary who obtains a signing key in some old epoch and then sees a signature and its updated versions, can compute all the signing keys up to the epoch of the latest updated version it sees. As this results in very weak security guarantees, we decided that our framework should not support schemes with these weak security guarantees.

Now, let us consider an UX scheme with optimal leakage to be the one where we only have signature upgrade (but no downgrade) and tokens are not useful to upgrade or downgrade keys in any way. The leakage would be limited for such schemes, however, the model would still need to restrict the adversary to retrieve the update token in epoch $e^* - 1$, i.e., Δ_{e^*}, where e^* is the forgery epoch. The reason is that otherwise the adversary could trivially win the game by updating any signature computed under a corrupted key to epoch e^*. Hence, also such strong schemes with so-called no-directional key updates would not achieve any stronger security in our model; at least with the applications we have in mind.

Now we are ready to introduce the leakage profile. We model leakage via key-update, token, and signature-update inferences where the leakage profile $(\mathcal{K}^*, \mathcal{T}^*, \mathcal{S}^*)$ of a concrete scheme is specified by the respective sets.

Key-update inferences. Key-update inferences of a specific UX scheme can be formally captured as $\mathcal{K}^*$ with corrupted-key set $\mathcal{K}$ and corrupted-token set $\mathcal{T}$ maintained by the oracles:

$$\mathcal{K}^* := \begin{cases} \{e \in [n] \mid \mathsf{corrupt\text{-}key}(e) = \mathsf{true}\} \text{ with } \mathsf{true} = \mathsf{corrupt\text{-}key}(e) \text{ iff:} \\ \quad (e \in \mathcal{K}) \lor (e - 1 \in \mathcal{K} \land e \in \mathcal{T}) \lor (e + 1 \in \mathcal{K} \land e + 1 \in \mathcal{T}). \end{cases}$$

Token inferences. Token inferences can be formally captured as $\mathcal{T}^*$ with corrupted-token set $\mathcal{T}$ and key-leakage set $\mathcal{K}^*$:

$$\mathcal{T}^* := \{e \in [n] \mid (e \in \mathcal{T}) \;\lor\; (e - 1 \in \mathcal{K}^* \;\land\; e \in \mathcal{K}^*)\}.$$

Signature-update inferences. Signature-update inferences can be formally captured as $\mathcal{S}^*$ with corrupted-signature set $\mathcal{S}$ maintained by the oracles and sets $\mathcal{K}^*$ and $\mathcal{T}^*$ with $M \in \mathcal{M} \cup \{\top\}$[7]:

$$\mathcal{S}^* := \begin{cases} \{(e, M) \mid \mathsf{corrupt\text{-}sig}(e, M) = \mathsf{true}\} \text{ with } \mathsf{true} = \mathsf{corrupt\text{-}sig}(e, M) \text{ iff:} \\ \quad ((e, M, \cdot) \in \mathcal{S}) \;\lor\; ((e, M) \in \mathcal{K}^* \times \{\top\}) \;\lor\; (\mathsf{corrupt\text{-}sig}(e - 1, M) \;\land \\ \quad e \in \mathcal{T}^*) \;\lor\; (\mathsf{corrupt\text{-}sig}(e + 1, M) \;\land\; e + 1 \in \mathcal{T}^*), \end{cases}$$

[7] $M = \top$ is a placeholder for "all messages" in $\mathcal{M}$ and helps us to construct the set $\mathcal{S}^*$ efficiently.

where $\mathsf{corrupt\text{-}sig}(0, M) = \mathsf{false}$.

In Figure 3 we provide an example of potential leakage in UX schemes with our leakage profile.

epoch:	$e-5$	$e-4$	$e-3$	$e-2$	$e-1$	e	$e+1$	$e+2$	$e+3$	$e+4$
keys:	🟦 k_{e-5}	🟩 k_{e-4}	🟦 k_{e-3}	🟩 k_{e-2}	k_{e-1}	k_e	🟩 k_{e+1}	🟦 k_{e+2}	k_{e+3}	k_{e+4}
tokens:	🟩 Δ_{e-4}	🟦 Δ_{e-3}	🟩 Δ_{e-2}	Δ_{e-1}	🟩 Δ_e	Δ_{e+1}	🟩 Δ_{e+2}	Δ_{e+3}	Δ_{e+4}	🟩 Δ_{e+5}
signature:	🟦 σ_{e-5}	🟩 σ_{e-4}	🟦 σ_{e-3}	🟦 σ_{e-2}	σ_{e-1}	σ_e	🟩 σ_{e+1}	🟦 σ_{e+2}	🟩 σ_{e+3}	σ_{e+4}

Fig. 3. Example of directly obtained (green) and inferable information (blue) for UX schemes.

Existential unforgeability under chosen-message attacks (UX-EUF-CMA). Informally, the UX-EUF-CMA notion ensures that no PPT adversary can non-trivially forge signatures even when the adversary adaptively compromises a number of keys and tokens. We say that an UX scheme is UX-EUF-CMA-secure if any PPT adversary succeeds in the following experiment only with negligible probability. The experiment starts by computing the initial keys $(pk_1, sk_1) \leftarrow \mathsf{Setup}(\lambda, n)$. During the experiment, via the oracles, the adversary may query signatures for any epoch e' up to the current epoch e, iterate to the next epoch $e+1$, update signatures, and corrupt tokens or keys for any epoch e' up to the current epoch e (note that the global state $\mathbf{S}$ is changed by the oracles). Eventually, the adversary outputs a message-signature pair $(M^*, \sigma^*_{e^*})$, for epoch $e^* \in [n]$, and succeeds if $\mathsf{Ver}(pk_{e^*}, M^*, \sigma^*_{e^*}) = 1$ in the US case and $\mathsf{Ver}(sk_{e^*}, M^*, \sigma^*_{e^*}) = 1$ in the UMAC case, and the adversary is *valid* which we define in Definition 10.

Definition 10 (Validity of A for UX-EUF-CMA). *Depending on $(\mathcal{S}^*, \mathcal{K}^*, \mathcal{T}^*)$, a PPT adversary A is valid in the UX-EUF-CMA experiment if*

$$\{\{(e^*, \top)\} \cup \{(e^*, M^*)\}\} \cap \mathcal{S}^* = \emptyset, \tag{1}$$

i.e., A has not learned any useful forgery-message information for epoch e^*.

Remark. Definition 10 essentially says that the adversary is not able to derive a valid message-signature pair for epoch e^* which excludes trivial wins. The leftmost term in Equation (1) "checks" that A does not possess a valid (derived) secret key in e^* while the middle term "checks" that A is not able to derive a valid signature for M^* in epoch e^* via corrupted tokens.

See that the keys (pk_{e^*}, sk_{e^*}) for any $e^* \in [n]$ can be derived, i.e., if $e^* \leq e$, we have that $((pk_{e^*}, sk_{e^*}), \cdot) \in \mathcal{I}$, otherwise, if $e^* > e$, we can derive (pk_{e^*}, sk_{e^*})

iteratively by invoking Next' starting with (pk_e, sk_e). If $e^* \leq e$ we set $e_{\max} := e$, else $e_{\max} := e^*$. Figure 4 depicts the UX-EUF-CMA experiments.

Definition 11 (UX-EUF-CMA security of UX). *A UX scheme* UX *is UX-EUF-CMA-secure iff for any valid PPT adversary A the advantage function*

$$\mathsf{Adv}^{\text{ux-euf-cma}}_{\mathsf{UX},A}(\lambda, n) := \Pr\left[\mathsf{Exp}^{\text{ux-euf-cma}}_{\mathsf{UX},A}(\lambda, n) = 1\right],$$

is negligible in λ, where $\mathsf{Exp}^{\text{ux-euf-cma}}_{\mathsf{US},A}(\lambda, n)$ is defined in Figure 4.

```
Experiment $\mathsf{Exp}^{\text{ux-euf-cma}}_{\mathsf{UX},A}(\lambda, n)$
  $(pk_1, sk_1) \leftarrow \mathsf{Setup}(\lambda, n)$
  $\mathbf{S} = (\mathcal{I}, \mathcal{K}, \mathcal{T}, \mathcal{S})$, for $\mathcal{I} := \{((pk_1, sk_1), \perp)\}, \mathcal{K} := \mathcal{T} := \mathcal{S} := \emptyset$
  $(M^*, \sigma^*_{e^*}) \leftarrow A^{\mathsf{Sig}', \mathsf{Next}', \mathsf{Update}', \mathsf{Ver}', \mathsf{Corrupt}}(\lambda)$
  if $A$ is valid and $\mathsf{Ver}(\{pk_{e^*}, sk_{e^*}\}, M^*, \sigma^*_{e^*}) = 1$ then return 1 else return 0
```

Fig. 4. The UX-EUF-CMA security notions for UX. For US, we verify $\mathsf{Ver}(pk_{e^*}, M^*, \sigma^*_{e^*}) = 1$; for UMAC, we use $\mathsf{Ver}(sk_{e^*}, M^*, \sigma^*_{e^*}) = 1$ in the last step.

Unlinkable updates under chosen-message attacks (UX-UU-CMA). Informally, the UX-UU-CMA notion ensures that no PPT adversary can distinguish fresh signatures from updated signatures even seeing (all) keys, update tokens and signatures from the past. We say that an UX scheme is UX-UU-CMA-secure if any PPT adversary succeeds in the following experiment only with negligible probability. The experiment starts by computing the initial keys $(pk_1, sk_1) \leftarrow \mathsf{Setup}(\lambda, n)$. During the experiment, via the oracles, the adversary may query signatures for any epoch e' up to the current epoch e, iterate to the next epoch $e + 1$, update signatures, and corrupt tokens or keys for any epoch e' up to the current epoch e, and has access to a verification oracle (note that the global state $\mathbf{S}$ is changed by the oracles). Then, the adversary outputs a message M^* and epoch e^* for which it queried Sig' in some epoch $e' < e^*$. It receives a challenge signature $\sigma^{(b)}$ which is either a fresh signature on M^*, $\sigma^{(0)}$, or the existing signature for M^* updated to epoch e^*, $\sigma^{(1)}$. For the latter case, we use the compact notation of $\mathsf{UpdateCh}(M^*)$ to denote the repeated application of Update' starting with $\sigma_{e'}$ for M^* and finally resulting in $\sigma^{(1)}$ as signature for M^* in epoch e^* (note that Update' implicitly checks the condition $\mathsf{Ver}(pk_{e'}, M^*, \sigma_{e'}) = 1$). In both cases, this might require calling repeatedly Next until (sk_{e^*}, pk_{e^*}) is defined. Eventually it outputs a bit b^* and wins if $b = b^*$. Note that the adversary can call $\mathsf{Corrupt}$ arbitrarily. We call an adversary valid if it queried M^* to Sig' in some epoch $e' < e^*$. Figure 5 depicts the UX-UU-CMA experiments.

Definition 12 (UX-UU-CMA security of UX). *A UX scheme* UX *is UX-UU-CMA-secure iff for any valid PPT adversary A the advantage function*

$$\mathsf{Adv}^{\text{ux-uu-cma}}_{\mathsf{UX},A}(\lambda, n) := |\Pr\left[\mathsf{Exp}^{\text{ux-uu-cma}}_{\mathsf{UX},A}(\lambda, n) = 1\right] - 1/2|,$$

is negligible in λ, where $\mathsf{Exp}^{\text{ux-uu-cma}}_{\mathsf{US},A}(\lambda, n)$ is defined in Figure 5.

```
**Experiment** $\mathsf{Exp}^{\text{ux-uu-cma}}_{\mathsf{UX},A}(\lambda, n)$
  $(pk_1, sk_1) \leftarrow \mathsf{Setup}(\lambda, n)$
  $\mathbf{S} = (\mathcal{I}, \mathcal{K}, \mathcal{T}, \mathcal{S})$, for $\mathcal{I} := \{((pk_1, sk_1), \perp)\}, \mathcal{K} =: \mathcal{T} =: \mathcal{S} := \emptyset$
  $(M^*, e^*) \leftarrow A^{\mathsf{Sig}', \mathsf{Next}', \mathsf{Update}', \mathsf{Ver}', \mathsf{Corrupt}}(\lambda)$
  $b \leftarrow \{0, 1\}$
  $\sigma^{(0)} \leftarrow \mathsf{Sig}(sk_{e^*}, M^*),\ \sigma^{(1)} \leftarrow \mathsf{UpdateCh}(M^*)$
  $b^* \leftarrow A^{\mathsf{Sig}', \mathsf{Next}', \mathsf{Update}', \mathsf{Ver}', \mathsf{Corrupt}}(\sigma^{(b)})$
  if $(e', M^*, \cdot) \in \mathcal{S}$, $e' < e^*$, and $b = b^*$ then return 1 else return 0
```

Fig. 5. The UX-UU-CMA security notions for UX.

4 Construction of Updatable Signatures

In this section, we will present different instantiations of updatable signatures from different assumptions (see Section 4.4 for an overview and discussion.)

4.1 Updatable Signatures from KH Signatures

Subsequently, we show how to generically construct US with polynomially many updates from key-homomorphic (KH) signatures. Let $\Sigma = (\mathsf{Gen}, \mathsf{Sig}, \mathsf{Adapt}, \mathsf{Ver})$ be a KH signature scheme providing perfect adaption, where we denote the secret key space by $\mathbb{H}$ and the secret key to public key homomorphism by μ. The so-obtained US scheme US is depicted in Figure 6. Before discussing the security,

```
$\mathsf{Setup}(1^\lambda, n)$:
  – Return $(pk_1, sk_1) \leftarrow \Sigma.\mathsf{Gen}(1^\lambda)$.
$\mathsf{Next}(pk_e, sk_e)$:
  – Choose random $sk' \in \mathbb{H}$ and set $\Delta'_{e+1} := sk'$ and $\Delta_{e+1} := (\Delta'_{e+1}, pk_e)$.
  – Compute $pk_{e+1} = pk_e \cdot \mu(\Delta'_{e+1})$ and $sk_{e+1} = sk_e + \Delta'_{e+1}$.
  – Return $(pk_{e+1}, sk_{e+1}, \Delta_{e+1})$.
$\mathsf{Sig}(sk_e, M)$:
  – Return $\Sigma.\mathsf{Sig}(sk_e, M)$.
$\mathsf{Update}(\Delta_{e+1}, M, \sigma_e)$:
  – Parse $\Delta_{e+1} = (\Delta'_{e+1}, pk_e)$
  – Compute $\sigma_{e+1} := \Sigma.\mathsf{Adapt}(pk_e, M, \sigma_e, \Delta'_{e+1})$.
  – Return $(M, \sigma_{e+1})$.
$\mathsf{Ver}(pk_e, M, \sigma_e)$:
  – Return $\Sigma.\mathsf{Ver}(pk_e, M, \sigma_e)$.
```

Fig. 6. US from KH signatures.

we will note that correctness straightforwardly follows from inspection.

Theorem 1. *Let $\Sigma = (\mathsf{Gen}, \mathsf{Sig}, \mathsf{Adapt}, \mathsf{Ver})$ be a uniform-keys key-homomorphic signature scheme. If Σ is EUF-CMA secure and provides perfect adaption, then the updatable signature scheme* US *from Figure 6 is US-EUF-CMA-secure and US-UU-CMA secure.*

In the above theorem, we require uniform-keys KH signatures, which we introduce now. This notion is satisfied by all natural schemes and in particular the ones discussed in [DS19]:

Definition 13 (Uniform-Keys Key-Homomorphic Signatures). *A key-homomorphic signature scheme Σ is said to be uniform-keys if the distribution of sk with $(sk, pk) \leftarrow \Sigma.\mathsf{Gen}(1^\lambda)$ is the uniform distribution over the secret key space $\mathbb{H}$.*

Helper Lemmas. Notice that a valid adversary should not trivially forge signatures using the updatable property of the US scheme and the information provided by the leakage profile (e.g., produce a valid signature for epoch $e' \in \mathcal{K}^*$ using $sk_{e'}$ and update it to a valid signature for epoch e^* using the update tokens in $\mathcal{T}^*$) for epochs in an appropriate window containing epoch e^*. More formally, we can define

$$e^- := \max_{1 \le e \le e^*} \{e \mid e \notin \mathcal{T}^* \cup \mathcal{K}^* \text{ and } e' \notin \mathcal{K}^*,\ e' \in \mathcal{T}^* \quad \forall\ e < e' \le e^*\},$$
$$e^+ := \min_{e^* < e \le e_{\max}} \{e \mid e \notin \mathcal{T}^* \text{ and } e' \notin \mathcal{K}^*,\ e' \in \mathcal{T}^* \quad \forall\ e^* < e' < e\},$$

and let the interval $[e^-, e^+[$ denote such a window and notice that e^- is well-defined. Suppose on the contrary that the set

$$E := \{e \in [1, e^*] \mid e \notin \mathcal{T}^* \cup \mathcal{K}^* \ \wedge\ e' \notin \mathcal{K}^*,\ e' \in \mathcal{T}^* \quad \forall\ e < e' \le e^*\},$$

is empty (if it is not empty then it has a maximum element). We claim, by backward induction on the epoch number k and starting from e^*, that this implies that $[k, e^*] \subseteq \mathcal{T}^*$ for all $k \in [1, e^*]$ which is a contradiction as $1 \notin \mathcal{T}^*$ by construction of $\mathcal{T}^*$. The base case is $k = e^*$. Indeed if E is empty then $e^* \notin E$ which implies that $e^* \in \mathcal{T}^*$ as $\nexists e'$ with $e^* < e' \le e^*$ and it cannot be that $e^* \in \mathcal{K}^*$. Now assume by induction hypothesis that $[k, e^*] \subseteq \mathcal{T}^*$ for $k < e^*$. We deduce from it, using that $k-1 \notin E$, that $k-1 \in \mathcal{T}^*$ as $k-1$ cannot be in $\mathcal{K}^*$ by validity of the adversary when $[k, e^*] \subseteq \mathcal{T}^*$. Hence, $[k-1, e^*-1] \subseteq \mathcal{T}^*$ which concludes the proof. A similar argument also proves the well-definedness of e^+. We can summarize the above discussion in the following lemma.

Lemma 1. *Let A be a valid adversary that produces a forgery in epoch $0 < e^* \le e_{\max}$ in the US-EUF-CMA experiment, then there exists a maximum integer $0 < e^- \le e^*$ and a minimum integer $e^* < e^+ \le e_{\max}$ s.t. A*

1) does not obtain tokens Δ_{e^-} and Δ_{e^+},
2) obtains no secret key sk_e for all $e^- \le e < e^+$ and
3) can obtain all tokens Δ_e for $e^- < e < e^+$,

from the queries made to the oracles. Subsequently, we often denote the interval $[e^-, e^+[$ as the window.

From Definition 13, the following lemma easily follows.

Lemma 2. *Let Σ be a uniform-keys key-homomorphic signature scheme. Then the following hold:*

1) For every $(sk, pk) \leftarrow \Sigma.\mathsf{Gen}(1^\lambda)$ the distributions of $sk + \Delta$ with $\Delta \leftarrow \mathbb{H}$ and sk' with $(sk', pk') \leftarrow \Sigma.\mathsf{Gen}(1^\lambda)$ are identical.

2) For every $(sk, pk) \leftarrow \Sigma.\mathsf{Gen}(1^\lambda)$ and $\Delta \in \mathbb{H}$ we have that $(sk+\Delta, pk\cdot\mu(\Delta)) \in \Sigma.\mathsf{Gen}(1^\lambda)$.

Now we are ready for the proof of Theorem 1.

Proof. First we observe that correctness follows straightforwardly from inspection.

For US-UU-CMA security, we can observe that in the US-UU-CMA experiment all keys, signing operations, updates and token computations are performed honestly by the experiment. Any adversary A is given access to all keys, tokens and signatures and what we need to consider is the computation of the challenge signature $\sigma^{(b)}$. Now, due to the adaption property of Σ, the outputs of $\Sigma.\mathsf{Sig}$ and $\Sigma.\mathsf{Adapt}$ are identical and thus also the outputs of $\mathsf{US.Sig}$ and $\mathsf{US.Update}$. By repeatedly applying Definition 8 within $\mathsf{US.UpdateCh}$ for computing $\sigma^{(1)}$ and using Lemma 2 we are done. More formally, let us consider the sequence of games as outlined below:

Game 0. This is the experiment $\mathsf{Exp}^{\text{us-uu-cma}}_{\mathsf{US},A}(\lambda, n)$ with $b = 0$, i.e., we always return $\sigma^{(0)} \leftarrow \mathsf{Sig}(sk_{e^*}, M^*)$.

Game 1. This is the experiment $\mathsf{Exp}^{\text{us-uu-cma}}_{\mathsf{US},A}(\lambda, n)$ with $b = 1$, i.e., we always return $\sigma^{(1)} \leftarrow \mathsf{UpdateCh}(M^*)$.

Lemma 3 (Game 0 to Game 1). *For any adversary A it holds that*

$$|\Pr[S_{A,0}] - \Pr[S_{A,1}]| = 0.$$

Observe that in both games the adversary A is given access to all keys, tokens and signatures and outputs a message M^* and epoch e^* for which it queried Sig' in some epoch $e' \leq e^*$. Now, in **Game 0** we finally output $\sigma^{(0)} \leftarrow \mathsf{Sig}(sk_{e^*}, M^*)$, i.e., a fresh signature of M^* that verifies under pk_{e^*} to A. In **Game 1** let us denote by $\sigma_{e'}$ the signature for M^* under $(sk_{e'}, pk_{e'})$ that the adversary queried for message M^* during the experiment. Let us w.l.o.g. assume that (sk_{e^*}, pk_{e^*}) is already defined (otherwise repeatedly call Next until it is defined) and let $(sk_{e'+1}, pk_{e'+1}, \Delta_{e'+1}), \ldots, (sk_{e^*}, pk_{e^*}, \Delta_{e^*})$ be the sequence of keys and update tokens. Now, $\mathsf{UpdateCh}$ does the following:

- For $i = e', \ldots, e^* - 1$ compute $\sigma_{i+1} \leftarrow \mathsf{Update}(\Delta_{i+1}, M^*, \sigma_i)$, where Update parses $\Delta_{i+1} = (\Delta'_{i+1}, pk_i)$ and calls $\sigma_{i+1} \leftarrow \Sigma.\mathsf{Adapt}(pk_i, M^*, \sigma_i, \Delta'_{i+1})$.

By using Lemma 2, we know that every key pair in the sequence $(sk_{e'}, pk_{e'}), \ldots, (sk_{e^*}, pk_{e^*})$ is distributed identical as one obtained from $\Sigma.\mathsf{Gen}$. Now, this allows us to repeatedly apply the perfect adaption notion to the output of the previous Update in this sequence to conclude that in the sequence of signatures $(\sigma_{e_i}, \ldots, \sigma_{e^*} =: \sigma^{(1)})$ the last signature $\sigma^{(1)}$ is distributed identically to a fresh signature computed as $\mathsf{Sig}(sk_{e^*}, M^*)$.

This lets us conclude that $\mathsf{Adv}^{\text{us-uu-cma}}_{\mathsf{US},A}(\lambda, n) = 0$ for any adversary A, which concludes the proof of US-UU-CMA security. $\square$

To prove US-EUF-CMA security, we reduce US-EUF-CMA security to the EUF-CMA security of $\varSigma$, where the challenger will be associated to period e^-. While we can use the Sig oracle of the EUF-CMA challenger for period e^-, we have to answer Sig queries for epoch e^* and adjacent ones, and Update queries from older epochs to e^* and from e^* to newer epochs. Moreover, we have to provide secret keys and tokens on $\mathsf{Corrupt}$ queries to the adversary. By Lemma 1 we know that in order for an adversary to be valid and to rule out a trivial forgery, there needs to be a maximum epoch $1 \leq e^- \leq e^*$ and a minimum epoch $e^* < e^+ \leq e_{\max}$ for which A does not query tokens $\varDelta_{e^-}$ and $\varDelta_{e^+}$ and consequently does not know any secret key but knows all tokens for the window of epochs $[e^-, e^+[$. We now can use the key insulation technique from Klooß et al. [KLR19] for optimization, so that instead of guessing the challenge epoch e^* and the window to the left and to the right, we only guess the boundaries of this region e^- and e^+ (containing the challenge epoch somewhere) and we can just associate the EUF-CMA challenger of $\varSigma$ to some epoch in this interval (lets say e^-). This reduces the overall reduction loss from $e_{\max}^2(e_{\max}+1)$ to $e_{\max}(e_{\max}+1)$.

Outside of the window, i.e., for epochs up to $e^- - 1$ and starting from e^+ upwards, we will behave in our simulation as in the original game and in particular choose and know all secret keys and update tokens (except $\varDelta_{e^+}$). For all epochs inside the window, our strategy will be as follows: we do not know the secret keys associated to the epochs, but they are implicitly set by choosing for every epoch e_i in the window a random token $\varDelta_{e_i}$ as in the real Next algorithm. Then for every epoch e_i in the window starting from e^- we use the secret key to public key homomorphism of $\varSigma$ and set the corresponding public key as $pk_{e_i} = pk_{e_{i-1}} \cdot \mu(\varDelta_{e_i})$ (for $e^- < e_i < e^+$). Now for any signature query for message M and epoch e_i within the window, we query M to the EUF-CMA challenger of $\varSigma$ associated to e^- and use the $\varSigma.\mathsf{Adapt}$ algorithm in the forwards direction to obtain the signature for M in epoch e_i within the window. Note that due to the adaption property of $\varSigma$ and thus the identical distribution of signatures from $\varSigma.\mathsf{Sig}$ and $\varSigma.\mathsf{Adapt}$, this is indistinguishable for A. The Update' oracle is performed as in the original game for all those epochs where the update token is knows. For the remaining epochs, i.e., e^- and e^+, when asked to update $(M, \sigma_{e^- -1})$ (or $(M, \sigma_{e^+ -1})$, respectively), we query M to the EUF-CMA challenger of $\varSigma$ associated to e^- (or produce a fresh signature using sk_{e^+}, respectively) and return it to the adversary. Again by the adaption property of $\varSigma$ and thus the identical distribution of freshly generated signatures and updated ones, this is indistinguishable. Now if A outputs a valid forgery $(M^*, \sigma^*_{e^*})$ for epoch e^*, if $e^* = e^-$ we can directly output it. Otherwise we use $\varSigma.\mathsf{Adapt}$ to adapt the forgery backwards into epoch e^- and output it. Note that in any case a valid forgery output by A represents a valid forgery for $\varSigma$, as validity guarantees that we have never queried M^* throughout the game for any epoch inside the window.

More formally, let us consider the following sequence of games.

Game 0. This is the experiment $\mathsf{Exp}^{\mathsf{us\text{-}euf\text{-}cma}}_{\mathsf{US},A}(\lambda, n)$.

Game 1. This is identical to the previous game with the exception that we guess the window $[e^-, e^+[$ in which the epoch e^* for which A outputs the forgery is located and abort if our guess is incorrect.

Game 2. This is identical to the previous game up to the following differences:

- For call to Next' in epoch $e^- - 1$ we set $\varDelta_{e^-} := \perp$ and run $(sk_{e^-}, pk_{e^-}) \leftarrow \varSigma.\mathsf{Gen}(1^\lambda)$ to obtain an independent key for epoch e^-. The same is done for a call to Next' in epoch $e^+ - 1$.
- For each call to the Next' oracle for epoch $e \in \{e^-, \ldots, e^+ - 2\}$, we run $\mathsf{Next}(pk_e, \perp)$ where we ignore the secret key and just set the key implicitly via the public-key, i.e., choose random $sk' \in \mathbb{H}$ and set $\varDelta'_{e+1} := sk'$ and $\varDelta_{e+1} := (\varDelta'_{e+1}, pk_e)$, compute $pk_{e+1} := pk_e \cdot \mu(\varDelta_{e+1})$, set $sk_{e+1} = \perp$.
- For each call to the Sig' oracle for message M in any epoch e within $[e^-, e^+[$, we compute $\sigma \leftarrow \varSigma.\mathsf{Sig}(sk_{e^-}, M)$ and then call $\varSigma.\mathsf{Adapt}$ using the respective public keys to adapt it to a signature σ_e valid under pk_e and add (e, M, σ_e) to $\mathcal{S}$.
- For each call to the Update' oracle for signature (M, σ_e) in epoch $e \in \{e^- - 1, e^+ - 1\}$, we compute $\sigma_{e+1} \leftarrow \varSigma.\mathsf{Sig}(sk_{e+1}, M)$. We then add $(e+1, M, \sigma_{e+1})$ to $\mathcal{S}$ and $\mathcal{U}$, and return σ_{e+1} to the adversary.

Now, let us analyze the transitions:

Lemma 4. *For any adversary A it holds that*

$$\left(\frac{1}{(e_{max}+1)e_{max}}\right) \Pr[S_{A,0}] \leq \Pr[S_{A,1}].$$

Proof. We guess the window by simply drawing $e^- \leftarrow \{0, ..., e_{max}\}$ and $e^+ \leftarrow \{e^- + 1, ..., e_{max}\}$ uniformly at random. Thus, this guess is correct with probability at least $\frac{1}{(e_{max}+1)e_{max}}$ and if the guess turns out to be wrong, we abort. Note that such a window always exists for a valid adversary A due to Lemma 1. □

Lemma 5. *For any adversary A it holds that*

$$|\Pr[S_{A,1}] - \Pr[S_{A,2}]| = 0.$$

Proof. We observe that due to having a valid adversary A w.r.t. window $[e^-, e^+[$ and due to Lemma 1 we recall that A

1) does not obtain tokens $\varDelta_{e^-}$ and $\varDelta_{e^+}$,
2) obtains no secret key sk_e for all $e^- \leq e < e^+$ and
3) can obtain all tokens $\varDelta_e$ for $e^- < e < e^+$,

from the queries made to the oracles, given the leakage profile. Note that due to 1) we know that there implicitly exists a token mapping keys and signatures from $e^- - 1$ to e^- and from $e^+ - 1$ to e^+ (but we do not need to know them) and all (implicit) keys due to Lemma 2 are distributed as expected. Also, all the tokens $\varDelta_e$ in this window that are given to A are distributed as expected. Finally, we can use the same argumentation as in the proof of Lemma 3 to show all signatures given to the adversary within the window in **Game 2** are distributed identical to the ones in **Game 1**. □

Lemma 6. *For any adversary A it holds* $\Pr[S_{A,2}] \leq \mathsf{Adv}^{\mathsf{euf\text{-}cma}}_{\Sigma,A}(\lambda)$.

Proof. Now we are at the point where we can associate an EUF-CMA challenger for Σ to the keys in time slot e^-. Now, for every signature query for message M and epoch e within the window to the Sig' oracle, we query M to the EUF-CMA challenger of Σ and then execute the remaining parts of the Sig' oracle to adapt the so obtained signature in the forwards direction to obtain the signature for M in epoch e within the window. Now if A eventually outputs a valid forgery $(M^*, \sigma^*_{e^*})$ for epoch e^*, we know that in order to be valid, A did not query message M^* for any epoch within the window (otherwise, since it knows all tokens this would be a trivial forgery). In case $e^* = e^-$, we can directly output $(M^*, \sigma^*_{e^*})$ to the EUF-CMA challenger of Σ. Otherwise, we use $\Sigma.\mathsf{Adapt}$ to adapt the forgery backwards into epoch e^- and then output the message M^* and the adapted signature σ as forgery to the EUF-CMA challenger.

Taking all together this concludes the proof. $\square$

4.2 Message-Independent US from the BLS Signature Scheme

Next, we discuss US schemes that do not require the message in order to update signatures, a feature which we call message-independence (MI). We prove that they are secure US in the conventional sense, i.e., in the model we always have access to the messages and verify their validity and we rather consider MI to be a practical feature in the following sense. In many practical applications, signatures can be verified offline at some point and then when performing (a batch of) updates at a later point in time, one does not need to access all the messages and verify the signatures. This helps to improve the performance of the updating procedure.

In Figure 7, we provide a message-independent US scheme from the BLS signature. In contrast to BLS viewed as a KH signature scheme, where key updates are additive and the next public key is $pk' = pk \cdot \tilde{g}^{\Delta'} = \tilde{g}^{sk+\Delta'}$, we here consider a slight variation where the key update is multiplicative, i.e., $pk' = pk^{\Delta'} = \tilde{g}^{sk\cdot\Delta'}$. While this does not anymore yield a KH signature scheme in the framework of [DS19], due to the absence of the secret to public key homomorphism, it is easy to see that this variant provides an Adapt algorithm satisfying Definition 7, i.e., given a signature $\sigma = H(M)^{sk}$ the update is computed as $\sigma' = \sigma^{\Delta'} = H(M)^{sk\cdot\Delta'}$. It is also easy to see that the BLS scheme with this Adapt algorithm satisfies perfect adaption (cf. Definition 8), where in the definition $\mu(sk)$ is replaced by $\tilde{g}^{sk}$ and $\mu(sk)\cdot\mu(\Delta)$ is replaced by $\tilde{g}^{sk\cdot\Delta}$.

Consequently, we can exactly follow the proof of Theorem 1 with the only exception that we do not use μ for computing the pk's in the window, but choose $\Delta_{e_i} \leftarrow \mathbb{Z}_p$ and compute $pk_{e_i} = pk_{e_{i-1}}^{\Delta_{e_i}}$ (for all $e^- \leq e_i < e^+$). Moreover, it is easy to see that we adapt Lemma 2 with the same changes as discussed above. Checking correctness is straightforward. Hence, we obtain the following:

Corollary 1. *Let* $\Sigma = (\mathsf{Gen}, \mathsf{Sig}, \mathsf{Adapt}, \mathsf{Ver})$ *be the BLS signature scheme with the* Adapt *algorithm defined as* $\left(pk^{\Delta'}, \sigma^{\Delta'}\right) \leftarrow \mathsf{Adapt}(pk, M, \sigma, \Delta')$, *then the*

```
$\mathsf{Setup}(1^\lambda, n)$:
  – Run $\mathsf{BG} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g, \tilde{g}, e, p) \leftarrow \mathsf{BGGen}(1^\lambda)$, choose a hash function $H : \{0,1\}^* \to \mathbb{G}_1$ uniformly at random from hash function family $\{H_k\}_k$.
  – Choose $x \leftarrow \mathbb{Z}_p^*$, and set $sk = x$ and $pk = \tilde{g}^x$.
  – Return $(pk_1, sk_1) \leftarrow (pk, sk)$.
$\mathsf{Next}(pk_e, sk_e)$:
  – Choose $x' \leftarrow \mathbb{Z}_p^*$ and set $\Delta'_{e+1} := x'$ and $\Delta_{e+1} := (\Delta'_{e+1}, pk_e)$.
  – Compute $pk_{e+1} = pk_e^{\Delta_{e+1}}$ and $sk_{e+1} = sk_e \cdot \Delta'_{e+1}$.
  – Return $(pk_{e+1}, sk_{e+1}, \Delta_{e+1})$.
$\mathsf{Sig}(sk_e, M)$:
  – Return $\sigma = H(M)^{sk_e}$.
$\mathsf{Update}(\Delta_{e+1}, \sigma_e)$:
  – Parse $\Delta_{e+1} = (\Delta'_{e+1}, pk_e)$.
  – Compute $\sigma_{e+1} = \sigma_e^{\Delta'_{e+1}}$.
  – Return $\sigma_{e+1}$.
$\mathsf{Ver}(pk_e, M, \sigma_e)$:
  – Return $e(H(M), pk_e) = e(\sigma_e, \tilde{g})$.
```

Fig. 7. US with message-independent updates from BLS signatures.

updatable signature scheme US *from Figure 7 is US-EUF-CMA secure and US-UU-CMA secure.*

MI US from PS signatures. We note that the same technique (i.e., using multiplicative updates to obtain MI) can be applied to other KH signatures, such as Pointcheval-Sanders (PS) [PS16]. More precisely, for PS we can set the public key to $pk := (\tilde{X}, \tilde{Y}) = (\tilde{g}^x, \tilde{g}^y)$ for the secret key $sk := (x, y)$ with $x \leftarrow \mathbb{Z}_p$ and $y \leftarrow \mathbb{Z}_p^*$. The signature is computed by sampling $h \leftarrow \mathbb{G}_1^*$ and setting $\sigma := (\sigma_1, \sigma_2) = (h, h^{x+y\cdot m})$. The verification holds for $e(\sigma_1, \tilde{X} \cdot \tilde{Y}^m) = e(\sigma_2, \tilde{g})$. In order to provide message-independent updates, we can sample $\Delta_1 \leftarrow \mathbb{Z}_p^*$ and $\Delta_2 \leftarrow \mathbb{Z}_p$, set the update token to $\Delta := (\Delta_1, \Delta_2)$, and compute the updated key pair as $sk' := (x \cdot \Delta_1 + \Delta_2, y \cdot \Delta_1)$ and $pk' := (\tilde{X}', \tilde{Y}') = (\tilde{X}^{\Delta_1} \cdot \tilde{g}^{\Delta_2}, \tilde{Y}^{\Delta_1})$. Then, the update procedure is performed by sampling a random $r \leftarrow \mathbb{Z}_p^*$ and computing $\sigma' := (\sigma'_1, \sigma'_2) = (\sigma_1^r, \sigma_2^{r\cdot\Delta_1} \cdot \sigma_1^{r\cdot\Delta_2})$.

4.3 Towards Updatable Signatures from Lattices

In this section, we aim to construct an US scheme from lattices. In particular, we start from the well-known GPV signature scheme [GPV08] and, by using methods inspired by the lattice-based proxy re-signature approach in [FL19], we obtain an US scheme that we call $\mathsf{US}_{\mathsf{GPV}}$. In order to prove its US-EUF-CMA security, however, we have to restrict the capabilities of the adversary, but will provide evidence that this does not seem to make a huge difference in the practical use of the scheme compared to the original leakage profile.

Let us briefly recall the construction of the GPV signature scheme in its probabilistic full-domain hash (FDH) variant. For a recollection of lattice preliminaries we refer the reader to Appendix A. In the following let $H : \{0,1\}^* \to \mathbb{Z}_q^n$ be a hash function modeled as a random oracle. The GPV signature scheme consists of the following algorithms:

- $\mathsf{Gen}(1^n)$: Run $\mathsf{TrapGen}(n, m, q, s)$ to get pair $(\mathbf{A}, \mathbf{T_A})$ ($\mathbf{A}$ is an $n \times m$ matrix over $\mathbb{Z}_q$ and $\mathbf{T_A}$ is a short basis of $\Lambda^{\perp}(\mathbf{A})$). Output $(pk = \mathbf{A}, sk = \mathbf{T_A})$.
- $\mathsf{Sig}(M, sk = \mathbf{T_A})$: Sample $t \leftarrow \{0,1\}^n$, compute $\mathbf{y} = H(M||t)$, and output $(\mathbf{u}, t)$, where $\mathbf{u}$ is a short vector computed via $\mathbf{u} \leftarrow \mathsf{SamplePre}(\mathbf{A}, \mathbf{T_A}, s, \mathbf{y})$.
- $\mathsf{Ver}((\mathbf{u}, t), M, pk = \mathbf{A})$: Compute $\mathbf{y} = H(M||t)$. Output 1 if and only if $\mathbf{A} \cdot \mathbf{u} = \mathbf{y}$ and $\|\mathbf{u}\|$ is small enough, and 0 otherwise.

To transform this scheme into an updatable one, we can apply a method similar to the one used in [FL19] to generate the re-signing keys and re-sign signatures: given the secret key of epoch $e+1$, using the $\mathsf{SamplePre}$ algorithm, it is possible to compute a small norm matrix Δ_{e+1} (for the sake of conciseness, we do not include the old public key as part of the update token) that maps, by left multiplication, the current public key to the previous one, i.e., $pk_{e+1} \cdot \Delta_{e+1} = pk_e$. Then, we can use this matrix to map, by right multiplication, a signature valid in the previous epoch to a signature valid in the current one. The small norm of Δ_{e+1} ensures that, in the update process, the norm of the signature does not increase "too much". Figure 8 describes the so obtained lattice-based US scheme $\mathsf{US_{GPV}}$.

Public parameters: security parameter λ, $T = \mathrm{polylog}(\lambda)$ maximum allowed updates, $q = n^{O(T)}$, $n = \mathrm{poly}(\lambda)$, $m = O(n \log q)$, $s = \omega(\sqrt{\log n})$ and $B = \omega(2^T)$.

```
KeyGen(1^λ):
– Let $(\mathbf{A}_1, \mathbf{T}_{\mathbf{A}_1}) \leftarrow \mathsf{TrapGen}(n, m, q, s)$.
– Return $(sk_1 := \mathbf{T}_{\mathbf{A}_1}, pk_1 := \mathbf{A}_1)$.
Next($pk_e, sk_e$):
– Let $(\mathbf{A}_{e+1}, \mathbf{T}_{\mathbf{A}_{e+1}}) \leftarrow \mathsf{TrapGen}(n, m, q, s)$.
– Let $\Delta_{e+1} \leftarrow \mathsf{SamplePre}(\mathbf{A}_{e+1}, \mathbf{T}_{\mathbf{A}_{e+1}}, s, \mathbf{A}_e)$.
– Return $(sk_{e+1} := \mathbf{T}_{\mathbf{A}_{e+1}}, pk_{e+1} := \mathbf{A}_{e+1}, \Delta_{e+1})$.
Sig($sk_e, M$):
– Sample $t \leftarrow \{0,1\}^k$.
– Compute $\mathbf{y} = H(M||t)$, and $\tau_e \leftarrow \mathsf{SamplePre}(\mathbf{A}_e, \mathbf{T}_{\mathbf{A}_e}, s, \mathbf{y})$.
– Return $\sigma_e = (\tau_e, t)$.
Update($\Delta_{e+1}, \sigma_e$):
– Parse $\sigma_e = (\tau_e, t)$.
– Compute $\tau_{e+1} = \Delta_{e+1} \cdot \tau_e$.
– Return $\sigma_{e+1} = (\tau_{e+1}, t)$.
Ver($pk_e, M, \sigma_e$):
– Parse $\sigma_e = (\tau_e, t)$.
– Compute $\mathbf{y} = H(M||t)$.
– Return 1 if $||\tau_e|| < B$ and $pk_e \cdot \tau_e = \mathbf{y}$, otherwise return $\perp$.
```

Fig. 8. Message-independent unidirectional US from GPV.

Correctness and leakage profile. In the Next algorithm, the token Δ_e is computed by running $\mathsf{SamplePre}(\mathbf{A}_{e+1}, \mathbf{T}_{\mathbf{A}_{e+1}}, s, \mathbf{A}_e)$. In this way we obtain an $m \times m$ matrix Δ_{e+1} (of short norm) such that

$$\mathbf{A}_{e+1} \cdot \Delta_{e+1} = \mathbf{A}_e.$$

If $\sigma_e = (\tau_e, t)$ is a valid signature of M under the public key $\mathbf{A}_e$, we must have that $\mathbf{A}_e \cdot \tau_e = H(M||t)$, and that τ_e is of small norm. When we update the signature σ_e we get $\tau_{e+1} = \Delta_{e+1} \cdot \tau_e$. Since both Δ_e and τ_e are of small norm, so is τ_{e+1}. Moreover

$$\begin{aligned}\mathbf{A}_{e+1} \cdot \tau_{e+1} &= \mathbf{A}_{e+1} \cdot (\Delta_{e+1} \cdot \tau_e) = (\mathbf{A}_{e+1} \cdot \Delta_{e+1}) \cdot \tau_e \\ &= \mathbf{A}_e \cdot \tau_e = H(M||t),\end{aligned}$$

which proves correctness of the updated signature. As a signature produced by algorithm Sign has size $O(s\sqrt{m})$, and after each update, the size grows at the rate of $O(sm)$, as in [FL19], we set the parameter used in verification to be $B = \omega(2^T)$. The US construction can support $T = \text{polylog}(\lambda)$ updates using the subexponential SIS assumption.

Modified leakage profile. As far as the leakage profile is concerned, in order to be able to prove the US-EUF-CMA security of $\mathsf{US}_{\mathsf{GPV}}$, we need to add a restriction that the adversary is not allowed to query the update oracle at epoch $e^- - 1$ to obtain signatures at epoch e^-. We note that this restriction is needed to allow the challenger to simulate all responses to the oracle queries of the adversary. In the general case, this weakened model would allow to prove US schemes US-EUF-CMA which leak the token when seeing a signature and its updated version (such as Schnorr type signatures). However, as proven in Proposition 1 below, even updating a large but limited amount of signatures will not leak the token for the $\mathsf{US}_{\mathsf{GPV}}$ scheme. Consequently, weakening the model merely seems to be an artifact that results from our proof technique, but does not seem to represent a significant weakness in practice.

Moreover, as updated signatures are distinguishable from fresh ones, one has to keep track of the different signatures given to the adversary: for this reason, in the security proof we will split the set $\mathcal{S}$ into sets $\mathcal{S}'$ and $\mathcal{U}'$, which will consist of the fresh and updated signatures respectively. In addition to also supporting the feature of message-independence, interestingly, the $\mathsf{US}_{\mathsf{GPV}}$ scheme satisfies also unidirectional updates: by construction, the secret key of epoch e alone is required to produce the update token Δ_e. In particular this implies that the token cannot be used to backward adapt signatures, since this will contradict the unforgeability of the underlying signature scheme. This "feature" can be seen as one reason for the weakening of the model, as it is incompatible with the proof technique used for the other US constructions.

Security of $\mathsf{US}_{\mathsf{GPV}}$. For this construction, we can prove the following theorem:

Theorem 2. *Assuming the hardness of* $\mathsf{SIS}_{q,n,m,2B}$*, the US scheme* $\mathsf{US}_{\mathsf{GPV}}$ *from Figure 8 is US-EUF-CMA secure, with the above discussed restriction on the adversary, in the random oracle model.*

Proof (Sketch). We can follow, here as well, the proof of Theorem 1: we guess the forgery period e^* and the window $[e^-, e^+[$ (by the above discussion regarding the uni-directionality of the US under consideration, the window will be, in this case, one-sided, i.e., we can even assume that A has access to Δ_{e^+}). Outside of the window, we will behave in our simulation as in the original game and will know all secret keys and update tokens. Inside the window, we start by embedding the SIS matrix $\mathbf{A}^*$ as public key of the forgery epoch e^*. We then have to distinguish the right part of the window, $e \in]e^*, e^+[$, from the left part, $e \in [e^-, e^*[$. For all epochs e in the right part, we can produce pk_e and Δ_e as in the real game (and thus we also have the corresponding secret key sk_e). For those in the left part, we start by sampling $\Delta_{e^*} \leftarrow \mathcal{D}_{\mathbb{Z}^{m\times m},s}$ and set $pk_{e^*-1} := pk_{e^*} \cdot \Delta_{e^*}$. Since this distribution is statistically close to the one of the matrices output by the $\mathsf{SamplePre}$ algorithm, the adversary will not be able to distinguish the simulation from the real game. We then iterate this process till we obtain pk_{e^-}. In this way we can respond to all secret key and token corruption queries. As far as the signature queries are concerned, we rely on the programmability of the random oracle model: when the adversary queries the signature oracle, we first sample the short vector that will serve as signature and then program the random oracle H accordingly. The presence of the "salt" t guarantees us that, except with negligible probability, we will be able to reply to all signing queries (e.g., even if the adversary asks for signature of the same message in different epochs, which would not be possible if there was no "salt" involved in the signing algorithm). The update queries can be answered as in the real game as, by the restriction imposed on the adversary, we will know all the tokens require to run the allowed update queries. Since simulation and real game are computationally indistinguishable, the reduction can derive a SIS solution from the forgery tuple. □

We provide a full proof in Appendix B.

Remark 1. The above $\mathsf{US}_{\mathsf{GPV}}$ scheme does not achieve US-UU-CMA security: Firstly, the tag associated to a signature is not changed during the update and, secondly, the norm of the signature acts as distinguishing feature between fresh and updated signatures.

The following proposition shows that, under the parameter restriction required by the $\mathsf{TrapGen}$ algorithm, i.e., $m \geq 5n \log q$, we can update a large but limited amount of signatures, namely k, without leaking the update token Δ to the adversary.

Proposition 1. *Let $m \geq 5n \log q$ and $k \leq n$. For any PPT adversary $\mathcal{A}$, the probability that $\mathcal{A}$ on input (pk_e, sk_e, pk_{e+1}) and any k pairs of updated signatures $(\tau_{M_i,t_i,e}, \Delta_{e+1} \cdot \tau_{M_i,t_i,e})$ outputs the update token Δ_{e+1} is negligible.*

Proof. By the second claim of Lemma 5.2 from [GPV08], Δ_{e+1} is distributed according to a discrete Gaussian over $\mathbb{Z}^{m\times m}$, which has, by Lemma 2.10 from [GPV08], at least min-entropy $m(m-1)$. By the chain rule of min-entropy, every pair of updated signatures $(\tau_e, \Delta_{e+1} \cdot \tau_e)$ lowers the entropy of Δ_{e+1} by $m \log q$. Hence the min-entropy of Δ_{e+1} conditioned on the view of the adversary is at least $m(m-1) - k \cdot m \log q$, which is greater than n by our bounds on k and m. □

4.4 Overview and Discussion

We provide a compact overview of US schemes obtained from different KH signatures as well as our dedicated BLS-, PS- and GPV-based constructions in Table 1. We present the scheme along with the required hardness assumption, whether it is in the standard model, in the generic group model (GGM) or require random oracles (RO), whether it is unlinkable (UU-CMA), whether it is message-dependent or -independent (MD/MI) and whether it supports an unbounded number of epochs (UB), i.e,. at least polynomially many in the security parameter, or a concrete bound T on the number of updates.

Table 1. Overview of updatable signature schemes.

Scheme	Assumption	Model	UU-CMA	MD/MI	UB
BLS (Sec. 4.2)	co-CDH	RO	✓	MI	✓
BLS (Sec. 4.1)	co-CDH	RO	✓	MD	✓
PS (Sec. 4.2)	P-LRSW	GGM	✓	MI	✓
PS (Sec. 4.1)	P-LRSW	GGM	✓	MD	✓
Waters (Sec. 4.1)	co-CDH	SM	✓	MD	✓
GPV (Sec. 4.3)†	SIS	RO	✗	MI	T

† Provides US-EUF-CMA security only in a weakened model.

As far as efficiency is concerned (counting only expensive operations), and in order to provide some intuition, the BLS construction requires 1 exponentiation for the Next algorithm, while the Update algorithm needs 1 hash to group operation and 1 exponentiation. On the other hand, the message-independent BLS from Section 4.2 requires 1 exponentiation in the Next algorithm and only 1 exponentiation in the Update algorithm. PS requires 2 exponentiations for the Next algorithm, followed by 3 exponentiations in the Update algorithm. On the other hand, message-independent PS from Section 4.2 requires 3 exponentiations for the Next algorithm, followed by 3 exponentiations in Update.

5 Construction of Updatable MACs

In this section we present a generic constructions of UMACs from (almost) key-homomorphic PRFs and then present a dedicated construction of a UMAC from the Naor, Pinkas, and Reingold (NPR) PRF [NPR99].

Before we start, we will discuss a well-known approach to turn a PRF $F : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ into a MAC by setting $\sigma \leftarrow \Pi.\mathsf{Sig}(sk, M) := F(sk, M)$ with the canonical verification that recomputes the tag σ and compares it to the obtained one. Analogously, a KH-PRF gives a KH-MAC due to the key-homomorphism property and security of the PRF. However, if we use an "almost" KH-PRF, then the canonical verification needs to be replaced with an "approximate" canonical verification (i.e., noisy equality check), where the verification involves a metric function (e.g., Euclidean distance) that gives the distance between input tag and recomputed tag, and verification only succeeds if the distance is smaller than some bound δ.

Clearly, the above discussed approach only yields a fixed-length MAC with message space $\mathcal{X}$. If the message space is too small, however, we can use a collision-resistant hash function family (Gen_H, H) with $H : S \times \{0,1\}^* \to \mathcal{X}$ to obtain a variable-length MAC that supports arbitrary length messages by defining $\Pi'.\mathsf{Sig}(sk, M) := F(sk, H(s, M))$. For that construction we can show the following (cf. [Bel06]):

Lemma 7. *If Π is a fixed-length EUF-CMA secure MAC for message space $\mathcal{X}$ and (Gen_H, H) a collision resistant hash function family, then Π' is a variable-length EUF-CMA secure MAC for messages of arbitrary length.*

Proof (Sketch). Let A be the adversary in the experiment $\mathsf{Exp}^{\mathsf{euf\text{-}cma}}_{\Pi,A}$ and let $(M_1, \ldots, M_q)$ be the messages queried by A to oracle Sig' and (M^*, σ^*) be the valid forgery output by the adversary. Now, we have two cases: In the first case $i)$ we have that $H(s, M^*) = H(s, M_i)$ for some $i \in [q]$, which yields a collision pair (M^*, M_i) for H, contradicting collision-resistance of (Gen_H, H). In the second case $ii)$ we have that $H(s, M^*) \neq H(s, M_i)$ for all $i \in [q]$. However, this means that $(H(s, M^*), \sigma^*)$ represents a valid tag (signature) for a new message $H(s, M^*)$ and thus a valid forgery for Π. □

Subsequently, in our generic construction we consider KH PRFs, which we can equivalently view as KH MACs for fixed-length inputs. We will not make it explicit in our construction, but straightforwardly applying Lemma 7 to our generic construction in Section 5.1 will yield UMACs for variable-length inputs.

5.1 UMACs from KH PRFs

Now we show how to obtain UMACs from (almost) KH-PRFs generically. For $\mathcal{K}$ we write $\oplus$ as the group operation and $-k$ as the inverse of k. For the group $\mathcal{Y}$, we use the common addition. The UMAC obtained from a KH-PRF can be seen in Figure 9, where the text in blue color is only required when using “almost” KH-PRF and $\mathcal{D}_\chi$ represents the error distribution (e.g., error distribution used in lattice-based constructions). We can show the following:

Theorem 3. *If $F : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ is a secure (almost) key-homomorphic PRF (equivalently an EUF-CMA secure (approximate) MAC for message space $\mathcal{X}$), then the UMAC construction in Figure 9 is UMAC-EUF-CMA secure and UMAC-UU-CMA secure.*

The proof of UMAC-EUF-CMA security follows exactly the strategy in the proof of Theorem 1 with the only exceptions that within the window we need to simulate the Ver' oracle, and for the almost KH-PRF we need to account for the additional error terms. For completeness, we provide a sketch of the proof in Appendix C.

A note on almost KH-PRFs. In the notion of almost KH-PRFs such as those from the (R)LWE assumption [BLMR13, Kim20, BEKS20] every homomorphic operations increases the error χ and thus the constructions are only T-time

```
Setup$(1^\lambda, n)$:
  – Sample a random key $k_1 \in \mathcal{K}$ and output $k_1$.
Next$(k_e)$ :
  – Sample a random key $k_{e+1} \in \mathcal{K}$.
  – Compute $\Delta_{e+1} = k_{e+1} \oplus -k_e$ and $(k_{e+1}, \Delta_{e+1})$.
Sig$(k_e, M)$ :
  – Sample $(\chi_2, \ldots, \chi_e) \leftarrow_\$ \mathcal{D}_\chi$.
  – Compute $\sigma_e = F(k_e, M) + \sum_{i=2}^{e} \chi_i$ and output $(M, \sigma_e)$.
Update$(\Delta_{e+1}, M, \sigma_e)$ :
  – Compute $\sigma_{e+1} = \sigma_e + F(\Delta_{e+1}, M)$ and output $(M, \sigma_{e+1})$.
Ver$(k_e, M, \sigma_e)$ :
  – If $F(k_e, M) = \sigma_e$ $(\|\sigma_e - F(k_e, M)\| \leq \delta)$ output 1, otherwise output 0.
```

Fig. 9. Bi-directional UMAC from (almost) KH-PRF. Blue parts correspond to the changes when using almost KH-PRF.

correct. This means, that UMACs constructed from such KH-PRFs by default will not satisfy the UMAC-UU-CMA notion, while the tags obtained from the Update algorithm will have higher error compared to fresh tags obtained from the Sig algorithm, thus making them trivially distinguishable. In order to circumvent this issue, in Figure 9, we use the trick to make the error depend on the epoch e that we are in. Hence, freshly computed tags and updated ones have the same amount of error, which makes them indistinguishable, and allows us to achieve the UMAC-UU-CMA notion.

Another issue to consider is the effect of the approximate canonical verification on the security of the UMAC. Since we have a noisy equality check during the verification algorithm, we can consider that we have a ball centered around the tag σ, such that verification accepts any vector within this ball as a valid tag. This implies that an adversary can just change the low-order bits of a valid tag σ to produce another valid tag σ' that will be within this ball and pass the verification, and hence, break the strong unforgeability. However, since in this work we are only interested in conventional unforgeability of the MAC (i.e., do not require strongly unforgeable MACs), this approach is not useful to a valid adversary against our UMAC. The adversary in our case is required to come up with a valid tag that lies sufficiently far away from any tags that it was provided with. Though, the adversary cannot do this due to the security of the underlying KH-PRF. Nevertheless, the security of the KH-MAC obtained from KH-PRF is correlated with the verification bound. If the verification bound is extremely large, then we have that the balls around the valid tags are overlapping (i.e., the balls are so large that they cover the entire space), and then with high probability any random vector is sufficiently close to a random tag. However, by setting the parameters appropriately we can bound this probability to be negligible. More precisely, when using lattice-based almost KH-PRFs as MACs, for a MAC verification bound δ, modulus q and lattice dimension n, we have that the ball around a valid tag σ takes up $(\delta/q)^n$ of the area, where the entire space has area of q^n. If the space taken by the ball is negligible, then it is hard for the adversary to forge a valid tag. Since this will depend on the instantiation and parameters of the almost KH-PRF, we leave it as an open problem to setup tighter bounds and compute exact parameters. For our construction in Figure 9, we can set the verification bound to $\delta = T \cdot B$, for a constant T denoting the maximum number of epochs for our UMAC, and B the bound on the errors sampled from $\mathcal{D}_\chi$.

5.2 Message-Independent UMAC from the NPR PRF

Since UMACs from KH-PRFs are inherently message-dependent, we now present a dedicated construction of a variable-length UMAC scheme that is message-independent (MI) from the PRF due to Naor, Pinkas, and Reingold (NPR) [NPR99]. Let us recall the NPR PRF and therefore let $\mathbb{G}$ be a cyclic group of prime oder p in which the DDH assumption holds and $H : \{0,1\}^* \rightarrow \mathbb{G}$ a hash function modeled as a random oracle, then the NPR PRF with $F : \mathbb{Z}_p^* \times \{0,1\}^* \rightarrow \mathbb{G}$ is defined as $F(k, M) := H(M)^k$. It is secure under the DDH assumption in the random oracle model. In contrast to the key-homomorphic variant of the NPR PRF which considers keys from the additive group $\mathbb{Z}_p$, we consider key updates multiplicatively with $\Delta \in \mathbb{Z}_p^*$, in the vein of the multiplicative variant of the US from the BLS scheme in Section 4.2. Note that as in Section 4.2 we consider MI to be a feature of UMACs for practical applications where one can assume that one operates on valid UMACs.

```
Setup(1^λ, n):
  – Run $\mathcal{G} = (\mathbb{G}, p, g) \leftarrow \mathsf{GGen}(1^\lambda)$, choose a hash function $H : \{0,1\}^* \rightarrow \mathbb{G}_1$
    uniformly at random from hash function family $\{H_k\}_k$.
  – Sample a random key $k_1 \in \mathbb{Z}_p^*$ and return $(\mathcal{G}, H, k_1)$.
Next($k_e$) :
  – Choose $\Delta_{e+1} \leftarrow \mathbb{Z}_p^*$ and return $(k_{e+1} := (\mathcal{G}, H, k_e \cdot \Delta_{e+1}), \Delta_{e+1})$.
Sig($k_e, M$) :
  – Compute $\sigma_e = H(M)^{k_e}$ and return $(M, \sigma_e)$.
Update($\Delta_{e+1}, \sigma_e$) :
  – Compute $\sigma_{e+1} = \sigma_e^{\Delta_{e+1}}$ and return $\sigma_{e+1}$.
Ver($k_e, M, \sigma_e$) :
  – If $H(M)^{k_e} = \sigma_e$ return 1, otherwise return 0.
```

Fig. 10. Bi-directional variable-length UMAC from NPR PRF.

To show the security of this construction, we can exactly follow the proof of Theorem 3 with the only exception that we do not use the key-homomorphic property of the PRF in Update, but choose $\Delta_{e_i+1} \leftarrow \mathbb{Z}_p^*$ and compute $\sigma = \sigma^{\Delta_{e_i+1}^{-1}}$ or $\sigma = \sigma^{\Delta_{e_i+1}}$ if we have to switch PRF evaluations back (from epoch $e_i + 1$ to epoch e_i) or forth (from epoch e_i to epoch $e_i + 1$). Checking correctness is straightforward and we obtain the following:

Corollary 2. *Let F be the NPR PRF, then the construction in Figure 7 is an UMAC-EUF-CMA-secure and UMAC-UU-CMA secure UMAC.*

5.3 Overview and Discussion

We provide a compact overview of UMACs obtained from different KH-PRFs as well as our dedicated NPR-based construction in Table 2. We use the same criteria for comparison as in Section 4.4.

Table 2. Overview of updatable MACs.

Scheme	Assumption	Model	UU-CMA	MD/MI	UB
BLMR (NPR) [BLMR13]	DDH	RO	✓	MD	✓
NPR (Sec.5.2)	DDH	RO	✓	MI	✓
BEKS [BEKS20]	RLWE	RO	✓	MD	T
Kim [Kim20]	LWE	SM	✓	MD	T

Regarding efficiency (again only counting expensive operations), for the key-homomorphic NPR UMAC for instance Update requires 1 hashing to the group as well as 1 exponentiation and Next only cheap operations. The variant of the NPR UMAC from Sec. 5.2 requires instead 1 exponentiation for Update and Next also only cheap operations.

Acknowledgements. We thank the anonymous reviewers for their comments. This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement n°871473 (KRAKEN), European Union's Horizon 2020 ECSEL Joint Undertaking project under grant agreement n°783119 (SECREDAS), by the Austrian Science Fund (FWF) and netidee SCIENCE grant P31621-N38 (PROFET) and FWF grant W1255-N23.

References

[ABK20] Vivek Arte, Mihir Bellare, and Louiza Khati. Incremental cryptography revisited: PRFs, nonces and modular design. In Karthikeyan Bhargavan, Elisabeth Oswald, and Manoj Prabhakaran, editors, *INDOCRYPT 2020*, volume 12578 of *LNCS*, pages 576–598. Springer, Heidelberg, December 2020.

[ACdMT05] Giuseppe Ateniese, Daniel H. Chou, Breno de Medeiros, and Gene Tsudik. Sanitizable signatures. In Sabrina De Capitani di Vimercati, Paul F. Syverson, and Dieter Gollmann, editors, *ESORICS 2005*, volume 3679 of *LNCS*, pages 159–177. Springer, Heidelberg, September 2005.

[ACJ17] Prabhanjan Ananth, Aloni Cohen, and Abhishek Jain. Cryptography with updates. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *EUROCRYPT 2017, Part II*, volume 10211 of *LNCS*, pages 445–472. Springer, Heidelberg, April / May 2017.

[AIK06] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. Computationally private randomizing polynomials and their applications. *Comput. Complex.*, 15(2):115–162, 2006.

[Ajt99] Miklós Ajtai. Generating hard instances of the short basis problem. In Jirí Wiedermann, Peter van Emde Boas, and Mogens Nielsen, editors, *ICALP 99*, volume 1644 of *LNCS*, pages 1–9. Springer, Heidelberg, July 1999.

[App20] Apple. Code signing, 2020. https://developer.apple.com/support/code-signing/.

[Arc20] Arch Linux Wiki. pacman/package signing, 2020. https://wiki.archlinux.org/index.php/Pacman/Package_signing.

[ARS20] Behzad Abdolmaleki, Sebastian Ramacher, and Daniel Slamanig. Lift-and-shift: Obtaining simulation extractable subversion and updatable SNARKs generically. In Jay Ligatti, Xinming Ou, Jonathan Katz, and Giovanni Vigna, editors, *ACM CCS 20*, pages 1987–2005. ACM Press, November 2020.

[BBS98] Matt Blaze, Gerrit Bleumer, and Martin Strauss. Divertible protocols and atomic proxy cryptography. In Kaisa Nyberg, editor, *EUROCRYPT'98*, volume 1403 of *LNCS*, pages 127–144. Springer, Heidelberg, May / June 1998.

[BCC$^+$16] Jonathan Bootle, Andrea Cerulli, Pyrros Chaidos, Essam Ghadafi, and Jens Groth. Foundations of fully dynamic group signatures. In Mark Manulis, Ahmad-Reza Sadeghi, and Steve Schneider, editors, *ACNS 16*, volume 9696 of *LNCS*, pages 117–136. Springer, Heidelberg, June 2016.

[BCN$^+$10] Patrik Bichsel, Jan Camenisch, Gregory Neven, Nigel P. Smart, and Bogdan Warinschi. Get shorty via group signatures without encryption. In Juan A. Garay and Roberto De Prisco, editors, *SCN 10*, volume 6280 of *LNCS*, pages 381–398. Springer, Heidelberg, September 2010.

[BDGJ20] Colin Boyd, Gareth T. Davies, Kristian Gjøsteen, and Yao Jiang. Fast and secure updatable encryption. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part I*, volume 12170 of *LNCS*, pages 464–493. Springer, Heidelberg, August 2020.

[BEKS20] Dan Boneh, Saba Eskandarian, Sam Kim, and Maurice Shih. Improving speed and security in updatable encryption schemes. In Shiho Moriai and Huaxiong Wang, editors, *ASIACRYPT 2020, Part III*, volume 12493 of *LNCS*, pages 559–589. Springer, Heidelberg, December 2020.

[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.

[BFF$^+$09] Christina Brzuska, Marc Fischlin, Tobias Freudenreich, Anja Lehmann, Marcus Page, Jakob Schelbert, Dominique Schröder, and Florian Volk. Security of sanitizable signatures revisited. In Stanislaw Jarecki and Gene Tsudik, editors, *PKC 2009*, volume 5443 of *LNCS*, pages 317–336. Springer, Heidelberg, March 2009.

[BFLS10] Christina Brzuska, Marc Fischlin, Anja Lehmann, and Dominique Schröder. Unlinkability of sanitizable signatures. In Phong Q. Nguyen and David Pointcheval, editors, *PKC 2010*, volume 6056 of *LNCS*, pages 444–461. Springer, Heidelberg, May 2010.

[BGG94] Mihir Bellare, Oded Goldreich, and Shafi Goldwasser. Incremental cryptography: The case of hashing and signing. In Yvo Desmedt, editor, *CRYPTO'94*, volume 839 of *LNCS*, pages 216–233. Springer, Heidelberg, August 1994.

[BGG95] Mihir Bellare, Oded Goldreich, and Shafi Goldwasser. Incremental cryptography and application to virus protection. In *27th ACM STOC*, pages 45–56. ACM Press, May / June 1995.

[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.

[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.

[CDHK15] Jan Camenisch, Maria Dubovitskaya, Kristiyan Haralambiev, and Markulf Kohlweiss. Composable and modular anonymous credentials: Definitions and practical constructions. In Tetsu Iwata and Jung Hee Cheon, editors, *ASIACRYPT 2015, Part II*, volume 9453 of *LNCS*, pages 262–288. Springer, Heidelberg, November / December 2015.

[CMZ14] Melissa Chase, Sarah Meiklejohn, and Greg Zaverucha. Algebraic MACs and keyed-verification anonymous credentials. In Gail-Joon Ahn, Moti Yung, and Ninghui Li, editors, *ACM CCS 2014*, pages 1205–1216. ACM Press, November 2014.

[DS16] David Derler and Daniel Slamanig. Key-homomorphic signatures: Definitions and applications to multiparty signatures and non-interactive zero-knowledge. Cryptology ePrint Archive, Report 2016/792, 2016. https://eprint.iacr.org/2016/792.

[DS18] David Derler and Daniel Slamanig. Highly-efficient fully-anonymous dynamic group signatures. In Jong Kim, Gail-Joon Ahn, Seungjoo Kim, Yongdae Kim, Javier López, and Taesoo Kim, editors, *ASIACCS 18*, pages 551–565. ACM Press, April 2018.

[DS19] David Derler and Daniel Slamanig. Key-homomorphic signatures: definitions and applications to multiparty signatures and non-interactive zero-knowledge. *Des. Codes Cryptogr.*, 87(6):1373–1413, 2019.

[Ele14] Nikolay Elenkov. *Android Security Internals: An In-Depth Guide to Android's Security Architecture*. No Starch Press, 2014.

[EPRS17] Adam Everspaugh, Kenneth G. Paterson, Thomas Ristenpart, and Samuel Scott. Key rotation for authenticated encryption. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part III*, volume 10403 of *LNCS*, pages 98–129. Springer, Heidelberg, August 2017.

[FKM$^+$16] Nils Fleischhacker, Johannes Krupp, Giulio Malavolta, Jonas Schneider, Dominique Schröder, and Mark Simkin. Efficient unlinkable sanitizable signatures from signatures with re-randomizable keys. In Chen-Mou Cheng, Kai-Min Chung, Giuseppe Persiano, and Bo-Yin Yang, editors, *PKC 2016, Part I*, volume 9614 of *LNCS*, pages 301–330. Springer, Heidelberg, March 2016.

[FL19] Xiong Fan and Feng-Hao Liu. Proxy re-encryption and re-signatures from lattices. In Robert H. Deng, Valérie Gauthier-Umaña, Martín Ochoa, and Moti Yung, editors, *ACNS 19*, volume 11464 of *LNCS*, pages 363–382. Springer, Heidelberg, June 2019.

[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.

[GKM$^+$18] Jens Groth, Markulf Kohlweiss, Mary Maller, Sarah Meiklejohn, and Ian Miers. Updatable and universal common reference strings with applications to zk-SNARKs. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part III*, volume 10993 of *LNCS*, pages 698–728. Springer, Heidelberg, August 2018.

[Goo19] Google Developers. Sign your app, 2019. https://developer.android.com/studio/publish/app-signing.

[GPV08] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In Richard E. Ladner and Cynthia Dwork, editors, *40th ACM STOC*, pages 197–206. ACM Press, May 2008.

[IK00] Yuval Ishai and Eyal Kushilevitz. Randomizing polynomials: A new representation with applications to round-efficient secure computation. In *41st FOCS*, pages 294–304. IEEE Computer Society Press, November 2000.

[Jia20] Yao Jiang. The direction of updatable encryption does not matter much. In Shiho Moriai and Huaxiong Wang, editors, *ASIACRYPT 2020, Part III*, volume 12493 of *LNCS*, pages 529–558. Springer, Heidelberg, December 2020.

[JKR19] Stanislaw Jarecki, Hugo Krawczyk, and Jason K. Resch. Updatable oblivious key management for storage systems. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *ACM CCS 2019*, pages 379–393. ACM Press, November 2019.

[JMM19] Daniel Jost, Ueli Maurer, and Marta Mularczyk. Efficient ratcheting: Almost-optimal guarantees for secure messaging. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part I*, volume 11476 of *LNCS*, pages 159–188. Springer, Heidelberg, May 2019.

[JMSW02] Robert Johnson, David Molnar, Dawn Xiaodong Song, and David Wagner. Homomorphic signature schemes. In Bart Preneel, editor, *CT-RSA 2002*, volume 2271 of *LNCS*, pages 244–262. Springer, Heidelberg, February 2002.

[JS18] Joseph Jaeger and Igors Stepanovs. Optimal channel security against fine-grained state compromise: The safety of messaging. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part I*, volume 10991 of *LNCS*, pages 33–62. Springer, Heidelberg, August 2018.

[Kim20] Sam Kim. Key-homomorphic pseudorandom functions from LWE with small modulus. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part II*, volume 12106 of *LNCS*, pages 576–607. Springer, Heidelberg, May 2020.

[KLR19] Michael Klooß, Anja Lehmann, and Andy Rupp. (R)CCA secure updatable encryption with integrity protection. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part I*, volume 11476 of *LNCS*, pages 68–99. Springer, Heidelberg, May 2019.

[Kra05] Martin F. Krafft. *The Debian System: Concepts and Techniques*. No Starch Press Series. No Starch Press, 2005.

[Lip20] Helger Lipmaa. Key-and-argument-updatable QA-NIZKs. In Clemente Galdi and Vladimir Kolesnikov, editors, *SCN 20*, volume 12238 of *LNCS*, pages 645–669. Springer, Heidelberg, September 2020.

[LSW10] Hans Löhr, Ahmad-Reza Sadeghi, and Marcel Winandy. Patterns for secure boot and secure storage in computer systems. In *ARES*, pages 569–573. IEEE Computer Society, 2010.

[LT18] Anja Lehmann and Björn Tackmann. Updatable encryption with post-compromise security. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part III*, volume 10822 of *LNCS*, pages 685–716. Springer, Heidelberg, April / May 2018.

[Mic20] Microsoft. Sign a windows 10 app package, 2020. https://docs.microsoft.com/en-us/windows/msix/package/signing-package-overview.

[MNT06] Einar Mykletun, Maithili Narasimha, and Gene Tsudik. Authentication and integrity in outsourced databases. *TOS*, 2(2):107–138, 2006.

[NPR99] Moni Naor, Benny Pinkas, and Omer Reingold. Distributed pseudo-random functions and KDCs. In Jacques Stern, editor, *EUROCRYPT'99*, volume 1592 of *LNCS*, pages 327–346. Springer, Heidelberg, May 1999.

[PS16] David Pointcheval and Olivier Sanders. Short randomizable signatures. In Kazue Sako, editor, *CT-RSA 2016*, volume 9610 of *LNCS*, pages 111–126. Springer, Heidelberg, February / March 2016.

[Red18] Red Hat. How to sign rpms with gpg, 2018. https://access.redhat.com/articles/3359321.

[San20] Olivier Sanders. Efficient redactable signature and application to anonymous credentials. In Aggelos Kiayias, Markulf Kohlweiss, Petros Wallden, and Vassilis Zikas, editors, *PKC 2020, Part II*, volume 12111 of *LNCS*, pages 628–656. Springer, Heidelberg, May 2020.

[SBZ02] Ron Steinfeld, Laurence Bull, and Yuliang Zheng. Content extraction signatures. In Kwangjo Kim, editor, *ICISC 01*, volume 2288 of *LNCS*, pages 285–304. Springer, Heidelberg, December 2002.

[WG18] Grisha Weintraub and Ehud Gudes. Data integrity verification in column-oriented nosql databases. In *DBSec*, volume 10980 of *LNCS*, pages 165–181. Springer, 2018.

[WLX$^+$19] Haoyu Wang, Hongxuan Liu, Xusheng Xiao, Guozhu Meng, and Yao Guo. Characterizing android app signing issues. In *ASE*, pages 280–292. IEEE, 2019.

[ZRAA10] Yupu Zhang, Abhishek Rajimwale, Andrea C. Arpaci-Dusseau, and Remzi H. Arpaci-Dusseau. End-to-end data integrity for file systems: A ZFS case study. In *FAST*, pages 29–42. USENIX, 2010.

Appendix

A Lattice Preliminaries

An n-dimensional lattice Λ is a discrete additive subgroup of $\mathbb{R}^n$. A lattice has rank $k \leq n$ if it is generated as the set of all $\mathbb{Z}$-linear combinations of some k linearly independent basis vectors. We say Λ is full-rank if $k = n$. For a positive semidefinite matrix $\Sigma \in \mathbb{R}^{n \times n}$, we define the continuous Gaussian distribution D_Σ as the probability distribution over $\mathbb{R}^n$ whose density function is proportional to $\rho_\Sigma(\mathbf{x}) = \exp(-\pi \mathbf{x}^t \Sigma^{-1} \mathbf{x})$. For a (full-rank) lattice $\Lambda \subset \mathbb{R}^n$, we define $\rho_\Sigma(\Lambda) = \sum_{\mathbf{x} \in \Lambda} \rho_\Sigma(\mathbf{x})$. Then, the discrete Gaussian distribution over Λ of covariance parameter Σ is defined as $D_{\Lambda,\Sigma} = \rho_\Sigma(\mathbf{x})/\rho_\Sigma(\Lambda)$.

We now recall the Short Integer Solution (SIS) problem, upon which the security of the GPV signature scheme [GPV08] is based.

Definition 14 (SIS). *Let $n, m, q, B \in \mathbb{N}$ be positive integers. For a given adversary A, we define the following experiment:*

- *The challenger samples $\mathbf{A} \leftarrow \mathbb{Z}_q^{n \times m}$, and gives $\mathbf{A}$ to the adversary A.*
- *The adversary A outputs some non-zero vector $\mathbf{z} \in \mathbb{Z}_q^m$.*

We define A's advantage in solving the SIS *problem for the set of parameters n, m, q, B, denoted $\mathsf{SISAdv}_{n,m,q,B}[A]$, to be the probability that $\mathbf{Az} = \mathbf{0} \mod q$ and $||\mathbf{z}|| \leq B$.*

The following proposition specifies the properties of the GPV's KeyGen algorithm.

Proposition 2 ([Ajt99]). *For any prime $q = \mathrm{poly}(n)$ and any $m \geq 5n \log q$, there is a probabilistic polynomial-time algorithm that, on input 1^n, outputs a matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ and a full-rank set $\mathbf{S} \subset \Lambda^\perp(\mathbf{A})$, where the distribution of $\mathbf{A}$ is statistically close to uniform over $\mathbb{Z}_q^{n \times m}$ and the length of $||\mathbf{S}|| \leq L = m^{2.5}$. Moreover, the set $\mathbf{S}$ can be converted efficiently to a "good" basis $\mathbf{T}$ of $\Lambda^\perp(\mathbf{A})$ such that $||\tilde{\mathbf{T}}|| \leq ||\tilde{\mathbf{S}}|| \leq L$.*

The following theorem describes the features of the GPV's Sign algorithm.

Theorem 4 ([GPV08]). *There is a probabilistic polynomial-time algorithm that, given a basis $\mathbf{B}$ of an n-dimensional lattice $\Lambda = \mathcal{L}(\mathbf{B})$, a parameter $s \geq ||\tilde{\mathbf{B}}|| \cdot \omega(\sqrt{\log n})$, and a center $\mathbf{c} \in \mathbb{R}^n$, outputs a sample from a distribution that is statistically close to $D_{\Lambda,s,\mathbf{c}}$.*

The next sequence of lemmas describe the properties of the discrete Gaussian which are necessary in the proof simulation.

Lemma 8 ([GPV08]). *For any m-dimensional lattice Λ, center $\mathbf{c} \in \mathbb{R}^m$, positive $\epsilon > 0$, and $s \geq 2\eta_\epsilon(\Lambda)$ and for every $\mathbf{x} \in \Lambda$, we have*

$$D_{\Lambda,s,\mathbf{c}}(\mathbf{x}) \leq \frac{1+\epsilon}{1-\epsilon} \cdot 2^{-m}.$$

In particular, for $\epsilon < \frac{1}{3}$, the min-entropy of $D_{\Lambda,s,\mathbf{c}}(\mathbf{x})$ is at least $m-1$.

In other words, the probability that an adversary A correctly guesses $\mathbf{x} \leftarrow D_{\Lambda,s,\mathbf{c}}$, given $\mathbf{c}$, is less than $2^{-(m-1)}$.

Lemma 9 ([GPV08]). *Let n and q be positive integers with q prime, and let $m \geq 2n \log q$. Then, for all but a $2q^{-n}$ fraction of all $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ and for any $s \geq \eta_\epsilon(\Lambda^\perp(\mathbf{A}))$, with $\epsilon \in (0, \frac{1}{2})$, the distribution of the syndrome $\mathbf{u} = \mathbf{A}\mathbf{e} \mod q$ is statistically close to uniform over $\mathbb{Z}_q^n$, where $\mathbf{e} \sim D_{\mathbb{Z}^m,s}$.*

The above lemma justifies the use of a random oracle to simulate the hash of the messages and the one below proves that sampling from a discrete Gaussian over the lattice $\Lambda_{\mathbf{t}}^\perp(\mathbf{A}) = \{\mathbf{x} \in \mathbb{Z}_q^m \mid \mathbf{A}\mathbf{x} = \mathbf{t} \mod q\}$, is the same as sampling from the discrete Gaussian over $\mathbb{Z}_q^m$ conditioned on that the sampled vector $\mathbf{x}$ satisfies $\mathbf{A}\mathbf{x} = \mathbf{t} \mod q$.

Lemma 10 ([GPV08]). *Assume the columns of $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ generate $\mathbb{Z}_q^n$, and let $\epsilon \in (0, \frac{1}{2})$ and $s \geq \eta_\epsilon(\Lambda^\perp(\mathbf{A}))$. Then, for $\mathbf{e} \sim D_{\mathbb{Z}^m,s}$, the distribution of the syndrome $\mathbf{u} = \mathbf{A}\mathbf{e} \mod q$ is within statistical distance 2ϵ of uniform over $\mathbb{Z}_q^n$. Furthermore, fix $\mathbf{u} \in \mathbb{Z}_q^n$ and let $\mathbf{t} \in \mathbb{Z}^m$ be an arbitrary solution to $\mathbf{A}\mathbf{t} = \mathbf{u} \mod q$. Then, the conditional distribution of $\mathbf{e} \sim D_{\mathbb{Z}^m,s}$ given $\mathbf{A}\mathbf{e} = \mathbf{u} \mod q$ is exactly $\mathbf{t} + D_{\Lambda^\perp,s,-\mathbf{t}}$.*

Finally, we recall the definition of collision-resistant preimage sampleable function (PSFs) and the fact the algorithms of the GPV signatures scheme provide a PSFs instantiation.

Definition 15 ([GPV08]). *A collection of collision-resistant preimage sampleable functions (PSFs) is a tuple of probabilistic polynomial-time algorithms* (TrapGen, SampleDom, SamplePre) *satisfying the following properties:*

1. TrapGen(1^n) *outputs (a, t), where a is the description of an efficiently-computable function $f_a \colon D_n \to R_n$ (for some efficiently recognizable domain D_n and range R_n depending on n), and t is some trapdoor information for f_a;*
2. SampleDom(1^n) *samples an x from some distribution over D_n, for which the distribution of $f_a(x)$ is uniform over R_n;*
3. *for every $y \in R_n$,* SamplePre(a, t, y) *samples from the conditional distribution $x \leftarrow$* SampleDom(1^n)*, given $f_a(x) = y$;*
4. *for any PPT algorithm A, the probability that $A(1^n, a, y) \in f_a^{-1}(y) \subset D_n$ is negligible, where the probability is taken over the choice of a, the target value $y \leftarrow R_n$ chosen uniformly at random, and A's random coins;*

5. *for every $y \in R_n$, the conditional min-entropy of $x \leftarrow \widetilde{\mathsf{SampleDom}}(1^n)$ given $f_a(x) = y$ is at least $\omega(\log n)$;*
6. *for any PPT algorithm A, the probability that $A(1^n, a)$ outputs distinct $x, x' \in D_n$ such that $f_a(x) = f_a(x')$ is negligible, where the probability is taken over the choice of a and A's random coins.*

A collection of PSFs based on the average-case hardness of SIS can be constructed as follows. Let q, m, and L be as in Proposition 2 and $s \geq L \cdot \omega(\sqrt{\log n})$.

– The function generator uses the algorithms from Proposition 2 to choose $(\mathbf{A}, \mathbf{T})$, where $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ is statistically close to uniform and $\mathbf{T} \subset \Lambda^{\perp}(\mathbf{A})$ is a good basis with $||\tilde{\mathbf{T}}|| \leq L$. The matrix $\mathbf{A}$ (and q) defines the function $f_{\mathbf{A}}(\cdot)$, and the good basis $\mathbf{T}$ is its trapdoor.
– The function $f_{\mathbf{A}}$ is defined as $f_{\mathbf{A}}(\mathbf{e}) = \mathbf{A}\mathbf{e} \mod q$, with domain $D_n = \{\mathbf{e} \in \mathbb{Z}^m \colon ||\mathbf{e}|| \leq s\sqrt{m}\}$ and range $R_n = \mathbb{Z}_q^n$. The input distribution is $D_{\mathbb{Z}^m,s}$, which can be sampled using $\mathsf{SampleDom}$ from [GPV08], with the standard basis of $\mathbb{Z}^m$.
– The trapdoor inversion algorithm $\mathsf{SamplePre}(\mathbf{A}, \mathbf{T}, s, \mathbf{u})$ samples from $f_{\mathbf{A}}^{-1}(\mathbf{u})$ as follows: it computes via linear algebra $\mathbf{t} \in \mathbb{Z}^m$ such that $\mathbf{A}\mathbf{t} = \mathbf{u} \mod q$. Then, samples $\mathbf{v} \sim D_{\Lambda^{\perp},s,-\mathbf{t}}$ using the trapdoor $\mathbf{T}$ and output $\mathbf{e} = \mathbf{t} + \mathbf{v}$.

Theorem 5 ([GPV08]). *The algorithms described above give a collection of collision-resistant PSFs if $\mathsf{SIS}_{q,n,m,2s\sqrt{m}}$ is hard.*

B Proof of Lattice Construction

B.1 Preliminaries

Before presenting the proof of Theorem 2, we need to adapt the definition of "window" from Lemma 1 to the leakage profile under consideration here.

Lemma 11. *Let A be a valid adversary that produces a forgery in epoch $0 < e^* < e_{\max}$ in the US-EUF-CMA experiment with respect to the weakened leakage profile of $\mathsf{US}_{\mathsf{GPV}}$ described in Sec. 4.3, then there exists a maximum integer $0 < e^- \leq e^*$ and a minimum integer $e^* < e^+ \leq e_{\max}$ s.t. A*

1) does not obtain the token Δ_{e^-},
2) obtains no secret key sk_e for all $e^- \leq e < e^+$, and
3) can obtain all tokens Δ_e for $e^- < e < e^+$,

from the queries made to the oracles, given the leakage profile. Subsequently, we often denote the interval $[e^-, e^+[$ as the window.

The proof of the above lemma is identical to the one for Lemma 1, as far as e^- is concerned. Regarding e^+, the unidirectionality of the $\mathsf{US}_{\mathsf{GPV}}$ construction implies that, for any epoch e, the knowledge of sk_e alone allows to obtain Δ_e, hence property 1) has to be changed accordingly, i.e., we do not require any more that the adversary A does not obtain Δ_{e^+}.

B.2 Proof of Theorem 2

We show now that, if there exists an adversary A that breaks the security as defined in Definition 11 with probability ϵ, then it can be used to break the underlying $\mathsf{SIS}_{n,m,q,2B}$ assumption with probability $< e_{max}^2(\epsilon + \text{negl}(\lambda))$, which suffices to prove the theorem. The reduction is invoked on an instance of the SIS assumption, namely $\mathbf{A}^* \in \mathbb{Z}_q^{n\times m}$, and is asked to return a solution $\mathbf{e} \in \mathbb{Z}_q^m$, such that $\mathbf{A}^* \cdot \mathbf{e} = 0 \mod q$ and $0 \neq ||\mathbf{e}|| < 2B$. We again proceed via a sequence of games.

Game 0. This is the experiment $\mathsf{Exp}_{\mathsf{US},A}^{\mathsf{us\text{-}euf\text{-}cma}}(\lambda, n)$.

Game 1. This is like the previous game with the exception that we guess the left side e^- of the window, and the epoch e^* for which A outputs the forgery and abort if our guess is incorrect.

Game 2. This game is like the previous one, but with the following differences. We embed the SIS instance $\mathbf{A}^*$ as the public key of the forgery period e^*, i.e., we set $pk_{e^*} := \mathbf{A}^*$. We keep track of the queried hashes using a list $\mathcal{H}$, which is initially empty, and simulate the next, corrupt, hash, sign and update oracles as follows:

- For call to Next' in epoch $e \in \{e^-, \cdots, e^* - 1\}$ we proceed as follow. As already stated, the public key of epoch e^* is given by the SIS challenge $\mathbf{A}^*$. We then sample $\varDelta_{e^*} \leftarrow \mathcal{D}_{\mathbb{Z}_s^{m\times m}}$ and set $pk_{e^*-1} := pk_{e^*} \cdot \varDelta_{e^*}$. We iterate this process till we obtain pk_{e^-} (we implicitly set $\varDelta_{e^-} = \perp$).
- For each call to the Next' oracle for epoch $e \in \{1, \dots, e^- - 1\} \cup \{e^*, \dots, e_{max} - 1\}$, we run $(sk_{e+1}, pk_{e+1}) \leftarrow \varSigma.\mathsf{Gen}(1^\lambda)$ to obtain an independent key for epoch $e+1$. The update token $\varDelta_{e+1}$ can be computed as in the real scheme by knowing only sk_{e+1} (and pk_e). In this way we can respond to all Next', Update' and $\mathsf{Corrupt}$ queries at epoch e.
- Whenever the adversary queries H on (M, t), we check if $(M, t, \cdot)$ is in $\mathcal{H}$. If so, we return $\mathbf{y}$ such that $(M, t, \mathbf{y}) \in \mathcal{H}$. Otherwise, we let $\tau_{e^*,M,t} \leftarrow \mathsf{SampleDom}(1^n)$, add $(M, t, \mathbf{y})$ to $\mathcal{H}$, where $\mathbf{y} = \mathbf{A}^* \cdot \tau_{e^*,M,t}$, add $(e^*, M, \sigma = (\tau_{e^*,M,t}, t))$ to $\mathcal{S}'$ and return $\mathbf{y}$ to the adversary.
- Whenever the adversary queries the signing oracle Sig' on message M at epoch e, we sample a random tag $t \leftarrow_\$ \{0,1\}^\lambda$ and proceed as follows:
  - if (e, M, σ) is already in $\mathcal{S}'$ for some $\sigma = (\tau, t)$, we return σ to the adversary;
  - else if $(\cdot, M, \sigma)$ is in $\mathcal{S}'$ with $\sigma = (\cdot, t)$, then we abort;
  - else if $e \in \{e^-, \dots, e^*\}$, we let $\tau_{e,M,t} \leftarrow \mathsf{SampleDom}(1^n)$, add $(M, t, \mathbf{y} = pk_e \cdot \tau_{e,M,t})$ to $\mathcal{H}$ and $(e, M, \sigma = (t, \tau_{e,M,t}))$ to $\mathcal{S}'$ and return it to the adversary.
  - else, we sample $\tau_{e^*,M,t} \leftarrow \mathsf{SampleDom}(1^n)$ and add $(M, t, \mathbf{y} = pk_e^* \cdot \tau_{e^*,M,t})$ to $\mathcal{H}$. Then, using sk_e, we compute $\tau_{e,M,t}$ such that $pk_e \cdot \tau_{e,M,t} = \mathbf{y}$ $(= pk_e^* \cdot \tau_{e^*,M,t})$, as in the real scheme, add $(e, M, \sigma := (\tau_{e,M,t}, t))$ to $\mathcal{S}'$ and return σ to the adversary.
- Whenever the adversary queries the update oracle Update' for signature (M, σ_{e-1}), at epoch $e - 1 \neq e^- - 1$ (recall that $e^- - 1$ is the only epoch

for which the adversary is not allowed to query the updated oracle by the given weakened leakage profile), parse $\sigma_{e-1} = (\tau_{e-1,M,t}, t)$ and verify that σ_{e-1} is a valid signature for message M with "salt" t at epoch $e-1$, i.e., that $pk_{e-1} \cdot \tau_{e-1,M,t} = \mathbf{y}$ for $(M, t, \mathbf{y}) \in \mathcal{H}$, and that $||\tau_{e-1,M,t}|| \leq B$. If any of these checks does not pass, then return $\perp$. Otherwise, add $(e, M, \sigma_e := (\tau_{e,M,t} := \varDelta_{e-1} \cdot \tau_{e-1,M,t}, t))$ to $\mathcal{S}'$ and return σ_e to the adversary (notice that this time the updated signatures are only added to the list $\mathcal{U}'$: we have to distinguish between updated and fresh signatures as they are not indistinguishable. We need to maintain a list with only the fresh signatures in order to be able to answer the adversary's queries to the Sig' oracle consistently with the real game).

Now, let us analyze the transitions:

Lemma 12. *For any adversary A it holds that*

$$\left(\frac{1}{e_{max}(e_{max}-1)}\right) \Pr[S_{A,0}] \leq \Pr[S_{A,1}].$$

Proof. We guess the window and the forgery epoch by simply sampling $e^- \leftarrow \{0, ..., e_{max}-1\}$ and $e^* \leftarrow \{e^-, ..., e_{max}\}$ uniformly at random. Thus, this guess is correct with probability at least $\frac{1}{e_{max}(e_{max}-1)}$ and if the guess turns out to be wrong, we abort. Note that such a window always exists for a valid adversary A due to Lemma 11. □

Lemma 13. *For any adversary A it holds that*

$$|\Pr[S_{A,1}] - \Pr[S_{A,2}]| \leq \frac{(Q_{hash} + Q_{sign})^2}{2^k}.$$

Proof. We observe that due to its validity, by Lemma 11, the adversary A

1) does not obtain the token $\varDelta_{e^-}$,
2) obtains no secret key sk_e for all $e^- \leq e < e^+$, and
3) can obtain all tokens $\varDelta_e$ for $e^- < e < e^+$,

from the queries made to the oracles, given the leakage profile. Note that due to 1) we know that there implicitly exists a token mapping keys and signatures from $e^- - 1$ to e^-, (but we do not need to know it since the adversary will never query the update oracle in epoch $e^- - 1$). By Theorem 4 and Lemma 9, all public keys and tokens given to A are distributed as expected. The only difference between **Game 2** and **Game 1** occurs if we abort during a query to Sig'. Such abort occurs when a newly sampled salt t for a message M is equal to an already used salt for the same message. Disregarding the associated message M, we can bound the salt's collision probability by $(Q_{hash} + Q_{sign})^2/2^k$, where k is the salt's bit-length and Q_{hash} and Q_{sign} represent the total hash and sign queries respectively, made by the adversary. This proves the claim. □

Lemma 14. *For any adversary A it holds that*

$$|\Pr[S_{A,2}] - \mathsf{SISAdv}_{q,n,m,2B}[A]| \leq \text{negl}(\lambda).$$

Proof. Suppose A outputs a valid forgery (e^*, M^*, σ^*). Because $\sigma^* = (\tau^*, t^*)$ is a valid forgery of M^* at epoch e^*, we have

$$pk_{e^*}\tau^* = \mathbf{A}^*\tau^* = H(M^* \,||\, t^*) \mod q, \quad \text{and} \quad 0 \neq ||\tau^*|| \leq B.$$

It remains to check that $\tau^* \neq \tau_{e^*,M^*,t^*}$ (the value of the random oracle on input $(M^*||t^*)$ must be know by the adversary, and hence, present in the list $\mathcal{H}$). Moreover, since the forgery produces by the adversary is valid, it must be the case that A did not obtain any signature for the message M^* with salt t^* for epochs $e \in \{e^-, \dots, e^*\}$. If that is the case, then we have a solution to the $\mathsf{SIS}_{q,n,m,2B}$ problem as

$$A^*(\tau^* - \tau_{e^*,M^*,t^*}) = \mathbf{0} \mod q, \quad \text{and} \quad 0 \neq ||\tau^* - \tau_{e^*,M^*,t^*}|| \leq 2B.$$

Therefore we have that:

- A received $H(M^*||t^*) = pk_e^* \cdot \tau_{e^*,M^*,t^*}$ for $\tau_{e^*,M^*,t^*} \leftarrow \mathsf{SampleDom}(1^n)$ but not τ_{e^*,M^*,t^*}. By the preimage min-entropy property of the hash family (Definition 15 and Theorem 5), the min-entropy of τ_{e^*,M^*,t^*} given $f_{pk_{e^*}}(\tau_{e^*,M^*,t^*})$ is at least $\omega(\log n)$. Thus the signature $\tau^* \neq \tau_{e^*,M^*,t^*}$, except with negligible probability $2^{-\omega(\log n)}$.

This concludes the proof. □

C Proof of Theorem 3

Proof (Sketch). We give the proof sketch here for the almost KH-PRF case, as the case of KH-PRF with perfect correctness follows trivially without the additional complications that arise due to the handling of the error terms and noisy equality check. First we observe that the correctness follows straightforwardly from the construction. Secondly, in our construction the errors induced by the almost KH-PRF are epoch-dependent. More precisely, for any message $M \in \mathcal{M}$, key and token $k_e, \Delta_e \in \mathcal{K}$, we have that $\sigma_e = F(k_e, M) + \sum_{i=2}^{e} \chi_i$ and $\sigma'_e = \sigma_{e-1} + F(\Delta_e, M)$ with $\chi_i \leftarrow_{\$} \mathcal{D}_\chi$ are identically distributed, as well as $k' \leftarrow_{\$} \mathcal{K}$ and $k_e \oplus \Delta$ with $\Delta \leftarrow_{\$} \mathcal{K}$ are identically distributed and latter always yields a valid key in $\mathcal{K}$. Consequently, for the challenge tag σ_b the outputs of $\mathsf{UMAC.Sig}$ and $\mathsf{UMAC.UpdateCh}$ are identically distributed and $\mathsf{Adv}^{\mathsf{umac\text{-}uu\text{-}cma}}_{\mathsf{UMAC},A}(\lambda, n) = 0$.

We can prove the UMAC-EUF-CMA security by relying on the EUF-CMA security of the underlying (approximate) KH-MAC Π obtained from the (almost) KH-PRF F. Though, we note that in order for the reduction to go through we require that the approximate KH-MAC Π has a verification bound that is at least twice that of the UMAC, i.e., $\delta_\Pi \geq 2\delta_{\mathsf{UMAC}}$. The reason for this is that later when we do the backwards adaption using the key-homomorphism property of the PRF we increase the error term with every adaption. However, the error can increase at most by δ_{UMAC} (assuming the forgery happens at the last epoch and needs to be backwards adapted to the first epoch), hence, setting $\delta_\Pi = 2\delta_{\mathsf{UMAC}}$ suffices for our security proof.

Then, we proceed exactly as in the proof of Theorem 1 to guess our window $[e^-, e^+[$ and associate an EUF-CMA of an (approximate) KH-MAC Π to the target epoch e^* (also we can use the optimization using the key insulation technique from Klooß et al. [KLR19] and associate it to e^-). Outside of the window, i.e., for epochs up to $e^- - 1$ and starting from e^+ upwards, we will behave in our simulation as in the original game and in particular choose and know all secret keys and update tokens. Again, for all epochs inside the window we do not know the secret keys associated to the epochs, but they are implicitly set by choosing for every epoch e_i in the window a random token Δ_{e_i} as in the real Next algorithm. Thus, the secret keys are implicitly set as $k_{e_i} = k_{e_{i-1}} \oplus \Delta_{e_i}$ for $e^- < e_i < e^+$. Now for any Sig' query for message M and epoch e_i within the window, we query M to the EUF-CMA challenger of Π associated to e^-, and then, using the key-homomorphism property of the underlying PRF we adapt in the forwards direction to obtain the tag for M in epoch e_i within the window, and furthermore, we add additional error terms sampled from $\mathcal{D}_\chi$ to make the error in the tag proportional to the epoch e_i. Also, for any query to Ver' within the window we use the same strategy and call Ver of our EUF-CMA challenger. Note that due to the increased error the distribution of tags from Sig or those adapted to the respective epoch only differ by at most δ_{UMAC}, and hence, the answers from the Ver' oracle are consistent. The Update' oracle is performed as in the original game for all the epochs where the update token is knows. For the remaining epochs, i.e., e^- and e^+, when asked to update $(M, \sigma_{e^- -1})$ (or $(M, \sigma_{e^+ -1})$, respectively), we query M to the EUF-CMA challenger of Π associated to e^- (or produce a fresh signature using k_{e^+}, respectively), adjust the error term in the output tag to match the epoch e^- (or e^+, respectively), and return it to the adversary. Again by the added artificial error terms this is indistinguishable. Now if A outputs a valid forgery $(M^*, \sigma^*_{e^*})$ for epoch e^*, if $e^* = e^-$ we can directly output it. Otherwise, we use the key-homomorphism property of the underlying PRF to adapt the forgery backwards into epoch e^- and output it. Note that in any case a valid forgery output by A represents a valid forgery for the challenger of Π, as validity guarantees that we have never queried M^* throughout the game for any epoch inside the window. Furthermore, the valid forgery needs to pass the verification with bound δ_{UMAC}, and the backwards adaption can increase the error term of the forged tag by at most another δ_{UMAC}. However, since we set the verification bound of Π to $\delta_\Pi = 2\delta_{\mathsf{UMAC}}$, it means that the backwards adapted forgery is a valid forgery for the challenger of Π as well. This concludes the proof. □