DSKE: Digital Signature with Key Extraction

Orestis Alpos*[1], Zhipeng Wang*[2], Alireza Kavousi[3], Sze Yiu Chau[4], Duc V. Le[5], and Christian Cachin[1]

[1] *University of Bern, {orestis.alpos, christian.cachin}@unibe.ch*
[2] *Imperial College London, zhipeng.wang20@imperial.ac.uk*
[3] *University College London, alireza.kavousi.21@ucl.ac.uk*
[4] *The Chinese University of Hong Kong, sychau@ie.cuhk.edu.hk*
[5] *VISA Research, duc.le@visa.com*

Abstract

In general, digital signatures can be used to prove authenticity for as long as the signature scheme is not broken and the private key is kept secret. While this "long-lived" authenticity might be useful in some scenarios, it is inherently undesirable for certain types of sensitive communication, for instance, whistleblowing. A particular concern in this case is that the communication could be leaked in the future, which might lead to potential retaliation and extortion. This calls for a scheme that lets signers prove authenticity for a limited period of time, while allowing them to deny having signed any messages afterwards. We argue that such a scheme could offer a desirable degree of protection to signers through deniability against future leaks, while reducing the incentives for criminals to obtain leaked communications for the sole purpose of blackmailing.

This paper introduces the concept of DSKE, digital signatures with key extraction. In a DSKE scheme, the secret key can be extracted if more than a threshold of signatures on arbitrary messages are ever created. Hence, it provides signers with plausible deniability, by demonstrating a group of recipients that can collectively extract the private key, while, within the threshold, each signature still proves authenticity. We give a formal definition of DSKE, as well as two provably secure constructions, one from hash-based digital signatures and one from polynomial commitments. We show that, in applications where a signer is expected to create a number of signatures, DSKE offers deniability for free. Moreover, DSKE can be employed to disincentivize malicious behavior, such as equivocation and double-signing.

Additionally, we present a forward-forgeable signature construction, GroupForge. To that end, we combine a DSKE scheme with a

*Both contributed equally and are considered to be co-first authors.

Merkle tree and timestamps, thereby obtaining a "short-lived" signature with extractable sets, which provide deniability under a fixed public key. Finally, we demonstrate that GroupForge can replace Keyforge in the non-attributable email protocol of Specter, Park, and Green (USENIX Sec '21), hence eliminating the need to continuously disclose outdated private keys.

1 Introduction

Digital signature schemes [27] play an important role in protecting the integrity of data transmitted over the Internet. In some jurisdictions [9, 40, 35], a digital signature applied to data can serve as evidence of the sender's authorship of the data. Moreover, the signature of a message remains valid until either the underlying signature scheme is broken or the private key is compromised. However, as pointed out by Borisov, Goldberg, and Brewer in their work on off-the-record (OTR) communication work [15], this "long-lived" property is unsuitable for certain types of messages. For instance, if Alice wishes to communicate privately with Bob, she can encrypt her messages using Bob's public key (i.e., for confidentiality) and sign them with her private key (i.e., for authenticity). However, if Eve compromises Bob's computer at some point in the future, Eve will be able to read all of Bob's previous messages from Alice, and use the signatures to prove to Judy that the messages indeed originated from Alice.

To address this problem, OTR messaging requires an interactive key agreement protocol between the sender and the recipient to agree on session keys before exchanging messages. However, this pair-wise key agreement required in OTR is not scalable for applications such as email protocols, in which there is often no prior end-to-end interaction among the involved parties.

Another way to achieve deniability is to simply require the sender to periodically rotate keys and publish their old private keys [28]. This method enables any party to forge signatures using the published private keys and thus offers deniability to old transcripts. In fact, this method is being suggested to offer deniability in *domain keys identified mail* (DKIM) [2], where SMTP servers sign outgoing emails on behalf of the whole domain using a single key, as a way to safeguard against email spoofing.

In this paper, we focus on answering the following research question that arises naturally from the limitations of existing attempts:

Is it possible to design a signature scheme that allows the recipients to verify the validity of the signature, while enabling the sender to gain plausible deniability, without requiring the constant publication of old private keys and the rotation of new public keys?

Table 1: Comparison of signature schemes with deniability.

Scheme	Requiring Future Actions (e.g., Publishing Key Material)	Requiring External Services (e.g., Random Beacon)	Requiring VDF
KeyForge [50]	Yes	No	No
TimeForge (version in [49])	No	Yes	Yes
Short-lived signature [3]	No	Yes	Yes
GroupForge$_{\mathsf{hash}}$ (based on DSKE$_{\mathsf{hash}}$)	No	No	No
GroupForge$_{\mathsf{poly}}$ (based on DSKE$_{\mathsf{poly}}$)	No	No	No

It is worth noting that the question itself is, seemingly, a contradiction due to the non-repudiation property of digital signature schemes. This work circumvents the contradiction by showing that any one-time hash-based signature scheme (e.g., Lamport OTS [38]) and our new signature scheme based on polynomial commitments [36] can effectively convince the recipients of the authenticity of the message for a fixed period of time, while offering the signer plausible deniability. This is achieved by introducing the notions of an *extractable set*, a set of signatures from which the private key can be extracted, and a *deniable group*, a group of recipients that can, using an extractable set of signatures, collectively reconstruct the private key.

Contributions. Our contributions are summarized as follows:

- We formally define the notion of digital signature scheme with key extraction (DSKE). In DSKE, the signer always has an *extractable set*, a set of signatures that can be used to extract the private key.
- We present a concrete construction of DSKE utilizing the Lamport Signature Scheme. We demonstrate that reusing a Lamport private key can result in an extractable set with an overwhelming probability. Moreover, our technique is applicable to other hash-based signature schemes, such as the Winternitz One-time signature scheme [41] and the Optimized Lamport signature scheme [41].
- We propose another digital signature construction based on a polynomial commitment primitive, DSKE$_{\mathsf{poly}}$, which might be of independent interest. DSKE$_{\mathsf{poly}}$ eliminates the limitations of hash-based signatures by providing short signatures and allowing signers to choose the size of the extractable set.
- Finally, we propose GroupForge, a forward-forgeable signature scheme that combines DSKE with a Merkle hash tree and timestamps to improve the non-attributable email system proposed by Specter, Park, and Green [50]. Specifically, GroupForge eliminates the need for email servers to periodically post expired key material, as well as the reliance on a trusted verifiable time-keeping service.

1.1 Related Work

DSKE. Designing digital signature schemes with a key-extraction property has already been explored in the context of double authentication preventing signatures (DAPS). This primitive enables the extraction of the private key if a signer creates multiple signatures on the same content. Since DAPS are genuinely designed with the purpose of double or multiple tauthentication prevention [44, 8, 22], they aim at messages of special form, namely $m = (a, p)$, where a is an address and p is a payload. In case a signer signs two (or more) messages with the same address but different payloads, its private key is leaked. However, DSKE essentially provides key extractability as an inherent feature without making assumption on the type of message, increasing its applicability. In particular, the key extraction in our polynomial commitment-based DSKE directly comes from the polynomial interpolation theorem. Note that one major downside of many DAPS schemes is their limited address space, i.e., exponentially large address space is not supported. Moreover, a notable difference between DSKE and DAPS is that the former has a sole design similar to the normal signatures in the literature like the BLS signature [13], while the latter has a hybrid design, containing different components. For instance, DAPS of [8] builds on trapdoor identification schemes [42] and that of [22] involves encryption scheme and secret sharing. This, in turn, results in DAPS having considerably larger key pairs and also signature size compared to the normal signatures [22].

GroupForge. Specter, Park, and Green formally defined the notion of Forward-Forgeable Signature (FFS) [50] and showed how FFS can be used to achieve deniability in the email protocol. The main idea of their scheme is to make the signatures become forgeable after a fixed delay. They present two concrete constructions: KeyForge and TimeForge. In the KeyForge construction, the email server needs to periodically publish expired keys to claim deniability over sent emails, and in the TimeForge construction, the signers need to rely on a trusted publicly verifiable time-keeping (PVTK) service (PVTK) to provide them with a verifiable clock time proof. The forgeability comes from the possibility of obtaining a valid proof by querying the PVTK service after a fixed delay.

Arun, Bonneau, and Clark proposed a very similar notion to FFS called Short-lived Signatures [3]. The main idea of their work is to leverage a disjunctive statement to achieve deniability, building up on the previous efforts in the literature, e.g., designated verifier proofs [33, 51], proofs-of-work-or-knowledge (PoWorKs) [5], ring signatures [47], and one-out-of-many proofs [30]. In particular, the construction in Arun *et al.*'s work [3] deploys verifiable delay functions [10, 53] as its main building block together with a trusted random beacon [45, 52]. They use a statement of the form: *I know the witness, x (e.g., private key) OR someone solved a VDF on a beacon value derived from the trusted beacon before a time, t.* Hence, their

construction offers deniability to the prover because anyone can produce a valid proof by evaluating the VDF through sequential computation. This work, on the other hand, offers a simpler approach without requiring costly VDF evaluations and a trusted random beacon. We provide a comparison of our work and state-of-the-art signature schemes with deniability in Table 1.

Finally, there is a subtle difference between the aforementioned types of signatures. While forward-forgeable signatures provide non-attributability by selective release of some *expiry* information, short-lived signatures *automatically* become non-attributable after some time without further action. Interestingly, GroupForge, proposed in this work, can get the best of both worlds. In other words, as the deniability property of GroupForge relies on the *number* of generated signatures, it essentially can be considered a short-lived signature in scenarios where the signer is supposed to generate a *certain* number of signatures, achieving deniability for free without requiring any trusted service or computing costly VDFs. Moreover, GroupForge can also be modelled as a forward-forgeable signature by having the signer generate a certain number of signatures on a set of distinct messages, achieving forgeability without constantly publishing key materials that is problematic in practice due to unreliable distribution [50].

Paper Organization. The remainder of the paper is organized as follows. Section 2 outlines the necessary background and building blocks for DSKE. In Section 3, we present the formal definition of DSKE. Sections 4 and 5 provide concrete constructions of DSKE, namely $\mathsf{DSKE}_{\mathsf{hash}}$, based on hash-based signature schemes, and $\mathsf{DSKE}_{\mathsf{poly}}$, based on polynomial commitment schemes, respectively. We show the construction of GroupForge from DSKE schemes and demonstrate its application to non-attributable email protocols in Section 6. We evaluate the performance of DSKE and GroupForge constructions, and discuss DSKE's limitations and potential improvements in Section 7. Section 8 concludes our work.

2 Preliminaries

In this section, we give a background on the building blocks used for our constructions, namely, properties of hash functions and polynomial commitments.

Notation. We denote by 1^λ the security parameter and by $\mathsf{negl}(\lambda)$ a negligible function of λ. We express by (pk, sk) a pair of public and private keys. We let $[n]$ denote the set $\{1, \ldots, n\}$. Moreover, we require that pk can always be efficiently derived from sk, and we denote $\mathsf{extractPK}(sk) = pk$ to be the deterministic function for doing so. We denote as $m||n$ the concatenation of two values m and n, and as $\mathbb{Z}_{\geq a}$ the set of integers that are greater than or equal to a. For a field $\mathbb{F}$, we denote $\mathbb{F}^{\leq d}(X)$ the set of polynomials in $\mathbb{F}[X]$ with degree at most d. We denote by $\mathcal{M}$ the message space and $\mathcal{S}$ the

signature space. For a non-negative integer j, we let $f^{(j)}$ be the j-th iterate of the function f, i.e., $f^{(j)}(x) = f(f(\cdots(x)\cdots))$ where f is repetitively calculated j times.

2.1 Hash Functions

Our constructions employ the following standard properties of cryptographic hash functions. We use $H : \mathcal{K} \times \mathcal{M} \to \{0,1\}^\lambda$ to denote a family of hash functions that is parameterized by a key $k \in \mathcal{K}$ and message $m \in \mathcal{M}$ and outputs a binary string of length λ. For this work, we consider two security properties of hash functions from [48], namely, preimage resistance and collision resistance.

Definition 1 (Preimage Resistance)**.** *A family H of hash functions is preimage-resistant, if for any PPT adversary $\mathcal{A}$, the adversary's advantage in finding the preimage of a given hash value is:*

$$\Pr\left[\begin{matrix} k \xleftarrow{\$} \mathcal{K}, x \xleftarrow{\$} \mathcal{M}, y \leftarrow H(k,x) \\ x' \leftarrow \mathcal{A}(k,y) \end{matrix} : H(k,x') = y\right] \leq \mathsf{negl}(\lambda)$$

Definition 2 (Collision Resistance)**.** *A family H of hash functions is collision-resistant, if for any PPT adversary $\mathcal{A}$, the adversary's advantage in finding collisions is:*

$$\Pr\left[\begin{matrix} k \xleftarrow{\$} \mathcal{K} \\ (x,x') \leftarrow \mathcal{A}(k) \end{matrix} : (x \neq x') \wedge (H(k,x) = H(k,x'))\right] \leq \mathsf{negl}(\lambda)$$

In practice, the key for standard hash functions is public; therefore, from this point, we refer to the cryptographic hash function h sampled from a family of hash functions as a fixed function $h : \mathcal{M} \to \{0,1\}^\lambda$.

2.2 Polynomial Commitment Scheme

A polynomial commitment scheme (PCS) allows a prover to commit to a polynomial $f(X) \in \mathbb{F}^{\leq d}(X)$ and later open $f(X)$ at arbitrary points x, revealing only the value $f(x)$. We assume a succinct PCS, where the commitment C_f to $f(X)$ and the opening proofs π consist of single group elements, for some group G. A PCS consists of the following algorithms.

- $\mathsf{Setup}(1^\lambda, d) \to (ck, vk)$: The generation algorithm takes as input a security parameter λ and a maximum degree number $d \in \mathbb{N}$ and outputs the public commitment key ck, which allows committing to polynomials in $\mathbb{F}^{\leq d}(X)$, and the public verification key vk.
- $\mathsf{Com}(ck, f(X)) \to C_f$: The commitment algorithm takes as input the commitment key ck and a polynomial $f(X) \in \mathbb{F}^{\leq d}(X)$ and outputs a commitment $C_f \in G$ to the polynomial $f(X)$.

- $\mathsf{Open}(ck, C_f, x, f(X)) \to (\pi, y)$: The opening algorithm takes as input a commitment key ck, a commitment C_f, an evaluation point x, and the polynomial $f(X)$, and outputs $y = f(x) \in \mathbb{F}$ and a proof $\pi \in G$.
- $\mathsf{Check}(vk, C_f, x, y, \pi) \to b \in \{0, 1\}$: The checking algorithm takes as input the verification key vk, the commitment C_f, a point x, the claimed evaluation y, and the opening proof π, and outputs 1 iff $y = f(x)$.

Our schemes demand the following *correctness*, *hiding*, *evaluation binding*, and *polynomial binding* properties from the polynomial commitment scheme.

Definition 3 (Correctness [36])**.** *Let* $(ck, vk) \leftarrow \mathsf{Setup}(1^\lambda, k)$, $f(X) \in \mathbb{F}^{\leq d}(X)$, *and* $C_f \leftarrow \mathsf{Com}(ck, f(X))$. *Then for any* (π, y) *output by* $\mathsf{Open}(ck, C_f, x, f(X))$, *we have that* $\mathsf{Check}(vk, C_f, x, y, \pi) \to 1$.

Definition 4 (Computational Hiding [36])**.** *Given* (ck, vk), *the commitment* C_f, *and up to* d *valid openings* (y_i, π_i) *for points* x_i, *where* $i \in \{1, \ldots, d\}$, *no PPT adversary can determine the value* $f(x')$, *for* $x' \notin \{x_1, \ldots, x_d\}$, *except with negligible probability.*

Definition 5 (Evaluation Binding [36])**.** *Given* (ck, vk), *no PPT adversary can compute commitment* C_f, *point* x, *and two openings* (π_1, y_1), (π_2, y_2) *for* x, *such that* $\mathsf{Check}(vk, C, x, y_1, \pi_1) = 1$, $\mathsf{Check}(vk, C, x, y_2, \pi_2) = 1$, *and* $y_1 \neq y_2$.

Definition 6 (Polynomial Binding [36])**.** *Given* (ck, vk), *no PPT adversary can compute polynomials* $f(X)$ *and* $f'(X)$, *such that* $f(X) \neq f'(X)$ *and* $\mathsf{Com}(ck, f(X)) = \mathsf{Com}(ck, f'(X))$.

2.3 KZG Polynomial Commitment Scheme

We now present a concrete polynomial commitment construction, the KZG [36] scheme. It works over a bilinear pairing group $\mathbb{G} = \langle e, G, G_t \rangle$, where G is a group of prime order p, e is a symmetric pairing $e : G \times G \to G_t$, g is a generator of G, and $h \in G$.

- $\mathsf{Setup}\ (1^\lambda, d) \to (\mathbb{G}, \mathsf{ck}, \mathsf{vk})$: the algorithm outputs a representation of the bilinear group $\mathbb{G}$, commitment key $ck = \{g, g^\alpha, \ldots, g^{\alpha^d}\}$, and verification key $vk = h^\alpha$, for an $\alpha \in \mathbb{Z}_p$ that is destroyed after setup.
- $\mathsf{Com}(ck, f(X)) \to C_f$: the algorithm computes $C_f = g^{f(\alpha)}$ using ck and outputs C_f.
- $\mathsf{Open}(ck, C_f, x, f(X)) \to (\pi, y)$: the algorithm computes $y = f(x)$ and the quotient polynomial $q(X) = \frac{f(X)-y}{X-x}$, and outputs y and $\pi = C_q = \mathsf{Com}(ck, q(X))$.

- $\mathsf{Check}(vk, C_f, x, y, \pi) \to b \in \{0,1\}$: the algorithm outputs 1 if $e(C_f \cdot g^{-y}, h) = e(C_q, h^{\alpha} \cdot h^{-x})$, and 0 otherwise.

The KZG scheme satisfies the *correctness*, *computational hiding*, *evaluation binding*, and *polynomial binding* properties, provided the DL and d-SDH assumptions hold in $\mathbb{G}$ [36].

3 Digital Signature with Key Extraction (DSKE)

In this section, we formally define the notion of Digital Signatures with Key Extraction (DSKE). We adopt the standard digital signature definition and introduce a new algorithm to capture the capability of extracting the private key from a set of signatures.

Definition 7 ((k,δ)-Digital signature with Key Extraction)**.** *A signature scheme, Σ, with key extraction consists of five algorithms:*

- $\mathsf{Setup}(1^{\lambda}) \to par$*: The setup algorithm takes a security parameter 1^{λ} and outputs a set of public parameters, par. This algorithm runs once, and the public parameters are implicitly input to all subsequent algorithms.*
- $\mathsf{KeyGen}() \to (pk, sk)$*: The* probabilistic *generation algorithm outputs a pair (pk, sk) of public key and private key.*
- $\mathsf{Sign}(sk, m) \to \sigma$*: The signing algorithm is a* probabilistic *algorithm that takes a private key sk and a message m from the message space $\mathcal{M}$ as input and outputs a signature σ in the signature space $\mathcal{S}$.*
- $\mathsf{Verify}(pk, m, \sigma) \to b \in \{0,1\}$*: The verifying algorithm is a* deterministic *algorithm that takes a public key pk, a message m, and a signature σ, and outputs the validity of the signature, $b \in \{0,1\}$.*
- $\mathsf{Extract}(\{(m_i, \sigma_i)\}_{i\in[k]}, pk) \to sk$*: The extraction algorithm is a* probabilistic *algorithm that takes as input a set of distinct message-signature pairs $\{m_i, \sigma_i\}_{i\in[k]}$, such that $\sigma_i \leftarrow \mathsf{Sign}(sk, m_i)$, and the public key pk, and outputs the underlying private key sk with probability δ and $\perp$ with probability $1-\delta$.*

Apart from the straightforward correctness definition, we consider two other properties of DSKE: unforgeability and the existence of an extractable set. The security of digital signature is defined through the following experiment.

The d-times signature experiment $\mathsf{SignExp}^{d}_{\mathcal{A},\Sigma}(\lambda)$.

1. $\mathsf{Setup}(1^{\lambda})$ and $\mathsf{KeyGen}()$ are run to obtain keys (pk, sk).

2. $\mathcal{A}$ is given pk and can ask up to d queries to the signing oracle $\mathsf{Sign}(sk, \cdot)$. Let $Q_{\mathcal{A}}^{\mathsf{Sign}(sk,\cdot)} = \{m_i\}_{i\in[d]}$ be the set of all messages for which $\mathcal{A}$ queries $\mathsf{Sign}(sk, \cdot)$, where the i^{th} query is a message $m_i \in \mathcal{M}$. Eventually, $\mathcal{A}$ outputs a pair $(m^*, \sigma^*) \in \mathcal{M} \times \mathcal{S}$.

3. The output of the experiment is defined to be 1 if and only if $m^* \notin Q_{\mathcal{A}}^{\mathsf{Sign}(sk,\cdot)}$ and $\mathsf{Verify}(pk, m^*, \sigma^*) = 1$.

Definition 8 (Existential Unforgeability)**.** *A digital signature scheme* Σ *is* existentially unforgeable under a d-times adaptive chosen-message attack*, or* d-times-secure*, if for all PPT adversaries* $\mathcal{A}$ *the success probability in the previous experiment is negligible, that is,*

$$\Pr[\mathsf{SignExp}^{d}_{\mathcal{A},\Sigma}(\lambda) = 1] \leq \mathsf{negl}(\lambda).$$

Definition 9 (Extractable Set)**.** *A digital signature scheme has a* (k, δ)*-extractable set when the extraction algorithm* $\mathsf{Extract}(\cdot)$ *on input* k *distinct message-signature pairs* $\{(m_i, \sigma_i)\}_{i\in[k]}$ *and the public key* pk*, such that each* σ_i *is a valid signature on* m_i *under* pk*, outputs the private key* sk *with probability* δ*. That is,*

$$\Pr\left[\begin{array}{ll} m_i \in \mathcal{M},\ s.t.\ m_i \neq m_j,\ for\ i, j \in [k], i \neq j & \\ (pk, sk) \leftarrow \mathsf{KeyGen}() & \\ \sigma_i \leftarrow \mathsf{Sign}(sk, m_i) & : sk = sk' \\ \mathsf{Extract}\left(\{(m_i, \sigma_i)\}_{i\in[k]}, pk\right) \rightarrow sk' & \end{array}\right] = \delta$$

4 DSKE from Hash-Based Signature Schemes

In this section, we provide $\mathrm{DSKE}_{\mathsf{hash}}$, a DSKE construction based on the Lamport signature scheme.

4.1 Lamport OTS Construction

In our construction of $\mathrm{DSKE}_{\mathsf{hash}}$, the key generation, signing, and verification algorithms remain unchanged as Lamport Signature Scheme. We provide an additional algorithm to extract the private key from a set of signatures.

Lamport Signature Scheme. The Lamport Signature Scheme is parameterized by a preimage-resistant hash function f and a collision-resistant hash function, g. The private key SK contains 2λ binary strings uniformly sampled from $\{0,1\}^{\lambda}$, where λ is the security parameter. The public key PK consists of 2λ binary strings that are evaluations of f on each element in the private key. To form a signature σ, on a message m, the signer reveals components of the private key SK, according to the binary representation of

$g(m)$ as the signature. Verification is carried out naturally: the verifier uses the binary representation of the digest, $g(m)$, to validate if each element contained in the signature is the actual preimage of the public key elements.

Extracting Private Key from Message-Signature Pairs. As shown in Figure 1, the intuition behind the extracting function is that each signature σ_i is a subset of the private key SK. To extract the complete private key, one needs to compute the union of k different signatures $\{\sigma_i\}_{i\in[k]}$. Since the underlying hash function g is unpredictable, the probability of successfully extracting the private key depends on the number of distinct message-signature pairs (i.e., k).

Lamport Hash-Based DSKE. Based on the above intuitions, we propose a hash-based construction, $\text{DSKE}_{\mathsf{hash}}$, which consists of:

- $\mathsf{Setup}(1^\lambda)$: On input the security parameter λ, the algorithm outputs a parameter, par, containing a hash function, $f : \{0,1\}^\lambda \to \{0,1\}^\lambda$, chosen from a family of preimage-resistant hash functions, and a hash function, $g : \mathcal{M} \to \{0,1\}^\lambda$, chosen from a family of collision-resistant hash function. The public parameters are implicitly input to all subsequent algorithms.

- $\mathsf{KeyGen}()$: For each $i \in [\lambda], b \in \{0,1\}$, sample $sk_i[b] \xleftarrow{\$} \{0,1\}^\lambda$; compute $pk_i[b] = f(sk_i[b])$; and output the private key, $SK = (sk_i[b])_{i\in[\lambda],b\in\{0,1\}}$, and the public key, $PK = (pk_i[b]))_{i\in[\lambda],b\in\{0,1\}}$.

- $\mathsf{Sign}(SK, m)$: Parse $SK = (sk_i[b])_{i\in[\lambda],b\in\{0,1\}}$; compute $d = g(m) = (d_i)_{i\in[\lambda]}$; and output $\sigma = (sk_i[d_i])_{i\in[\lambda]}$.

- $\mathsf{Verify}(PK, m, \sigma)$: Parse $PK = (pk_i[b])_{i\in[\lambda],b\in\{0,1\}}$, $\sigma = (\sigma_i)_{i\in[\lambda]}$, and compute $d = g(m) = (d_i)_{i\in[\lambda]}$. For all $i \in [\lambda]$, if $f(\sigma_i) \neq pk_i[d_i]$, return 0. Otherwise, return 1.

- $\mathsf{Extract}(\{m_j, \sigma_j\}_{j\in[k]}, PK)$: For each message-signature pair (m_j, σ_j), compute $d_j = g(m_j) = (d_{ji})_{i\in[\lambda]}$, and parse $\sigma_j = (\sigma_{ji})_{i\in[\lambda]}$. For each $j \in [k], i \in [\lambda], b \in \{0,1\}$, if $\exists d_{ji} = b$, then let $sk_i[b] = \sigma_{ji}$. Set $SK = (sk_i[b])_{i\in[\lambda],b\in\{0,1\}}$ and parse $PK = (pk_i[b])_{i\in[\lambda],b\in\{0,1\}}$. If for all $i \in [\lambda]$ and $b \in \{0,1\}$, $pk_i[b] = f(sk_i[b])$, then output SK. Otherwise, the algorithm outputs $\perp$.

Security Analysis. In this part, we prove that $\text{DSKE}_{\mathsf{hash}}$ satisfies unforgeability and has an extractable set depending on the number of distinct message-signature pairs.

Theorem 1 (Existential Unforgeability)**.** *$DSKE_{\mathsf{hash}}$ is* existentially unforgeable under a 1-time adaptive chosen-message attack*, that is,*

$$\Pr[\mathsf{SignExp}^1_{\mathcal{A},DSKE_{\mathsf{hash}}}(\lambda) = 1] \leq \mathsf{negl}(\lambda)$$

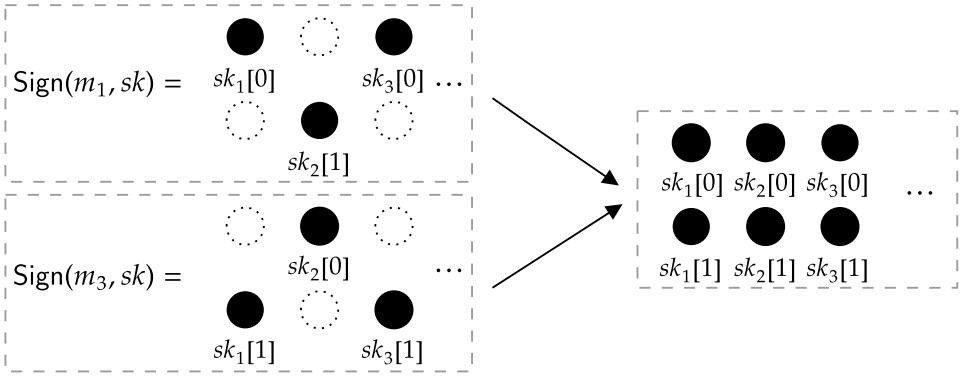

Figure 1: Example of the algorithm Extract for $\mathrm{DSKE}_{\mathsf{hash}}$. In this example, the original private key can be collectively reconstructed from $\{(m_i, \sigma_i)\}_{i\in[2]}$.

Proof. We refer reader to Theorem 14.2 in [14] to prove Theorem 1. □

Theorem 2 (Extractable Set)**.** *If $g(\cdot)$ is modeled as a random oracle, then $DSKE_{\mathsf{hash}}$ has a (k, δ)-extractable set, where:*

$$\delta \geq 1 - \frac{\lambda}{2^{k-1}} - \mathsf{negl}(\lambda)$$

Proof. We define $d_i = g(m_i)$. All d_i for $i \in \{1, \ldots, k\}$ forms a $k \times \lambda$ matrix:

$$\begin{pmatrix} g(m_1) \\ \cdots \\ g(m_k) \end{pmatrix} = \begin{pmatrix} d_1 \\ \cdots \\ d_k \end{pmatrix} = \begin{pmatrix} d_{1,1} & \ldots & d_{1,\lambda} \\ \ldots & \ddots & \ldots \\ d_{k,1} & \ldots & d_{k,\lambda} \end{pmatrix}$$

where $d_{i,j} \in \{0, 1\}$.

As shown in Figure 1, for $j \in [\lambda], b \in \{0, 1\}$, to extract the value of $sk_j[b]$, there should be at least one d_{ij} that satisfies $d_{ij} = b$, i.e., for $j \in [\lambda]$, d_{ij} should not all be $1 - b$. Therefore, to be able to extract the private key SK, we do not want any columns in the matrix that contain all 0s or all 1s.

Since $g(\cdot)$ is modeled as a random oracle, the probability that one single column consists of all 0s or all 1s is:

$$\Pr\left[x \xleftarrow{\$} \{0,1\}^k : x = 0^k \vee x = 1^k\right] = \frac{1}{2^{k-1}}$$

Let's denote Bad_0 to be the event that at least one of the columns is all 0s or all 1s. We have:

$$\begin{aligned} \Pr[\mathsf{Bad}_0] &= \Pr\left[\vee_{j=1}^{\lambda}\left(x_j \xleftarrow{\$} \{0,1\}^k : x_j = 0^k \wedge x_j = 1^k\right)\right] \\ &\leq \sum_{j=1}^{\lambda} \Pr\left[x_j \xleftarrow{\$} \{0,1\}^k : x_j = 0^k \vee x_j = 1^k\right] \\ &= \frac{\lambda}{2^{k-1}} \end{aligned}$$

Also, let's denote Bad_1 to be the event that there are two private key elements that map to the same public key element. However, since $f(\cdot)$ is modeled as a random oracle, this probability is negligible; hence, $\Pr[\mathsf{Bad}_1] = \mathsf{negl}(\lambda)$.

If g and f are modeled as a random oracle, the probability that the extraction algorithm $\mathsf{Extract}(\cdot)$ on input k distinct message-signature pairs $\{(m_i, \sigma_i)\}_{i\in[k]}$ and the public key PK, such that σ_i is a valid signature on m_i under PK, outputs the original private key SK that corresponds to PK is $\delta \geq 1 - \Pr[\mathsf{Bad}_0 \vee \mathsf{Bad}_1] \geq 1 - \frac{\lambda}{2^{k-1}} - \mathsf{negl}(\lambda)$. □

4.2 Winternitz OTS construction

In the following, we demonstrate how hash-based Winternitz can also be a good candidate for DSKE construction.

Winternitz Hash-based DSKE. The Winternitz signature scheme is another hash-based signature scheme, that allows a trade-off between computation and the size of signature. Winternitz Hash-based DSKE, $\mathrm{DSKE}_{\mathsf{wots}}$, which consists of the following algorithms:

- $\mathsf{Setup}(1^\lambda)$: On input the security parameter λ, the algorithm outputs the public parameter, *par*, which contains: *(1)* a hash function, $f : \{0,1\}^\lambda \to \{0,1\}^\lambda$, chosen from a family of preimage-resistant hash functions, *(2)* a hash function, $g : \mathcal{M} \to \{0,1\}^\nu$, chosen from a family of collision-resistant hash function *(3)* a parameter integer w, and *(4)* a domination free function P parameterized by g and w, which maps a message m to a vector $\vec{s}$ of length λ, and each component of s is a number in $\{0, \cdots, w\}$, namely $P : \mathcal{M} \to I_w^\lambda$, where $I_w^\lambda = (\{0, \ldots, w\})^\lambda$. The public parameters are implicitly input to all subsequent algorithms.

- $\mathsf{KeyGen}()$: For each $i \in [\lambda]$, sample $sk_i \xleftarrow{\$} \{0,1\}^\lambda$; compute $pk_i = f^{(w)}(sk_i)$; and output the private key, $SK = (sk_i)_{i\in[\lambda]}$, and the public key, $PK = (pk_i)_{i\in[\lambda]}$.

- $\mathsf{Sign}(SK, m)$: Parse $SK = (sk_i)_{i\in[\lambda]}$; compute $\vec{s} = P(m) = (s_1, \cdots, s_\lambda) \in I_w^\lambda$; and output $\sigma = (f^{(s_1)}(sk_1), \cdots, f^{(s_\lambda)}(sk_\lambda))$.

- $\mathsf{Verify}(PK, m, \sigma)$: Parse $PK = (pk_i)_{i\in[\lambda]}$, $\sigma = (\sigma_i)_{i\in[\lambda]}$, and compute $\vec{s} = P(m) = (s_1, \cdots, s_\lambda) \in I_w^\lambda$. For all $i \in [\lambda]$, if $f^{(w-s_i)}(\sigma_i) \neq pk_i$, return 0. Otherwise, return 1.

- $\mathsf{Extract}(\{m_j, \sigma_j\}_{j\in[k]}, PK)$: For each message-signature pair (m_j, σ_j), compute $\vec{s_j} = P(m_j) = (s_{ji})_{i\in[\lambda]}$, and parse $\sigma_j = (\sigma_{ji})_{i\in[\lambda]}$. For each $j \in [k], i \in [\lambda]$, if $\exists s_{ji} = 0$, then let $sk_i = \sigma_{ji}$. Set $SK = (sk_i)_{i\in[\lambda]}$ and parse $PK = (pk_i)_{i\in[\lambda]}$. If for all $i \in [\lambda]$, $pk_i = f^{(w)}(sk_i)$, then output SK. Otherwise, the algorithm outputs $\perp$.

Domination Free Function. The domination free function P determines the existential unforgeability and extractable set of our Winternitz Hash-based $\mathrm{DSKE}_{\mathsf{wots}}$. We use the definition of the domination free function construction from [14].

Given a message $m \in \mathcal{M}$, we compute $g(m)$ and convert it to an integer in $[0, 2^{\nu})$. Given the public parameter w, let λ_0 be the smallest number satisfying $2^{\nu} \leq (w+1)^{\lambda_0}$, set $\lambda_1 = \log_{w+1}(w \cdot \lambda_0) + 1$ and $\lambda = \lambda_0 + \lambda_1$. Given an input message m, the function P works as follows:

- Compute $g(m)$, convert $g(m)$ to a λ_0-digit number in base $(w+1)$: $(s_1, \cdots, s_{\lambda_0})$.
- Compute the checksum $c = w \cdot \lambda_0 - (s_1 + \cdots + s_{\lambda_0})$.
- Convert c to a λ_1-digit number in base $(w+1)$: $(c_1, \cdots, c_{\lambda_1})$.
- Output $(s_1, \cdots, s_{\lambda_0}, c_1, \cdots, c_{\lambda_1})$.

In this case, the function P is domination free [14] and can map the message m to a vector $(s_1, \cdots, s_{\lambda_0}, c_1, \cdots, c_{\lambda_1}) \in I_w^{\lambda}$.

The intuition of the domination free function is to ensure that if a forger can obtain a valid signature for m' from one signature for m, domination property of P ensures that there exists one element s_i' in $P(m')$ that is smaller than $s_i \in P(m)$. Therefore, it implies the fact that such a forger can be used to invert the preimage-resistant hash function f.

Security Analysis. In this part, we prove that $\mathrm{DSKE}_{\mathsf{wots}}$ satisfies unforgeability and has an extractable set depending on the number of distinct message-signature pairs.

Theorem 3 (Existential Unforgeability)**.** *$DSKE_{\mathsf{wots}}$ is* existentially unforgeable under a 1-time adaptive chosen-message attack*, that is,*

$$\Pr[\mathsf{SignExp}^{1}_{\mathcal{A}, DSKE_{\mathsf{wots}}}(\lambda) = 1] \leq \mathsf{negl}(\lambda)$$

Proof. We defer the proof of Theorem 3 to Theorem 14.4 in [14]. □

In the following our analysis for the extractable set, we assume that the checksum computed by P is independent and uniformly random. This is not really the case, because there is a dependency between the check sum elements and other elements in the vector s computed by P. However, as mentioned in [29], without such assumption, the analysis will become significantly more complex.

Theorem 4 (Extractable Set)**.** *If g is modelled as a random oracle and the domination free function $P(\cdot)$ outputs uniformly random vector in I_w^{λ}, then $DSKE_{\mathsf{wots}}$ has an (k, δ)-extractable set, where:*

$$\delta \geq 1 - \frac{\lambda \cdot w^k}{(w+1)^k} - \mathsf{negl}(\lambda)$$

Proof. Following the existing work [29], we assume that in a domination free function P, the checksum c is independent of the digests $(s_1, \cdots, s_{\lambda_0})$. Therefore, given a message m to the function P, we can denote its output as $(s_1, \cdots, s_{\lambda_0}, s_{\lambda_0+1}, \cdots, s_\lambda) \in I_w^\lambda$.

We define $s_i = P(m_i)$. All s_i for $i \in \{1, \ldots, k\}$ forms a $k \times \lambda$ matrix:

$$\begin{pmatrix} P(m_1) \\ \cdots \\ P(m_k) \end{pmatrix} = \begin{pmatrix} s_1 \\ \cdots \\ s_k \end{pmatrix} = \begin{pmatrix} s_{1,1} & \cdots & s_{1,\lambda} \\ \cdots & \ddots & \cdots \\ s_{k,1} & \cdots & s_{k,\lambda} \end{pmatrix}$$

where $s_{i,j}$ is a number in $\{0, \cdots, w\}$.

For $j \in [\lambda]$, to extract the value of sk_j, there should be at least one s_{ij} that satisfies $s_{ij} = 0$, i.e., for $j \in [\lambda]$, s_{ij} should not all be non-zero. Therefore, to be able to extract the private key SK, we do not want any columns in the matrix that contain all non-zero values.

We further assume that the output of the function P is uniformly random, then the probability that one single column consists of all non-zero values is:

$$\Pr\left[x \xleftarrow{\$} \{0, \cdots, w\}^k : x \in \{1, \cdots, w\}^k\right] = \frac{w^k}{(w+1)^k}$$

Let's denote $\mathsf{Bad_0}$ to be the event that at least one of the columns is all non-zero values. We have:

$$\begin{aligned} \Pr[\mathsf{Bad_0}] &= \Pr\left[\vee_{j=1}^{\lambda}\left(x_j \xleftarrow{\$} \{0, \cdots, w\}^k : x_j \in \{1, \cdots, w\}^k\right)\right] \\ &\leq \sum_{j=1}^{\lambda} \Pr\left[x_j \xleftarrow{\$} \{0, \cdots, w\}^k : x_j \in \{1, \cdots, w\}^k\right] \\ &= \frac{\lambda \cdot w^k}{(w+1)^k} \end{aligned}$$

Also, we denote Bad_1 to be the event that there are two private key elements that map to the same public key element. However, since $f(\cdot)$ is modeled as a random oracle, this probability is negligible; hence, $\Pr[\mathsf{Bad}_1] = \mathsf{negl}(\lambda)$.

If P and f are modeled as a random oracle, the probability that the extraction algorithm $\mathsf{Extract}(\cdot)$ on input k distinct message-signature pairs $\{(m_i, \sigma_i)\}_{i\in[k]}$ and the public key PK, such that σ_i is a valid signature on m_i under PK, outputs the original private key SK that corresponds to PK is $\delta \geq 1 - \Pr[\mathsf{Bad_0} \vee \mathsf{Bad}_1] \geq 1 - \frac{\lambda \cdot w^k}{(w+1)^k} - \mathsf{negl}(\lambda)$.

□

Generalization to Other One-time Signature. Our constructions of DSKE can be generalized to all hash-based OTS. The basic intuition behind

$\mathrm{DSKE}_{\mathsf{hash}}$ is that each signature σ_j discloses partial information of the private key SK. Therefore, besides Lamport signature, other hash-based OTS such as optimized Lamport's scheme, Winternitz [41], and BiBa [43, 46], which derive signatures from the private key, can also be used to construct DSKE.

5 DSKE from Polynomial Commitment Schemes

Although the hash-based DSKE scheme $\mathrm{DSKE}_{\mathsf{hash}}$ proposed in Section 4 satisfies our definition of signature with key extraction, it has two inherent weaknesses: *(i)* The key and signature sizes of $\mathrm{DSKE}_{\mathsf{hash}}$ are linear with the security parameter λ, i.e., $O(\lambda)$, making it inefficient in practice. *(ii)* The algorithm $\mathsf{Extract}$ of $\mathrm{DSKE}_{\mathsf{hash}}$ is not deterministic and the probability δ of extracting the private key depends on the size of the extractable set (i.e., k) [1]. The smaller k is, the lower the success probability is when extracting the private key. In this section, we propose a DSKE construction based on a polynomial commitment scheme that overcomes all the drawbacks of hash-based signature schemes.

5.1 Generic Transformation

In this section, we construct a DSKE scheme, $\mathrm{DSKE}_{\mathsf{poly}}$, from a polynomial commitment scheme, Π, which can overcome the drawbacks of $\mathrm{DSKE}_{\mathsf{hash}}$. In the following, we assume a degree bound $d \in \mathbb{Z}$ and a polynomial degree $\ell \in \mathbb{Z}$ that satisfy $1 \leq \ell \leq d$. The idea is to use the polynomial $f(X)$ of degree ℓ as the private key. Then the signature on a message m is the evaluation of $f(X)$ at point $x = h(m)$, where h is a collision-resistant hash function. For key extraction we employ polynomial interpolation: any set of $\ell+1$ valid message-signature pairs (m_i, σ_i) can reconstruct $f(X)$. $\mathrm{DSKE}_{\mathsf{poly}}$ works as follows.

- $\mathsf{Setup}(1^\lambda, d)$: On input the security parameter λ and degree $d \in \mathbb{N}$, the algorithm runs $\Pi.\mathsf{Setup}(1^\lambda, d)$ to obtain (ck, vk), which allows the signer to commit to polynomials in $\mathbb{F}^{\leq d}(X)$, and it samples a collision-resistant hash function $h : \mathcal{M} \to \mathbb{F}$. The public parameters, par, contain ck, vk, d, and the specification of h.

- $\mathsf{KeyGen}(\ell)$: On input $1 \leq \ell \leq d$, sample $f(X) \xleftarrow{\$} \mathbb{F}^{\ell}(X)$ as an ℓ-degree polynomial, compute $\Pi.\mathsf{Com}(ck, f(X)) \to C_f$, and set $sk = f(X)$, $pk = C_f$.

[1] This drawback can be circumvented by requiring signers to add an extra nonce to k different messages so that the formed signatures guarantee the existence of the extractable set.

- $\mathsf{Sign}(sk, m)$: Parse $sk = f(X)$, compute $x = h(m)$, and run $\Pi.\mathsf{Open}(ck, C_f, x, f(X)) \to (\pi, y)$. Output the signature, $\sigma = (\pi, y)$.

- $\mathsf{Verify}(pk, m, \sigma)$: Parse $pk = C_f$ and $\sigma = (\pi, y)$, compute $x = h(m)$, and output $\Pi.\mathsf{Check}(vk, C_f, x, y, \pi) \in \{0, 1\}$.

- $\mathsf{Extract}(\{(m_i, \sigma_i)\}_{i\in[k]}, pk)$: If $k \leq \ell$, or if m_i are not all distinct, then return $\perp$. If $\mathsf{Verify}(pk, m_i, \sigma_i) = 0$ for some $i \in [k]$, then return $\perp$. Otherwise, compute $x_i = h(m_i)$ and parse $\sigma_i = (\pi_i, y_i)$, for $i \in [k]$. The (at least $\ell + 1$) pairs (x_i, y_i) interpolate the unique polynomial $\phi(X) \in \mathbb{F}^{\ell}(X)$, where $\lambda_i(X)$ are the Lagrange coefficients: $\phi(X) = \sum_{i\in[k]} y_i\lambda_i(X)$ and $\lambda_i(X) = \prod_{m\in[k]}^{m\neq i} \frac{X-x_m}{x_i-x_m}$.

Remarks on the degree of $f(X)$. The extraction of the private key from $k \geq \ell + 1$ points requires the signer to commit to a polynomial of degree at most ℓ. As the publicly available information in ck allows the signer to commit to any polynomial in $\mathbb{F}^{\leq d}(X)$, stronger properties, such as *strong correctness* [36], *bounded-polynomial extractability* [39], and *knowledge soundness* [11], have been formulated in the literature to enforce the claimed degree on $f(X)$. However, the signer in our scheme is allowed to choose any $\ell \in [1, d]$ and has no incentive to commit to a polynomial of degree larger than ℓ, as that would cost them the deniability, as we discuss in the following sections. Hence, we can assume that $f(X)$ is indeed of degree ℓ and do not require Π to satisfy any stronger property.

Theorem 5 (Existential Unforgeability)**.** *Assuming the underlying polynomial commitment scheme* Π *satisfies the* computational hiding, evaluation binding, *and* polynomial binding *properties, and that h is a collision-resistant hash function, the DSKE scheme $DSKE_{\mathsf{poly}}$ is* existentially unforgeable *under an ℓ-times adaptive chosen-message attack. That is,*

$$\Pr[\mathsf{SignExp}^{\ell}_{\mathcal{A},\Sigma_{\mathsf{poly}}}(\lambda) = 1] \leq \mathsf{negl}(\lambda)$$

Proof. Let $\mathcal{A}$ be a PPT adversary breaking the signature scheme $\mathrm{DSKE}_{\mathsf{poly}}$. We construct a PPT algorithm $\mathcal{B}$ that runs $\mathcal{A}$ as a subroutine and attacks the hiding property of the polynomial commitment scheme Π, given that Π is evaluation and polynomial binding and that h is collision-resistant. Specifically, $\mathcal{B}$ receives from its challenger up to ℓ openings $\big(i_j, f(i_j), w_{i_j}\big)$, for $i_j \in \mathbb{F}$ and $j \in [\ell]$, and for $f(X)$ not known to $\mathcal{B}$. It outputs (i^*, y^*) for unqueried i^* and wins if $y^* = f(i^*)$. Algorithm $\mathcal{B}$ also receives C_f and d as input and is given access to ck, vk. Algorithm $\mathcal{B}$ works as follows:

- Initiate $\mathcal{A}$ with input $pk = C_f$, and create an empty set S_{quer}.

- Whenever $\mathcal{A}$ requests a signature on message m, compute $x = h(m)$ and check whether $x \in S_{\text{quer}}$. If this is the case, then $\mathcal{B}$ has already asked his challenger for the opening of point x, so $\mathcal{B}$ does not have to ask again. Otherwise, add x to S_{quer} and obtain the opening (π, y) of point x from the challenger of $\mathcal{B}$. Return $\sigma = (\pi, y)$ to $\mathcal{A}$.

- If $\mathcal{A}$ fails to output a valid forgery on an unqueried message, then abort. Otherwise $\mathcal{A}$ has output a message m^* and a forgery $\sigma^* = (\pi^*, y^*)$ on m^*. We assume wlog $\mathcal{A}$ has made ℓ signature queries (if not, $\mathcal{B}$ queries these values itself) and hence $\mathcal{B}$ has openings (π_i, y_i) for points x_i, with $i \in [\ell]$. Calculate $x^* = h(m^*)$. If $x^* \in S_{\text{quer}}$, then set $\mathsf{bad}_1 \leftarrow 1$ and abort. Otherwise interpolate $f'(X) \in \mathbb{F}^{\ell}(X)$ from the $\ell + 1$ points $\{(x_1, y_1), \ldots, (x_\ell, y_\ell), (x^*, y^*)\}$ and compute $C_{f'} = \Pi.\mathsf{Com}(ck, f'(X))$. $\mathcal{B}$ distinguishes two cases.

  1. If $C_{f'} \neq C_f$, then set $\mathsf{bad}_2 \leftarrow 1$ and abort.
  2. Otherwise, output (x^*, y^*). Observe that, even though $C_{f'} = C_f$, it could still be the case that $f(X) \neq f'(X)$.

Denote by $\Pr[\mathsf{HidExp}^{\ell}_{\mathcal{B},\Pi}(\lambda) = 1]$ the probability that $\mathcal{B}$ wins the game above, by Output the event that $\mathcal{B}$ outputs some (x^*, y^*), and by bad_3 the event that $f(X) \neq f'(X)$. Observe that $\mathcal{B}$ wins if and only if Output happens and bad_3 does not happen. Moreover, Output happens if $\mathcal{A}$ succeeds in forging a valid signature and bad_1 and bad_2 do not happen. Therefore, we have:

$$\begin{aligned}
&\Pr[\mathsf{HidExp}^{\ell}_{\mathcal{B},\Pi}(\lambda) = 1] \\
&\quad = \Pr[\mathsf{Output} \wedge \overline{\mathsf{bad}_3}] \\
&\quad \geq \Pr[\mathsf{Output}] - \Pr[\mathsf{bad}_3] \\
&\quad = \Pr[\mathsf{SignExp}^{\ell}_{\mathcal{A},\Sigma_{\mathsf{poly}}}(\lambda) = 1 \wedge \overline{\mathsf{bad}_1} \wedge \overline{\mathsf{bad}_2}] - \Pr[\mathsf{bad}_3] \\
&\quad \geq \Pr[\mathsf{SignExp}^{\ell}_{\mathcal{A},\Sigma_{\mathsf{poly}}}(\lambda) = 1] - \Pr[\mathsf{bad}_1] - \Pr[\mathsf{bad}_2] \\
&\qquad - \Pr[\mathsf{bad}_3]
\end{aligned}$$

For the bad events, we have the following.

1. The event bad_1 implies that $\mathcal{A}$ breaks the collision resistance property of $\mathcal{A}$, which is assumed secure, hence $\Pr[\mathsf{bad}_1] = \mathsf{negl}(\lambda)$.

2. Event bad_2 implies $f'(X) \neq f(X)$, and hence it must be that (x^*, y^*) is not a point of $f(X)$, i.e., $f(x^*) \neq y^*$. Since $\mathcal{A}$ succeeded, point (x^*, y^*) and proof π^* satisfy $\Pi.\mathsf{Check}(vk, C_f, x^*, y^*, \pi^*) = 1$. But in this case, $\mathcal{B}$ can break the *evaluation binding* of Π in the following way. It asks for the opening of point x^*, hence obtaining (π, y), where $y = f(x^*)$. This destroys any hopes of $\mathcal{B}$ to break the hiding property, but can attack evaluation binding, using the points (x^*, y^*) and (x^*, y), for which $y \neq$

y^*, $\Pi.\mathsf{Check}(vk, C_f, x^*, y^*, \pi^*) = 1$, and $\Pi.\mathsf{Check}(vk, C_f, x^*, y, \pi) = 1$. Since by assumption Π satisfies the evaluation-binding property, we get $\Pr[\mathsf{bad}_2] = \mathsf{negl}(\lambda)$.

3. Event bad_3 would violate the polynomial-binding property of Π, since $f(X) \neq f'(X)$ and $\mathsf{Com}(ck, f(X)) = \mathsf{Com}(ck, f'(X))$, thus $\Pr[\mathsf{bad}_3] = \mathsf{negl}(\lambda)$.

From the above, and from the assumption that Π is a hiding PCS (i.e., $\Pr[\mathsf{HidExp}^{\ell}_{\mathcal{B},\Pi}(\lambda) = 1] \leq \mathsf{negl}(\lambda)$), we get

$$\begin{aligned}\Pr[\mathsf{SignExp}^{\ell}_{\mathcal{A},\Sigma_{\mathsf{poly}}}(\lambda) = 1] \leq & \Pr[\mathsf{HidExp}^{\ell}_{\mathcal{B},\Pi}(\lambda) = 1] + \Pr[\mathsf{Bad}_1] + \\ & \Pr[\mathsf{Bad}_2] + \Pr[\mathsf{Bad}_3] \\ \leq & \mathsf{negl}(\lambda)\end{aligned}$$

□

Theorem 6 (Extractable Set)**.** *Assuming the underlying polynomial commitment scheme* Π *satisfies the* evaluation binding *property, the DSKE scheme* $DSKE_{\mathsf{poly}}$ *has a* $(k, 1 - \mathsf{negl}(\lambda))$*-extractable set for any* $k \geq \ell + 1$*. That is,*

$$\begin{aligned}\delta = \Pr & \left[\begin{array}{l} m_i \in \mathcal{M}, \ \textit{for } i \in [k] \\ m_i \neq m_j, \ \textit{for } i, j \in [k], i \neq j, \ \textit{and } k \geq \ell + 1 \\ \sigma_i \leftarrow \mathsf{Sign}(sk, m_i), \ \textit{for } i \in [k] \\ \mathsf{Extract}\left(\{(m_i, \sigma_i)\}_{i \in [k]}, pk\right) \rightarrow sk' \end{array} : sk = sk'\right] \\ = & \ 1 - \mathsf{negl}(\lambda)\end{aligned}$$

Proof. The proof follows from two facts. First, by assuming that the signer does not commit to polynomials of degree bigger than ℓ. Second, from the *evaluation binding property*, and since the points (x_i, y_i) correspond to valid signatures, we know that $y_i = f(x_i)$, for some polynomial $f(X) \in \mathbb{F}^{\leq \ell}(X)$ and for all $i \in [k]$, except with negligible probability. Due to the uniqueness of polynomial interpolation, we know that any $\ell + 1$ distinct points (x_i, y_i) define a unique polynomial $\phi(X)$ of degree at most ℓ, hence $\phi(X)$ must be the same as $f(X)$, hence $sk = sk'$ with probability $1 - \mathsf{negl}(\lambda)$. □

In this work, we use KZG scheme [37] to be our concrete construction of the polynomial commitment. Due to the space constraint, he detailed description of KZG polynomial commitment is shifted to the appendix.

6 Application to the Non-Attributable Email Protocol

6.1 Non-Attributable Email

Domain-Keys Identified Mail (DKIM) [2]. DKIM is a protocol that has been developed to prevent fraud and spam in email communication. A server sending an email cryptographically signs it, so that the recipient can verify that it originates from the reported server. A side effect of this necessary measurement is email attributability which stems from the fact that the digital signature remains valid for a long time, potentially forever [28]. As a result, a malicious actor, who at any time gains access to these emails, can provably link them to their sender, which in turn incentivizes extortion and retaliation, among others.

Forward-Forgeable Signatures. Specter *et al.* define the notion of a *forward-forgeable signature (FFS)* scheme. An FFS scheme consists of the standard $\mathsf{KeyGen}(1^\lambda)$, $\mathsf{Sign}(sk, m)$, and $\mathsf{Verify}(pk, m, \sigma)$ algorithms, and additionally an $\mathsf{Expire}(sk, \mathcal{T})$ algorithm, that generates expiry information η for a private key sk, possibly using additional information $\mathcal{T}$, and a $\mathsf{Forge}(\eta, m)$ algorithm, that, given the expiry information of a key, outputs a signature on a message m. The scheme satisfies the standard *correctness* and *unforgeability* properties, and the *forgeability on expiry* property.

Definition 10 (Forgeability on Expiry [28])**.** *A digital signature scheme satisfies the* forgeability on expiry *property if no PPT adversary can distinguish a signature created with private key sk from a signature created with the expiry information η of sk. Formally, for any $m \in \mathcal{M}$ and any PPT distinguisher $\mathcal{D}$, there is a negligible function $\mathsf{negl}(\cdot)$, such that for all λ,*

$$\Pr\left[\begin{array}{ll} (pk, sk) \leftarrow \mathsf{KeyGen}(1^\lambda) & \\ \sigma_0 \leftarrow \mathsf{Sign}(sk, m) & \\ \eta \leftarrow \mathsf{Expire}(sk, \mathcal{T}) & : b = b' \\ \sigma_1 \leftarrow \mathsf{Forge}(\eta, m) & \\ b \xleftarrow{\$} \{0,1\}, \quad b' \leftarrow \mathcal{D}(\sigma_b, \eta) & \end{array}\right] \leq 1/2 + \mathsf{negl}(\lambda)$$

Non-Attributable Emails and KeyForge. Specter, Park, and Green [50] introduce the concept of non-attributable emails that addresses the inherent drawbacks of DKIM. The core idea of non-attributable emails is to replace the signature scheme in DKIM with an FFS scheme. They provide two constructions, KeyForge and Timeforge. Both satisfy the FFS properties by requiring signers either to publish old private keys (i.e., KeyForge) or to issue signed updates to a publicly verifiable timekeeping service (i.e., TimeForge).

KeyForge, specifically, is an FFS scheme where signatures expire after a predefined delay Δ. The expiry information is the private key itself, which has to be published by the server every Δ time. It is based on a hierarchical identity-based signature scheme (HIBS). In a HIBS every signer has an identity, from which the private key can be derived. The identities are organized in a tree structure, with the property that any node (encoding an identity) can derive the private keys of its children (sub-identities). The master private key (the root of the tree) can generate all the other private keys. KeyForge uses a 4-level tree structure, and each level of the tree represents different timespans (i.e., years, months, days, and minutes). Each tag (i.e., leaf in the tree) represents a unique 15-minute time chunk, and the signer of HIBS can derive a private key for this time chunk from the master private key. The underlying HIBS enables email servers to publish succinct expiration information; instead of publishing all private keys for, say, one day, it is enough to publish the private key that corresponds to the node that encodes that day on the 3rd level.

Threat Models. In non-attributable email protocol, there are two types of attack scenarios: *Real-Time Attacks* and *After-the-fact Attacks*. In the first attack model, a *real-time attacker* can be the recipient of the message that immediately discloses received messages to the third parties. In the after-the-fact attack model, the recipient of the message is assumed to be honest at the time of email receipt, but can be compromised afterwards by an *after-the-fact attacker*; hence, the attacker can take a snapshot of all emails received in the past. More importantly, Specter *et al.* emphasize that it is impossible to defend against a real-time attacker without interaction. Their KeyForge protocol, as well as our *GroupForge* protocol described in the following paragraph, aim to address the after-the-fact attack scenario. For the real-time attack scenario, since interaction is needed, we believe that interactive protocols such as OTR [15] are more suitable for achieving deniability.

Table 2: Performance of $\text{DSKE}_{\mathsf{hash}}$ and $\text{DSKE}_{\mathsf{kzg}}$. Note that the probability of extracting the key in $\text{DSKE}_{\mathsf{hash}}$ is overwhelming in k, so the actual size of the extractable group can be smaller than $k = 64$.

Scheme	Group Size	Key Generation	Signing	Verification	Extraction
$\text{DSKE}_{\mathsf{hash}}(\mathsf{SHA256})$	$k = 64$	$451.583\mu s$	$17.625\mu s$	$160.458\mu s$	$45.083\mu s$
$\text{DSKE}_{\mathsf{kzg}}(\mathsf{BLS12\text{-}377})$	$k = 16$	$2.795ms$	$2.425ms$	$4.949ms$	$1.785ms$
	$k = 32$	$2.825ms$	$1.550ms$	$6.123ms$	$7.813ms$
	$k = 64$	$2.852ms$	$2.756ms$	$7.202ms$	$35.341ms$
	$k = 128$	$3.624ms$	$3.404ms$	$8.045ms$	$166.433ms$

6.2 GroupForge from DSKE

We now show how a Digital Signature with Key Extraction (DSKE) scheme, such as $\text{DSKE}_{\mathsf{poly}}$, can be used in a non-attributable email protocol [50]. More importantly, DSKE removes the requirement for email servers to periodically publish expiration information. We introduce GroupForge, an FFS scheme that builds on top of a DSKE scheme and achieves the same properties as KeyForge, while it does not require the email servers to publish *any* expiry information, i.e., algorithm $\mathsf{Expire}()$ requires only information already published by the email servers.

GroupForge uses a signature scheme Σ_{DSKE} with extractable sets of size k, a collision-resistant hash function, g, and a *local* clock $\mathsf{Time}()$ that returns the current time. GroupForge works as follows[2]:

- $\mathsf{KeyGen}(1^\lambda, \Delta, h, t)$: It takes as input the security parameter λ, the length of the time chunk Δ, the height of the Merkle tree $h \in \mathbb{N}$, and the starting time t. It calls $\Sigma_{\mathsf{DSKE}}.\mathsf{Setup}()$, and generates 2^h key pairs $\{(pk_i, sk_i)\}_{i\in[2^h]}$ using the algorithm $\Sigma_{\mathsf{DSKE}}.\mathsf{KeyGen}()$. The algorithm then uses $g(\cdot)$ to construct a Merkle tree from $\{pk_i\}_{i\in[2^h]}$. Let R be the root of this Merkle tree. Then it sets $PK = (R, \Delta, h, t)$ and $SK = (R, \Delta, h, t, \{sk_i\}_{i\in[2^h]})$, and outputs (SK, PK).

- $\mathsf{Sign}(SK, m)$: It parses $SK = (R, \Delta, h, t, \{sk_i\}_{i\in[2^h]})$, obtains the current time $t' \leftarrow \mathsf{Time}()$, and computes $\mathsf{leaf}_i = \lfloor (t' - t)/\Delta \rfloor$ (the leaf that corresponds to the current time chunk i). It computes $s = \Sigma_{\mathsf{DSKE}}.\mathsf{Sign}(sk_i, m)$, the authenticity proof proof_i for the path from leaf_i to the root R, and outputs the signature $\sigma = (s, pk_i, \mathsf{leaf}_i, \mathsf{proof}_i)$.

- $\mathsf{Verify}(PK, m, \sigma)$: It parses $PK = (R, \Delta, h, t)$, and $\sigma = (s, pk_i, \mathsf{leaf}_i, \mathsf{proof}_i)$, obtains the current time $t' \leftarrow \mathsf{Time}()$, computes $\mathsf{leaf} \leftarrow \lfloor (t' - t)/\Delta \rfloor$. If $\mathsf{leaf}_i \neq \mathsf{leaf}$, then returns 0. It validates proof_i starting from leaf_i; if invalid, it returns 0. Otherwise, returns $\Sigma_{\mathsf{DSKE}}.\mathsf{Verify}(pk_i, m, s)$.

- $\mathsf{Expire}(PK, \{(m_j, \sigma_j)\}_{j\in[k]})$: The inputs are the public key PK and k message-signature pairs $\{(m_j, \sigma_j)\}_{j\in[k]}$. The algorithm checks that the messages are pairwise different, i.e., for all $j_1, j_2 \in [k]$, $m_{j_1} \neq m_{j_2}$, that all signatures are valid, i.e., for all $j \in [k]$, $\mathsf{Verify}(PK, m_j, \sigma_j) = 1$, and that all signatures are created in the same time chunk, i.e., there exists $i \in [2^h]$ such that for all $j \in [k]$, $\sigma_j = (s_j, pk_i, \mathsf{leaf}_i, \mathsf{proof}_i)$. If the conditions do not hold, it returns $\perp$. Otherwise, it calls $\Sigma_{\mathsf{DSKE}}.\mathsf{Extract}(\{(m_j, \sigma_j)\}_{j\in[k]}, pk_i)$. If this call returns $\perp$, then $\mathsf{Expire}()$ also returns $\perp$. If it returns an extracted key sk_i, then $\mathsf{Expire}()$ returns the expiry information $h = (\mathsf{leaf}_i, \mathsf{proof}_i, sk_i)$.

[2]For simplicity we do not show parameters specific to the Σ_{poly} scheme, i.e., d and ℓ.

- $\mathsf{Forge}(PK, m, h)$: The inputs are the public key PK, a message m, and the expiry information $h = (\mathsf{leaf}_i, \mathsf{proof}_i, sk_i)$, for some $i \in [2^h]$. It computes $s = \Sigma_{\mathsf{DSKE}}.\mathsf{Sign}(sk_i, m)$ and returns the signature $\sigma = (s, pk_i, \mathsf{leaf}_i, \mathsf{proof}_i)$, where $pk_i = \mathsf{extractPK}(sk_i)$ is the public key of leaf_i.

Figure 2 gives a pictorial example of how GroupForge works.

Security. GroupForge satisfies the *correctness*, *unforgeability*, and *forgeability on expiry* properties of an FFS scheme.

Correctness follows from the underlying Σ_{DSKE} scheme and from correctness of the Merkle proofs: every signature $\sigma = (s, pk_i, \mathsf{leaf}_i, \mathsf{proof}_i)$ constructed with $\mathsf{Sign}()$ will be valid according to $\Sigma_{\mathsf{DSKE}}.\mathsf{Verify}()$, and proof_i will be a valid Merkle proof that pk_i is the public key that corresponds to leaf_i.

Unforgeability is also reduced to the unforgeability of Σ_{DSKE} through standard arguments on Merkle-based constructions [17]. Assume an adversary $\mathcal{A}$ that is given PK and attacks GroupForge. For each time chunk $i \in [2^h]$, $\mathcal{A}$ is allowed to make up to k signing queries. The challenger of $\mathcal{A}$ delegates these queries to Σ_{DSKE}. Assume a successful forgery $\sigma^* = (s^*, pk^*, \mathsf{leaf}^*, \mathsf{proof}^*)$, for some leaf leaf^*. Since Σ_{DSKE} is unforgeable, pk^* must be a public key under which s^* is a valid signature. In this case, however, the hash of pk^* and the Merkle proof proof^* can be used by the challenger to break the collision resistance of g.

Forgeability on expiry is proven in the following theorem.

Theorem 7. *GroupForge satisfies the* forgeability on expiry *property.*

Proof. Let $\mathcal{D}$ be an adversary that is given a signature σ_b, for $b \in \{0, 1\}$, and, as described in the definition of FFS, aims to distinguish the value of b. Signature σ_0 is created with the private key sk_i of some time chunk i, i.e., $\sigma_0 \leftarrow \mathsf{Sign}(sk_i, m)$. Signature σ_1 is created using the expiry information η, which in turn is computed from the additional material $\mathcal{T}$, i.e., $\eta \leftarrow \mathsf{Expire}(\mathcal{T})$ and $\sigma_1 \leftarrow \mathsf{Forge}(\eta, m)$. Let us assume that $\mathcal{T}$ contains k valid message-signature pairs, i.e., $\mathcal{T} = \{(m_j, \sigma_j)\}_{j \in [k]}$, that all m_j are pairwise different, and that all signatures are created in time chunk i, i.e., $\sigma_j = (s_j, pk_i, \mathsf{leaf}_i, \mathsf{proof}_i)$, for each $j \in [k]$. In this case, since the underlying DSKE scheme Σ_{DSKE} has (k, δ)-extractable sets, $\mathsf{Expire}(\mathcal{T})$ returns $h \neq \perp$ with probability δ. For this $h = (\mathsf{leaf}_i, \mathsf{proof}_i, sk'_i)$ we know from the properties of Σ_{DSKE} that $sk_i = sk'_i$, and $\mathsf{Forge}()$ will compute σ_1 by calling $s = \Sigma_{\mathsf{DSKE}}.\mathsf{Sign}(sk_i, m)$, which is identical to how σ_0 is created. Hence, $\mathcal{D}$ can only guess the value of b with probability $1/2$.

Overall, assuming that the signer has created enough message-signature pairs $\mathcal{T}$ with the properties described above, with probability δ, $\mathcal{D}$ can only at random guess the value of b, while with probability $1 - \delta$ (when $\mathsf{Expire}(\mathcal{T})$ returns $\perp$), the challenger of $\mathcal{D}$ cannot produce a valid σ_1 and $\mathcal{D}$ can correctly

guess b. The overall success probability of $\mathcal{D}$ is $P = 1 - \delta/2$. Observe that for the polynomial commitment-based DSKE scheme $\delta = 1 - \mathsf{negl}(\lambda)$, hence $P = 1/2 + \mathsf{negl}(\lambda)$, while for the hash-based DSKE scheme $\delta \geq 1 - \lambda/(2^{k-1}) + \mathsf{negl}(\lambda)$, hence P asymptotically approaches $1/2$ as k grows. □

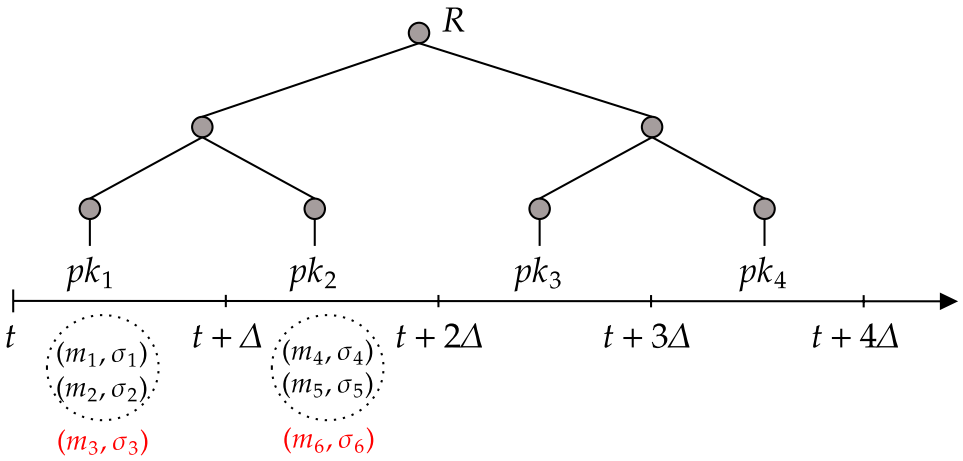

Figure 2: An example of GroupForge for $h = 2$ and between the interval $[t, t + 2^h \cdot \Delta]$. In this example, $\{(m_i, \sigma_i)\}_{i\in[2]}$ can potentially act as the deniable group for the message-signature pair (m_3, σ_3), and $\{(m_i, \sigma_i)\}_{i\in\{4,5\}}$ can act as the deniable group for the message-signature pair (m_6, σ_6).

Setting the Deniable Group Size k and Delay Δ. GroupForge requires that no recipients of k signatures collaborate within a time chunk Δ. This requirement can be made plausible by adjusting two variables. First, the size of the deniable group k can be set to a value large enough, for example, $\lceil n/2 \rceil$, where n is the number of recipient email servers. This would be equivalent to the assumption that no more than half of the nodes may be Byzantine, which is typical in distributed protocols. Second, time delay Δ can be set to a low value. Similar to that of KeyForge, we can assume an upper bound $\hat{\Delta}$ in the time required for email delivery and then set $\Delta = \hat{\Delta}$. Hence, even if the signature recipients collaborate, the forged signature has a high probability of reaching the recipient email server in the next time chunk.

Moreover, the value of k and the choice of the underlying DSKE scheme affect the probability of the algorithm $\mathsf{Forge}()$ outputting a signature and not $\perp$. For Σ_{poly} we have $\delta = 1$ for all $k \geq \ell + 1$, and the signer can choose the size of the group by selecting the degree of the polynomial accordingly. For Σ_{hash} the probability δ changes with k.

Table 3: Performance of GroupForge Constructions with $\Delta = 0.5h$.

Scheme	Parameters: Tree height (Duration)	Key Generation	Signing	Verification
$\text{GroupForge}_{\mathsf{hash}}$ ($\mathsf{SHA256}$)	$h = 14$ (0.93 years)	$7.407s$	$21.791\mu s$	$176.541\mu s$
	$h = 15$ (1.87 years)	$14.815s$	$34.708\mu s$	$177.416\mu s$
	$h = 16$ (3.74 years)	$29.631s$	$41.291\mu s$	$178.083\mu s$
	$h = 17$ (7.48 years)	$59.262s$	$45.916\mu s$	$179.374\mu s$
$\text{GroupForge}_{\mathsf{kzg}}$ ($\mathsf{SHA256}, \mathsf{BLS12\text{-}377}, k = 32$)	$h = 14$ (0.93 years)	$46.294s$	$1.554ms$	$6.139ms$
	$h = 15$ (1.87 years)	$92.588s$	$1.591ms$	$6.140ms$
	$h = 16$ (3.74 years)	$185.175s$	$1.611ms$	$6.141ms$
	$h = 17$ (7.48 years)	$370.350s$	$1.722ms$	$6.143ms$

7 Evaluation and Discussion

In this section, we evaluate the performance of both hash-based and polynomial commitment-based DSKE constructions, as well as the performance of GroupForge constructions. Finally, we discuss potential future directions for DSKE.

7.1 Evaluation

In this part, we evaluate the performance of Lamport signature-based DSKE ($\text{DSKE}_{\mathsf{hash}}$) and KZG-based DSKE scheme ($\text{DSKE}_{\mathsf{kzg}}$) as well as the performance of Groupforce constructions.

Testbed. We evaluate the performance of our schemes on a macOS Monterey machine with an Apple M1 chip (8-core, 3.2 GHz), 8 GB of RAM.

DSKE Constructions. We have implemented prototypes of our proposed constructions of $\text{DSKE}_{\mathsf{hash}}$ and $\text{DSKE}_{\mathsf{kzg}}$ in Rust. For our $\text{DSKE}_{\mathsf{hash}}$, we use the $\mathsf{SHA\text{-}256}$ implementation. For $\text{DSKE}_{\mathsf{kzg}}$, we adapt the KZG polynomial commitment implementation using the curve $\mathsf{BLS12\text{-}377}$ from `arkworks` library [3].

Table 2 shows the performance of $\text{DSKE}_{\mathsf{hash}}$ and $\text{DSKE}_{\mathsf{kzg}}$. We observe that the running time of the functions of $\text{DSKE}_{\mathsf{hash}}$ is less than $1ms$. The extraction time of $\text{DSKE}_{\mathsf{kzg}}$ appears as approximately a quadratic function of the group size k.

GroupForge Constructions. We also implement two GroupForge constructions based on $\text{DSKE}_{\mathsf{hash}}$ and $\text{DSKE}_{\mathsf{kzg}}$ respectively. We leverage the Merkle tree library using $\mathsf{SHA\text{-}256}$ from `arkworks` [4] to construct our GroupForge. We implement $\text{GroupForge}_{\mathsf{hash}}$ with $\text{DSKE}_{\mathsf{hash}}$, and $\text{GroupForge}_{\mathsf{kzg}}$ with $\text{DSKE}_{\mathsf{kzg}}$ using $\mathsf{BLS12\text{-}377}$ curve and the group size $k = 32$. Table 3 summarizes the performance of GroupForge constructions. We evaluate our constructions with Merkle tree heights of 14, 15, 16, and

[3] https://github.com/arkworks-rs/poly-commit

[4] https://github.com/arkworks-rs/crypto-primitives

17, which correspond to a duration of 0.93, 1.87, 3.74 and 7.48 years respectively (given a time chunk of $\Delta = 0.5h$). For both $\text{GroupForge}_{\mathsf{hash}}$ and $\text{GroupForge}_{\mathsf{kzg}}$, the most expensive computation is the key generation. This is because we need to generate 2^h key pairs, where h is the Merkle tree height.

7.2 Discussion

In this part, we discuss relevant properties and examine potential future directions and improvements regarding the constructions presented in the paper.

Improving GroupForge with Forward-Secure Signature Schemes. Forward-secure signature schemes (FSS) [5] [7] allow signers to derive future keys from past keys while preventing users from deriving past keys from future keys. This property of FSS can be combined with GroupForge to demonstrate that if there is a deniable group at some point, the same group can be used as a deniable proof for all message-signature pairs before that point. In particular, to achieve this property, the signer can produce several FSS key pairs from a master private key, and use them in reverse order in the leaves of the Merkle hash tree. For instance, a private key sk_i used in the interval $[t_i, t_i + \Delta]$ can produce all previous private keys sk_j used before t_i. Hence, it is not difficult to see that, if a deniable group exists at some time t', the same group can derive all private keys before t'. Figure 3 gives a high-level example of how FSS can help improve GroupForge. For concrete instantiations of FSS, hash-based XMSS [16] can be a candidate for the hash-based DSKE construction, and we leave the development of forward-secure polynomial commitment-based construction as an interesting open question for the future.

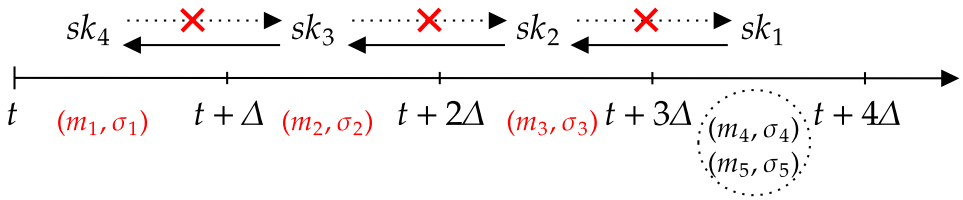

Figure 3: An example of GroupForge using FSS. The group $\{(m_i, \sigma_i)\}_{i\in\{4,5\}}$ can act as a deniable group for all message-signature pairs in the past, $\{(m_i, \sigma_i)\}_{i\in[3]}$.

Trusted setup in PCS. The KZG scheme used in $\text{DSKE}_{\mathsf{poly}}$ requires a trusted setup to generate the public parameters. Although the scheme can

[5] FSS is not to be confused with forward-forgeable signatures

leverage a trusted third party to run the setup, such a reliance on a centralized and trusted party is typically not welcome in various settings. To remove this trusted assumption, practitioners and the academic community have developed practical solutions to securely generate these parameters. Since PCS play an important role in recent developments of Non-Interactive Zero-knowledge Proof (e.g., zkSnark), various works [20, 21, 39] have also proposed improved zkSnarks with a universal setup, which can be used for any bounded-degree polynomials. Participants can also leverage a distributed key generation protocol [26, 31] to generate the KZG keys in a distributed way. With new advances in the field [18], one can also deploy PCS that does not require trusted setup.

Time-Agnostic Forgeability. Although not explicitly mentioned, making a synchronous assumption on the communication is required to ensure parties (i.e., group of size k) receive their signatures in the respective time chuck. Moreover, the proper operation of the system relies on parties having access to a local clock that advances at the same pace. The original work of [49] requires the stronger assumption of a shared global clock. However, we observe that the forgeability aspect of the DSKE is generally independent of the notion of time as it comes from the availability of a sufficient number of signatures for key extraction. This essentially opens up the door to use DSKE for adding forgeability property in settings where reliance on time is not desirable.

DSKE as a Unique Signature. Apart from the aspect of key extractability, $\mathsf{DSKE}_{\mathsf{poly}}$ also features uniqueness property, i.e., there is only one valid signature for any combination of the message and public key. This property is of importance in applications where we need to ensure the consistency of the produced signature like for generating shared randomness [32]. For instance, BLS [13] is a unique-signature scheme while ECDSA [34] is not.

Comparison with Adaptor Signatures. A related notion to DSKE is that of adaptor signatures [4, 24] that bind the authorization of a message and the extraction of a secret value. The extractability here, however, is with respect to a hard relation representing a witness and not the underlying private key. Observe that, setting the private key as witness by the signer enables the receiver to extract it using only two signatures, namely the pre-signature and the full signature. Moreover, it turns out that adaptor signatures cannot offer uniqueness [23]. Adaptor signatures are used in constructing secure payment channels in blockchain protocols; hence, it can be an interesting future work to use DSKE as a building block in these settings.

Compacting the Keys and Signatures. We can also consider extending $\mathsf{DSKE}_{\mathsf{poly}}$ to multi-signature schemes where a group of individuals, each with their own key pairs, collectively sign a message verified under the respective set of public keys. In recent years, there has been a surge of interest in de-

veloping signature schemes that support aggregation property for the public key and signature to reduce the size of the blockchain ledger [12]. Due to the features provided by the underlying KZG polynomial commitment scheme such as homomorphism, $\mathsf{DSKE}_{\mathsf{kzg}}$ also takes advantage of key and signature aggregation, offering a high level of space efficiency.

One the Size of the private keys. The private key in $\mathsf{DSKE}_{\mathsf{poly}}$, i.e., the polynomial $f(X)$, contains ℓ elements of $\mathbb{F}$, i.e., the coefficients $f_0, \ldots, f_{\ell-1}$. A possible optimization is to employ a pseudorandom function (PRF) $F : \{0,1\}^\lambda \times \mathbb{Z} \to \mathbb{F}$ and pick a $k \in \{0,1\}^\lambda$ as the private key. The coefficients are derived as $f_i = F(k, i)$, for $i \in 0, \ldots, \ell - 1$. In the security proof the challenger now chooses k uniformly at random and uses the PRF to derive $f(X)$. From the security definition of PRF, $f(X)$ still looks random to the simulator $\mathcal{B}$. Observe, however, that $\mathsf{Extract}()$ would still return all ℓ coefficients. Similarly, the KeyGen algorithm in GroupForge generates 2^h private keys for the underlying Σ_{DSKE} scheme, i.e., $\{sk_i\}_{i\in[2^h]}$, which are then stored in the private key SK. Using again a PRF $F' : \{0,1\}^\lambda \times \mathbb{Z} \to \mathbb{Z}$ and a private key $k' \in \{0,1\}^\lambda$ we can generate them as $sk_i = F'(k', i)$, for $i \in 1, \ldots, 2^h$, and store only k' in SK. As long as 2^h is polynomial in λ the outputs of F' are pseudorandom and security holds. We leave the formalization of these optimizations as future work.

Applications Beyond Non-attributable Emails. DSKE can also facilitate other application scenarios where the need for authenticity is momentary and long-term deniability could be desirable to signers, for example, in electronic voting and monetary donations. It has been argued that, even if future adversaries cannot go back to change the integrity and results of a finished election, the privacy requirements of voters in the face of retaliation are everlasting [25]. Similarly, leaked donation records have reportedly caused individuals to be targeted and threatened [1], and transaction deniability might be desirable in such scenarios.

k-Conditional Anonymity. By design, DKSE offers extractability merely from the existence of a set of k signatures. Hence, in applications where the creation of only a certain number of signatures is expected our DSKE schemes enable extractability at no additional cost (e.g., computing VDFs, publishing expired keys). Concretely, $\mathsf{DSKE}_{\mathsf{kzg}}$ might be useful in designing systems with *k-conditional anonymity* [19, 6]. In such systems, an issuer is responsible to provide a user with a set of credentials on some attributes. When a restricted usage of credentials is desired, the DSKE can come into play by enabling the extraction of user's private key in case she happens to take advantage of a higher number of credentials than is supposed to.

Disincentivizing Equivocation. In leader-based consensus protocols, the leader broadcasts a signed block on each epoch. If the leader sends conflicting blocks to different parties, all signed with the private key of the same epoch, then this private key can be extracted. Similarly, DSKE can po-

tentially replace adaptor signature used in payment channels to mitigate equivocation between senders and receivers.

8 Conclusion

In this paper, we defined the concept of a signature scheme with key extraction. We presented two concrete constructions based on the Lamport Signature scheme and polynomial commitment schemes. We formally proved the security of our constructions and showed that signers can always achieve deniability by showing a set of signatures used to reconstruct old private keys. More importantly, we proposed GroupForge by combining DSKE with Merkle hash trees and timestamps to offer properties similar to KeyForge [50], and GroupForge can be used in non-attributable email protocols to remove the requirement for email servers to constantly publish old key material. Finally, to the best of our knowledge, our work is the first that introduces the notion of signature with extractable private key, and we conjecture that DSKE can be used in other protocols that require short-term authenticity.

References

[1] Threats close stella luna gelato café after owner's name appears in givesendgo data leak. `https://ottawacitizen.com/news/local-news/threats-close-stella-luna-gelato-cafe-after-owners-name-appears-in-givesendgo-data-leak`.

[2] E Allman, Jon Callas, M Delany, Miles Libbey, J Fenton, and M Thomas. Domainkeys identified mail (dkim) signatures. Technical report, RFC 4871, May, 2007.

[3] Arasu Arun, Joseph Bonneau, and Jeremy Clark. Short-lived zero-knowledge proofs and signatures. 2019. `https://ia.cr/2019/390`.

[4] Lukas Aumayr, Oguzhan Ersoy, Andreas Erwig, Sebastian Faust, Kristina Hostáková, Matteo Maffei, Pedro Moreno-Sanchez, and Siavash Riahi. Generalized channels from limited blockchain scripts and adaptor signatures. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 635–664. Springer, 2021.

[5] Foteini Baldimtsi, Aggelos Kiayias, Thomas Zacharias, and Bingsheng Zhang. Indistinguishable proofs of work or knowledge. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 902–933. Springer, 2016.

[6] Foteini Baldimtsi and Anna Lysyanskaya. Anonymous credentials light. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pages 1087–1098, 2013.

[7] Mihir Bellare and Sara K Miner. A forward-secure digital signature scheme. In *Annual international cryptology conference*, pages 431–448. Springer, 1999.

[8] Mihir Bellare, Bertram Poettering, and Douglas Stebila. Deterring certificate subversion: efficient double-authentication-preventing signatures. In *Public-Key Cryptography–PKC 2017: 20th IACR International Conference on Practice and Theory in Public-Key Cryptography, Amsterdam, The Netherlands, March 28-31, 2017, Proceedings, Part II 20*, pages 121–151. Springer, 2017.

[9] Stephen E Blythe. Digital signature law of the united nations, european union, united kingdom and united states: Promotion of growth in e-commerce with enhanced security. *Richmond Journal of Law & Technology*, 11(2):6, 2005.

[10] Dan Boneh, Joseph Bonneau, Benedikt Bünz, and Ben Fisch. Verifiable delay functions. In *Annual international cryptology conference*, pages 757–788. Springer, 2018.

[11] Dan Boneh, Justin Drake, Ben Fisch, and Ariel Gabizon. Halo infinite: Proof-carrying data from additive polynomial commitments. In *Annual International Cryptology Conference*, pages 649–680. Springer, 2021.

[12] Dan Boneh, Manu Drijvers, and Gregory Neven. Compact multi-signatures for smaller blockchains. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 435–464. Springer, 2018.

[13] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the weil pairing. In *International conference on the theory and application of cryptology and information security*, pages 514–532. Springer, 2001.

[14] Dan Boneh and Victor Shoup. A graduate course in applied cryptography. *Draft 0.5*, 2020.

[15] Nikita Borisov, Ian Goldberg, and Eric Brewer. Off-the-record communication, or, why not to use pgp. In *Proceedings of the 2004 ACM workshop on Privacy in the electronic society*, pages 77–84, 2004.

[16] Johannes Buchmann, Erik Dahmen, and Andreas Hülsing. Xmss-a practical forward secure signature scheme based on minimal security assumptions. In *International Workshop on Post-Quantum Cryptography*, pages 117–129. Springer, 2011.

[17] Johannes Buchmann, Erik Dahmen, and Andreas Hülsing. XMSS - A Practical Forward Secure Signature Scheme Based on Minimal Security Assumptions. In Bo-Yin Yang, editor, *Post-Quantum Cryptography*, pages 117–129, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[18] Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Greg Maxwell. Bulletproofs: Short proofs for confidential transactions and more. In *2018 IEEE symposium on security and privacy (SP)*, pages 315–334. IEEE, 2018.

[19] Jan Camenisch, Susan Hohenberger, Markulf Kohlweiss, Anna Lysyanskaya, and Mira Meyerovich. How to win the clonewars: efficient periodic n-times anonymous authentication. In *Proceedings of the 13th ACM conference on Computer and communications security*, pages 201–210, 2006.

[20] Matteo Campanelli, Dario Fiore, and Anaïs Querol. Legosnark: Modular design and composition of succinct zero-knowledge proofs. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pages 2075–2092, 2019.

[21] Alessandro Chiesa, Yuncong Hu, Mary Maller, Pratyush Mishra, Noah Vesely, and Nicholas Ward. Marlin: Preprocessing zksnarks with universal and updatable srs. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 738–768. Springer, 2020.

[22] David Derler, Sebastian Ramacher, and Daniel Slamanig. Short double- and n-times-authentication-preventing signatures from ecdsa and more. In *2018 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 273–287. IEEE, 2018.

[23] Andreas Erwig, Sebastian Faust, Kristina Hostáková, Monosij Maitra, and Siavash Riahi. Two-party adaptor signatures from identification schemes. In *IACR International Conference on Public-Key Cryptography*, pages 451–480. Springer, 2021.

[24] Muhammed F Esgin, Oğuzhan Ersoy, and Zekeriya Erkin. Post-quantum adaptor signatures and payment channel networks. In *European Symposium on Research in Computer Security*, pages 378–397. Springer, 2020.

[25] Huangyi Ge, Sze Yiu Chau, Victor E Gonsalves, Huian Li, Tianhao Wang, Xukai Zou, and Ninghui Li. Koinonia: verifiable e-voting with long-term privacy. In *Proceedings of the 35th Annual Computer Security Applications Conference*, pages 270–285, 2019.

[26] Rosario Gennaro, Stanisław Jarecki, Hugo Krawczyk, and Tal Rabin. Secure distributed key generation for discrete-log based cryptosystems. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 295–310. Springer, 1999.

[27] Shafi Goldwasser, Silvio Micali, and Ronald L Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on computing*, 17(2):281–308, 1988.

[28] Matthew Green. Ok Google: please publish your DKIM secret keys. `https://blog.cryptographyengineering.com/2020/11/16/ok-google-please-publish-your-dkim-secret-keys/`.

[29] Leon Groot Bruinderink and Andreas Hülsing. "oops, i did it again"–security of one-time signatures under two-message attacks. In *International Conference on Selected Areas in Cryptography*, pages 299–322. Springer, 2017.

[30] Jens Groth and Markulf Kohlweiss. One-out-of-many proofs: Or how to leak a secret and spend a coin. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 253–280. Springer, 2015.

[31] Kobi Gurkan, Philipp Jovanovic, Mary Maller, Sarah Meiklejohn, Gilad Stern, and Alin Tomescu. Aggregatable distributed key generation. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 147–176. Springer, 2021.

[32] Timo Hanke, Mahnush Movahedi, and Dominic Williams. Dfinity technology overview series, consensus system. *arXiv preprint arXiv:1805.04548*, 2018.

[33] Markus Jakobsson, Kazue Sako, and Russell Impagliazzo. Designated verifier proofs and their applications. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 143–154. Springer, 1996.

[34] Don Johnson, Alfred Menezes, and Scott Vanstone. The elliptic curve digital signature algorithm (ecdsa). *International journal of information security*, 1(1):36–63, 2001.

[35] Nikolaos Karanikolas. Digital signature legality in different jurisdictions: Legally binding issues. 2019.

[36] Aniket Kate, Gregory M Zaverucha, and Ian Goldberg. Constant-size commitments to polynomials and their applications. In *International conference on the theory and application of cryptology and information security*, pages 177–194. Springer, 2010.

[37] Aniket Kate, Gregory M. Zaverucha, and Ian Goldberg. Constant-size commitments to polynomials and their applications. In *Advances in Cryptology - ASIACRYPT 2010 - 16th International Conference on the Theory and Application of Cryptology and Information Security*, volume 6477, pages 177–194. Springer, 2010.

[38] Leslie Lamport. Constructing digital signatures from a one way function. 1979.

[39] Mary Maller, Sean Bowe, Markulf Kohlweiss, and Sarah Meiklejohn. Sonic: Zero-knowledge snarks from linear-size universal and updatable structured reference strings. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pages 2111–2128, 2019.

[40] Stephen Mason. *Electronic signatures in law.* University of London Press, 2016.

[41] Ralph C Merkle. A certified digital signature. In *Conference on the Theory and Application of Cryptology*, pages 218–238. Springer, 1989.

[42] Silvio Micali and Leonid Reyzin. Improving the exact security of digital signature schemes. *Journal of Cryptology*, 15(1):1–18, 2002.

[43] Adrian Perrig. The biba one-time signature and broadcast authentication protocol. In *Proceedings of the 8th ACM Conference on Computer and Communications Security*, pages 28–37, 2001.

[44] Bertram Poettering and Douglas Stebila. Double-authentication-preventing signatures. *International Journal of Information Security*, 16:1–22, 2017.

[45] Michael O Rabin. Transaction protection by beacons. *Journal of Computer and System Sciences*, 27(2):256–267, 1983.

[46] Leonid Reyzin and Natan Reyzin. Better than biba: Short one-time signatures with fast signing and verifying. In *Australasian Conference on Information Security and Privacy*, pages 144–153. Springer, 2002.

[47] Ronald L Rivest, Adi Shamir, and Yael Tauman. How to leak a secret. In *International conference on the theory and application of cryptology and information security*, pages 552–565. Springer, 2001.

[48] Phillip Rogaway and Thomas Shrimpton. Cryptographic hash-function basics: Definitions, implications, and separations for preimage resistance, second-preimage resistance, and collision resistance. In *International workshop on fast software encryption*, pages 371–388. Springer, 2004.

[49] Michael Specter, Sunoo Park, and Matthew Green. Keyforge: Mitigating email breaches with forward-forgeable signatures, 2019. `https://ia.cr/2019/390`.

[50] Michael A Specter, Sunoo Park, and Matthew Green. KeyForge:non-attributable email from Forward-Forgeable Signatures. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 1755–1773, 2021.

[51] Ron Steinfeld, Huaxiong Wang, and Josef Pieprzyk. Efficient extension of standard schnorr/rsa signatures into universal designated-verifier signatures. In *International Workshop on Public Key Cryptography*, pages 86–100. Springer, 2004.

[52] Ewa Syta, Philipp Jovanovic, Eleftherios Kokoris Kogias, Nicolas Gailly, Linus Gasser, Ismail Khoffi, Michael J Fischer, and Bryan Ford. Scalable bias-resistant distributed randomness. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 444–460. Ieee, 2017.

[53] Benjamin Wesolowski. Efficient verifiable delay functions. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 379–407. Springer, 2019.

A KZG-Based DSKE Scheme

We now show $\mathrm{DSKE}_{\mathsf{kzg}}$, a concrete construction of $\mathrm{DSKE}_{\mathsf{poly}}$ from the Π_{kzg} polynomial commitment scheme [36] (cf. Section 2.3).

- $\mathsf{Setup}(1^\lambda, d)$: Run $\Pi_{\mathsf{kzg}}.\mathsf{Setup}(1^\lambda, d)$ to obtain $ck = \{g^{\alpha^i} \in G\}_{i\in[d]}$ and $vk = h^\alpha \in G$. Return ck, vk, d, and a collision-resistant hash function $h : \mathcal{M} \to \mathbb{Z}_p$.

- $\mathsf{KeyGen}(1^\lambda, \ell)$: Sample $f(X) \xleftarrow{\$} \mathbb{F}^\ell(X)$ as an ℓ-degree polynomial. Return $pk = C_f = \Pi_{\mathsf{kzg}}.\mathsf{Com}(ck, f(X)) = g^{f(\alpha)} \in G$ and $sk = f(X)$.

- $\mathsf{Sign}(sk, m)$: Parse $sk = f(X)$, compute $x = h(m)$, and output the signature $\sigma = (\pi, y) = \Pi_{\mathsf{kzg}}.\mathsf{Open}(ck, C_f, x, f(X)) \in G \times \mathbb{Z}_p$.

- $\mathsf{Verify}(pk, m, \sigma)$: Parse $pk = C_f$ and $\sigma = (\pi, y)$, compute $x = h(m)$, and output $\Pi_{\mathsf{kzg}}.\mathsf{Check}(vk, C_f, x, y, \pi) \in \{0, 1\}$.

- $\mathsf{Extract}(\{(m_i, \sigma_i)\}_{i\in[k]}, pk)$: As in the general construction, check the validity of σ_i, for $i \in [k]$, interpolate $\phi(X)$ from points (x_i, y_i), and output $sk' = \phi(X) \in \mathbb{F}^\ell(X)$.

Theorem 8 (Unforgeability)**.** *The scheme $DSKE_{\mathsf{kzg}}$ is existentially unforgeable under an ℓ-times adaptive chosen-message attack, under the same cryptographic assumptions as the Π_{kzg} polynomial commitment scheme.*

Theorem 9 (Extractable set)**.** *The scheme $DSKE_{\mathsf{kzg}}$ has an extractable set of size k for any $k \geq \ell + 1$ with probability $\delta = 1 - \mathsf{negl}(\lambda)$, under the same cryptographic assumptions as the Π_{kzg} polynomial commitment scheme.*

The proofs for Theorems 8 and 9 follow directly from Theorems 5 and 6, because Π_{kzg} satisfies the properties required from the polynomial commitment scheme Π in the generic construction.