An Equivalence Between Attribute-Based Signatures and Homomorphic Signatures, and New Constructions for Both

Rotem Tsabary*

December 31, 2017

Abstract

In Attribute-Based Signatures (ABS; first defined by Maji, Prabhakaran and Rosulek, CT-RSA 2011) an authority can generate multiple signing keys, where each key is associated with an attribute x. Messages are signed with respect to a constraint f, such that a key for x can sign messages respective to f only if $f(x) = 0$. The security requirements are unforgeability and key privacy (signatures should not expose the specific signing key used). In (single-hop) Homomorphic Signatures (HS; first defined by Boneh and Freeman, PKC 2011), given a signature for a data-set x, one can evaluate a signature for the pair $(f(x), f)$, for functions f. In *context-hiding* HS, evaluated signatures do not reveal information about the original (pre-evaluated) signatures.

In this work we start by showing that these two notions are in fact equivalent. The first implication of this equivalence is a new lattice-based ABS scheme for polynomial-depth circuits, based on the HS construction of Gorbunov, Vaikuntanathan and Wichs (GVW; STOC 2015).

We then construct a new ABS candidate from a worst case lattice assumption (SIS), with different parameters. Using our equivalence again, now in the opposite direction, our new ABS implies a new lattice-based HS scheme with different parameter trade-off, compared to the aforementioned GVW.

*Weizmann Institute of Science, rotem.tsabary@weizmann.ac.il. Supported by the Israel Science Foundation (Grant No. 468/14) and Binational Science Foundation (Grant No. 712307).

Contents

1 Introduction

In a standard digital signature scheme an authority generates a public verification key vk and a secret signing key sk. Given sk, it is possible to sign any message, and signatures can be verified publicly with vk. Recent works study more powerful notions of digital signatures, where the authority can generate multiple signing keys, each with limited signing permissions. An example use case is when an organization wants to allow its employees to sign on behalf of its name, while controlling which messages each employee can sign. A signature should not reveal any information about the signing permissions of the signer, other than whether he is allowed to sign the message corresponding to the same signature. In stronger notions, the signature should not reveal any information about the *identity* of the signer. Main notions of this form are attribute-based signatures (ABS) [MPR11], policy-based signatures (PBS) [BF14], constrained signatures (CS) [BZ14] and functional signatures (FS) [BGI13]. In this work we use a slightly modified definition of *constrained signatures*, with two flavors that capture ABS and PBS for languages in $\mathbf{P}$.

In a homomorphic signatures (HS) scheme, given a signature for a data-set x, one can evaluate a signature for the pair $(f(x), f)$, for any f in the supported function space of the scheme. Context-hiding HS has the security guarantee that an evaluated signature does not reveal information about the original (pre-evaluated) signature. In particular, it does not reveal x. Context-hiding homomorphic signatures are useful, for example, when one wants to prove that he has a signature for a data-set which satisfies some condition, without revealing the data-set itself. We show in this work that CS is equivalent to context-hiding 1-hop HS.

1.1 Overview

Two flavors of CS will be alternately used throughout this work. In *key-policy constrained signatures*, each signing key sk_f is associated with a circuit $f : \{0,1\}^* \to \{0,1\}$, which we refer to as the *constraint*, and a key sk_f can sign an *attribute* $x \in \{0,1\}^*$ only if $f(x) = 0$. In *message-policy constrained signatures*, each key is associated with an attribute $x \in \{0,1\}^*$, and a key sk_x can sign a constraint $f : \{0,1\}^* \to \{0,1\}$ only if $f(x) = 0$. Message-policy CS is equivalent to attribute-based signatures, and key-policy CS is equivalent policy-based signatures for languages in $\mathbf{P}$ [1]. When presented as two flavors of a single primitive, we can take advantage of the similarities and alternately use the definition that best fits the context. Note that the flavors are interchangeable up to switching the constraint space and attribute space.

Security. We consider two aspects of security – unforgeability and key privacy. Unforgeability requires that an adversary cannot sign a message which it does not have a permission to sign, even after seeing other signatures. We also define a relaxed notion where the adversary has only a single key, and a selective notion where the adversary has to announce the message for which it is going to forge a signature before seeing any public data. Key privacy bounds the information revealed by a signature regarding the key that was used to produce it. In the strongest notion, *key-hiding* privacy, the signature completely hides the key. In particular, it is impossible to determine whether two signatures were derived from the same key. In *constraint-hiding* privacy (or *attribute-hiding* privacy, in the message-policy flavor) we only aim to hide the constraint (or to hide the attribute,

[1] The original definition of ABS [MPR11] (PBS [BF14]) considers an additional message space $\mathcal{M}$, where messages $m \in \mathcal{M}$ are signed respective to an attribute (a policy). The two definitions are equivalent since m can always be encoded into the signed attribute (policy).

in the message-policy flavor), possibly leaving the identity of the signing key public. We note that without any privacy requirements, CS are trivial to achieve using standard signatures.

Delegation. A CS scheme can be extended to support key delegation. In this setting, a party with a singing key sk_f can derive a signing key $\mathsf{sk}_{(f,g)}$ that is authorized to sign a message x only when $f(x) = 0$ *and* $g(x) = 0$. Note that the permissions of $\mathsf{sk}_{(f,g)}$ are never stronger than the permissions of sk_f, since otherwise the scheme is forgeable.

Motivation. CS is weaker than PBS for **NP** but strong enough for some of the motivations that lead to the study of PBS, such as constructing group signatures and attribute-based signatures. See the applications discussion in [BF14] for details. We exploit this gap and construct CS with a different approach than previous results that were using variations of NIZK. Indeed, as noted in [BF14], PBS for general languages in **NP** implies simulation-extractable NIZK proofs. We also see in this work a contribution to the understanding of homomorphic signatures – prior to this work there was only a single known construction of (leveled) fully HS [GVW15].

1.2 Results

Unforgeability Amplification. In our first construction we assume a (key-policy) CS scheme with *single-key-selective unforgeability*. This notion is captured by a security game where the adversary is only allowed to query for a single key sk_f, and it has to announce f before seeing any public data. It wins if it manages to forge a signature for an attribute x that is not authorized by f, i.e. where $f(x) = 1$. We use a standard signatures scheme to construct a (key-policy) CS scheme with full unforgeability. The downside of this general amplification is the loss in key privacy – while the new CS scheme is constraint-hiding (i.e. it hides the functionality of the signing key, as long as the underlying CS scheme does as well), signatures reveal other key-specific information and therefore it is not key-hiding (i.e. one can learn from a signature the identity of the signing key). The amplification maintains the delegation properties of the underlying CS scheme.

Equivalence of CS and Homomorphic Signatures. We first construct a (message-policy) CS scheme which is *single-key-selective unforgeable* and *key-hiding*, from context-hiding 1-hop homomorphic signatures. [GVW15] construct a context-hiding HS scheme which is secure under the Short Integer Solution (SIS) hardness assumption. When used as the underlying HS scheme to our construction, this results in a SIS-based (message-policy) CS scheme with bounded attribute space, and constraint space of boolean circuits with bounded depth. In the other direction, we construct a selectively-unforgeable context-hiding 1-hop HS scheme from a single-key-selective unforgeable key-hiding (message-policy) CS scheme. As shown in [GVW15], it is possible to amplify the unforgeability of such HS scheme to the adaptive notion.

CS from Lattice Trapdoors. We construct a (key-policy) CS scheme from lattice trapdoors, which is *message-selective unforgeable* and *key-hiding*. The key privacy is statistical, and the unforgeability relies on the Short Integers Solution (SIS) hardness assumption. The construction supports attribute space of fixed size and constraint space of boolean circuits with bounded depth. When translated to the message-policy flavor, the attribute space is unbounded and the policy space is bounded in depth and size.

A New Homomorphic Signatures Construction. An immediate conclusion of the above two results is a new lattice-based (leveled) fully homomorphic signatures scheme, where fresh signatures are of fixed size (independent of the signed data-set size), and evaluated signatures grow with the size of the policy description. It means that for any policy with a short description succinctness is maintained.

Two New CS Constructions. Combining the first two results gives a new CS construction – first construct the HS-based (message-policy) CS scheme which is single-key-selective unforgeable, and then amplify it to full unforgeability, while compromising on key privacy. We summarize the different properties of this CS construction and the lattice-based CS construction in the table below. Note that the HS-based scheme is presented in the message-policy flavor, and the lattice-based scheme is presented in the key-policy flavor. Implementing each of them in the opposite flavor will result in a constraint space of bounded depth *and size*, and an *unbounded* attribute space.

	HS-based message-policy CS	Lattice-based key-policy CS
Assumption	SIS	SIS
Attribute Space	fixed	fixed
Constraint Space	bounded depth	bounded depth
Unforgeability	full	message-selective
Privacy	constraint-hiding	key-hiding
Supports Delegation	no	yes

1.3 Technical Overview

Definition of CS. A (key-policy) CS scheme consists of 4 algorithms ($\mathsf{Setup}, \mathsf{Keygen}, \mathsf{Sign}, \mathsf{Ver}$). Setup is an initialization algorithm that generates a verification key vk and a master signing key msk. Keygen produces constrained signing keys – it takes as input the master signing key msk and a constraint f, and outputs a constrained key sk_f. The signing algorithm Sign takes as input an attribute x and a constrained singing key sk_f, and outputs a signature σ_x, which is valid if and only if $f(x) = 0$. The verification algorithm Ver takes an attribute x and a signature σ_x, and either accepts or rejects.

Unforgeability Amplification. We now give a brief description of the amplification. Assume a (key-policy) constrained signatures scheme CS' which is single-key-selective unforgeable, constraint-hiding and possibly supports delegation. Let S be an existentially unforgeable standard signatures scheme. The construction is as follows. In Setup, the authority initializes S and sets $(\mathsf{vk}, \mathsf{msk}) = (\mathsf{S.vk}, \mathsf{S.sk})$. Every time a key is generated, the authority initializes a fresh instance of CS' and generates a constrained key for the desired f under this instance: $(\mathsf{CS}'.\mathsf{vk}', \mathsf{CS}'.\mathsf{sk}'_f)$. It also generates a fresh instance of S : $(\mathsf{S.vk}'', \mathsf{S.sk}'')$. The authority then signs $(\mathsf{CS}'.\mathsf{vk}', \mathsf{S.vk}'')$ under the standard scheme S using $\mathsf{msk} = \mathsf{S.sk}$ and gets $\mathsf{S}.\sigma_{(\mathsf{vk}', \mathsf{vk}'')}$. The constrained key is therefore $\mathsf{sk}_f = (\mathsf{CS}'.\mathsf{vk}, \mathsf{CS}'.\mathsf{sk}_f, \mathsf{S.vk}'', \mathsf{S.sk}'', \mathsf{S}.\sigma_{(\mathsf{vk}', \mathsf{vk}'')})$. To sign an attribute x with a key of this form, one signs x with $(\mathsf{CS}'.\mathsf{vk}, \mathsf{CS}'.\mathsf{sk}_f)$, signs x with $(\mathsf{S.vk}'', \mathsf{S.sk}'')$ and outputs these signatures along with $\mathsf{S}.\sigma_{(\mathsf{vk}', \mathsf{vk}'')}$. Verification is done by verifying the signatures for x under $\mathsf{CS}'.\mathsf{vk}'$ and $\mathsf{S.vk}''$, and verifying $\mathsf{S}.\sigma_{(\mathsf{vk}', \mathsf{vk}'')}$ under $\mathsf{S.vk}$. Since for each instance of CS' the authority only generates a single key, the unforgeability for each such instance is maintained. The existential unforgeability of S guarantees that it is not possible to forge a signature for an instance of CS' that was not initialized

by the authority. Note that $\mathsf{CS}'.\mathsf{vk}$ is a part of the signature, and since this value is different for each key, it reveals the identity of the key. For that reason the construction is not key-hiding but solely constraint-hiding.

CS from Homomorphic Signatures. The construction of (message-policy) CS from context-hiding HS works as follows. The CS authority initializes the HS scheme. In order to produce a CS key for an attribute x, it signs x under the HS scheme and outputs $\mathsf{sk}_x = \mathsf{HS}.\sigma_x$. A signature for a policy f is derived from sk_x by homomorphically evaluating f on $\mathsf{HS}.\sigma_x$. This results in an HS signature for the pair $(f, f(x))$. In order to verify one checks the validity of the HS signature, and that $f(x) = 0$. The context-hiding property of HS ensures that $\sigma_{(f,f(x))}$ reveals nothing about σ_x, and thus the construction is key-hiding.

Homomorphic Signatures from CS. The construction of context-hiding 1-hop HS from (message-policy) CS works as follows. The HS authority initializes the CS scheme. In order to sign a data-set x, generate a CS key for the attribute x and outputs $\sigma_x = \mathsf{CS}.\mathsf{sk}_x$. To homomorphically evaluate a function f on a signature σ_x, first compute $y = f(x)$, then define the function f_y that on input x' outputs 0 if and only if $f(x') = y$. Sign the constraint f_y under the CS scheme (using $\mathsf{CS}.\mathsf{sk}_x$) and output this CS signature: $\mathsf{HS}.\sigma_{(f,y)} = \mathsf{CS}.\sigma_{f_y}$. In order to verify one checks the validity of the CS signature. The key-hiding property of CS ensures that $\mathsf{CS}.\sigma_{f_y}$ reveals nothing about $\mathsf{CS}.\mathsf{sk}_x$, and thus the construction is context-hiding.

CS from Lattice Trapdoors. We use techniques that were developed in [GVW13, BGG+14] for the purpose of attribute-based encryption (ABE). Let ℓ be the attribute length, i.e. $x \in \{0,1\}^\ell$. The constraint space is all the circuits $f : \{0,1\}^\ell \to \{0,1\}$ of bounded depth. The verification key vk consists of a uniformly sampled matrix $\vec{\mathbf{A}} = [\mathbf{A}_1 \| \dots \| \mathbf{A}_{\ell_\mathcal{X}}]$ and a close-to-uniform matrix $\mathbf{A}$, and the master signing key msk is a trapdoor for $\mathbf{A}$, i.e. $\mathbf{A}_{\tau_0}^{-1}$. A valid signature for an attribute x is a non-zero short-entries vector $\mathbf{v}_x$ such that $[\mathbf{A} \| \vec{\mathbf{A}} - x \otimes \mathbf{G}] \cdot \mathbf{v}_x = \mathbf{0}$, where $\mathbf{G}$ is a special fixed gadget matrix. The constrained signing key sk_f respective to a circuit f is a trapdoor $[\mathbf{A} \| \mathbf{A}_f]_\tau^{-1}$, where $\mathbf{A}_f$ is computed from $\vec{\mathbf{A}}$ and f. Given $\mathsf{msk} = \mathbf{A}_{\tau_0}^{-1}$ it is possible to generate a trapdoor $[\mathbf{A} \| \mathbf{M}]_\tau^{-1}$ for any matrix $\mathbf{M}$, so the authority can generate such keys efficiently. For any pair (x, f), a trapdoor $[\mathbf{A} \| \vec{\mathbf{A}} - x \otimes \mathbf{G}]_{\tau'}^{-1}$ can be derived from the trapdoor $[\mathbf{A} \| \mathbf{A}_f - f(x)\mathbf{G}]_\tau^{-1}$. This implies that when $f(x) = 0$, it can be derived from the signing key $\mathsf{sk}_f = [\mathbf{A} \| \mathbf{A}_f]_\tau^{-1}$. The trapdoor $[\mathbf{A} \| \vec{\mathbf{A}} - x \otimes \mathbf{G}]_{\tau'}^{-1}$ allows to sample a short vector $\mathbf{v}_x$ which is a valid signature for x. Since the signature is sampled from the same distribution regardless of the signing key, the scheme is statistically key-hiding. The proof of message-selective unforgeability is similar to the selective security proof in [BGG+14]. Recall that the adversary has to announce x for which it is going to forge a signature at the beginning of the game. The matrix $\vec{\mathbf{A}}$ is then generated from $\mathbf{A}$ based on x in such way that it is possible to generate a key for any function f for which $f(x) = 1$ without $\mathbf{A}_{\tau_0}^{-1}$. It is then shown that forging a signature for x implies breaking SIS respective to the matrix $\mathbf{A}$.

1.4 Related Work

Policy-based signatures were introduced in [BF14], where it was also shown that PBS for **NP** can be constructed from NIZK. [CNW16] construct lattice-based PBS in the random oracle model. [MPR11] introduced attribute-based signatures, and suggested a general framework for constructing

ABS from NIZK. In [SAH16] ABS for circuits is constructed from bilinear maps. [BK16] construct ABS for threshold functions and ($\vee$,$\wedge$)- functions from lattice assumptions. Our construction in Section 6 is the first ABS candidate for circuits that does not use NIZK or non-standard assumptions.

[Fuc14, CRV14] define *constrained verifiable random functions* (CVRF), which are constraint PRFs where given a constraint key one can compute, in addition to the function value, a non-interactive proof for the computed function value, where the proof is key-hiding. ABS can be constructed from CVRF trivially, however the pseudo-randomness property of known CVRF constructions implies *single-key* unforgeability of the derived ABS. [Fuc14, CRV14] show existence of CVRFs for poly-sized circuits, where the constructions assume mulitilinar-maps and the multilinear DDH assumption respectively.

Homomorphic signatures were constructed in [BF11, CFW14] for polynomials, and later in [GVW15] for boolean circuits. [LTWC16] define *multi-key* homomorphic signatures and show how to derive ABS from it. [FMNP16] define multi-key homomorphic MACS and signatures, and extend the [GVW15] HS construction to support multi-key evaluation.

Other notions of digital signatures with fine-grained control over signing permissions are *functional signatures* (FE) [BGI13] and *delegatable functional signatures* [BMS16]. In FE, a key respective to a function f can sign a message y if and only if the signer provides a preimage x such that $f(x) = y$. FE can be derived from CS for any function space consisting of efficiently invertible functions.

2 Preliminaries

2.1 Digital Signatures

Definition 2.1 ((Standard) Signature Scheme)**.** *A signature scheme is a tuple of PPT algorithms* $(\mathsf{Setup}, \mathsf{Sign}, \mathsf{Ver})$ *with the following syntax.*

- $\mathsf{Setup}(1^\lambda) \to (\mathsf{vk}, \mathsf{sk})$ *takes as input the security parameter* λ *and outputs a verification key* vk *and a signing key* sk*.*
- $\mathsf{Sign}(\mathsf{sk}, m) \to \sigma_m$ *takes as input a signing key* sk *and a message* m*, and outputs a signature* σ_m *for* m*.*
- $\mathsf{Ver}_{\mathsf{vk}}(m, \sigma_m)$ *takes as input a message* m *and a signature* σ_m*, and either accepts or rejects.*

Correctness. *The scheme is correct for a message space* $\mathcal{M}$*, if for all* $m \in \mathcal{M}$ *it holds that* $\mathsf{Ver}_{\mathsf{vk}}(m, \mathsf{Sign}(\mathsf{sk}, m)) = accept$*, where* $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{Setup}(1^\lambda)$*.*

Existential Unforgeability. *The scheme is existentially unforgeable for a message space* $\mathcal{M}$ *if every PPTM adversary* $\mathcal{A}$ *has no more than negligible advantage in the following game:*

1. *The challenger computes* $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{Setup}(1^\lambda)$ *and sends* vk *to* $\mathcal{A}$*.*
2. $\mathcal{A}$ *makes queries: it sends* $m \in \mathcal{M}$ *and gets in response* $\sigma_m \leftarrow \mathsf{Sign}(m, \mathsf{sk})$*.*
3. $\mathcal{A}$ *wins if it manages to output* (m^*, σ_{m^*}) *such that* $\mathsf{Ver}_{\mathsf{vk}}(m^*, \sigma_{m^*}) = accept$*, where* $m^* \neq m$ *for any signature queried by* $\mathcal{A}$ *for a message* $m \in \mathcal{M}$*.*

2.2 Short Integer Solution (SIS)

Below is the definition and hardness assumption of SIS, as phrased in [Pei16].

Definition 2.2 (Short Integer Solution ($\mathrm{SIS}_{n,q,B,m}$)). *Given a uniformly random matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, find a nonzero integer vector $\mathbf{r} \in \mathbb{Z}^m$ of norm $\|\mathbf{r}\|_\infty \leq B$ such that $\mathbf{A}\mathbf{r} = \mathbf{0}$.*

Theorem 2.1. *[Ajt96, Mic04, MR07, MP13] For any $m = \mathrm{poly}(n)$, $B > 0$, and sufficiently large $q \geq B \cdot \mathrm{poly}(n)$, solving $\mathrm{SIS}_{n,q,B,m}$ with non-negligible probability is at least as hard as solving the decisional approximate shortest vector problem GapSVP_γ and the approximate shortest independent vectors problem SIVP_γ on arbitrary n-dimensional lattices (i.e., in the worst case) with overwhelming probability, for some $\gamma = B \cdot \mathrm{poly}(n)$.*

2.3 Lattice Trapdoors

Let $n, q \in \mathbb{Z}$, $\mathbf{g} = (1, 2, 4, \ldots, 2^{\lceil \log q \rceil - 1}) \in \mathbb{Z}_q^{\lceil \log q \rceil}$ and $m = n\lceil \log q \rceil$. The *gadget matrix* $\mathbf{G}$ is defined as the diagonal concatenation of $\mathbf{g}$ n times. Formally, $\mathbf{G} = \mathbf{g} \otimes \mathbf{I}_n \in \mathbb{Z}_q^{n \times m}$. For any $t \in \mathbb{Z}$, the function $\mathbf{G}^{-1} : \mathbb{Z}_q^{n \times t} \to \{0,1\}^{m \times t}$ expands each entry $a \in \mathbb{Z}_q$ of the input matrix into a column of size $\lceil \log q \rceil$ consisting of the bits representation of a. For any matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times t}$, it holds that $\mathbf{G} \cdot \mathbf{G}^{-1}(\mathbf{A}) = \mathbf{A}$.

The (centered) discrete Gaussian distribution over $\mathbb{Z}^m$ with parameter τ, denoted $D_{\mathbb{Z}^m,\tau}$, is the distribution over $\mathbb{Z}^m$ where for all $\mathbf{x}$, $\Pr[\mathbf{x}] \propto e^{-\pi \|\mathbf{x}\|^2/\tau^2}$. Let $n, m, q \in \mathbb{N}$ and consider a matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$. For all $\mathbf{v} \in \mathbb{Z}_q^n$ we let $\mathbf{A}_\tau^{-1}(\mathbf{v})$ denote the random variable whose distribution is the Discrete Gaussian $D_{\mathbb{Z}^m,\tau}$ conditioned on $\mathbf{A} \cdot \mathbf{A}_\tau^{-1}(\mathbf{v}) = \mathbf{v}$.

A τ-trapdoor for $\mathbf{A}$ is a procedure that can sample from a distribution within 2^{-n} statistical distance of $\mathbf{A}_\tau^{-1}(\mathbf{v})$ in time $\mathrm{poly}(n, m, \log q)$, for any $\mathbf{v} \in \mathbb{Z}_q^n$. We slightly overload notation and denote a τ-trapdoor for $\mathbf{A}$ by $\mathbf{A}_\tau^{-1}$. The following properties had been established in a long sequence of works.

Corollary 2.2 (Trapdoor Generation [Ajt96, MP12]). *There exists an efficiently computable value $m_0 = O(n \log q)$ and an efficient procedure $\mathsf{TrapGen}(1^n, q, m)$ such that for all $m \geq m_0$ outputs $(\mathbf{A}, \mathbf{A}_{\tau_0}^{-1})$, where $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ is 2^{-n}-uniform and $\tau_0 = O(\sqrt{n \log q \log n})$.*

We use the most general form of trapdoor extension as formalized in [MP12].

Theorem 2.3 (Trapdoor Extension [ABB10, MP12]). *Given $\bar{\mathbf{A}} \in \mathbb{Z}_q^{n \times m}$ with a trapdoor $\bar{\mathbf{A}}_\tau^{-1}$, and letting $\bar{\mathbf{B}} \in \mathbb{Z}_q^{n \times m'}$ be s.t. $\bar{\mathbf{A}} = \bar{\mathbf{B}}\mathbf{S} \pmod q$ where $\mathbf{S} \in \mathbb{Z}^{m' \times m}$ with largest singular value $s_1(\mathbf{S})$, then $(\bar{\mathbf{A}}_\tau^{-1}, \mathbf{S})$ can be used to sample from $\bar{\mathbf{B}}_{\tau'}^{-1}$ for any $\tau' \geq \tau \cdot s_1(\mathbf{S})$.*

A few additional important corollaries are derived from this theorem. We recall that $s_1(\mathbf{S}) \leq \sqrt{m'm}\,\|\mathbf{S}\|_\infty$ and that a trapdoor $\mathbf{G}_{O(1)}^{-1}$ is trivial. The first is a trapdoor extension that follows by taking $\mathbf{S} = [\mathbf{I}_{m'} \| \mathbf{0}_m]^T$.

Corollary 2.4. *Given $\mathbf{A} \in \mathbb{Z}_q^{n \times m'}$, with a trapdoor $\mathbf{A}_\tau^{-1}$, it is efficient to generate a trapdoor $[\mathbf{A}\|\mathbf{B}]_{\tau'}^{-1}$ for all $\mathbf{B} \in \mathbb{Z}_q^{n \times m}$, for any $m \in \mathbb{N}$ and any $\tau' \geq \tau$.*

Next is a trapdoor extension that had been used extensively in prior work. It follows from Theorem 2.3 with $\mathbf{S} = [-\mathbf{R}^T \| \mathbf{I}_m]^T$.

Corollary 2.5. *Given* $\mathbf{A} \in \mathbb{Z}_q^{n \times m'}$, *and* $\mathbf{R} \in \mathbb{Z}^{m' \times m}$ *with* $m = n\lceil \log q \rceil$, *it is efficient to compute* $[\mathbf{A} \| \mathbf{AR} + \mathbf{G}]_\tau^{-1}$ *for* $\tau = O(\sqrt{mm'}\, \|\mathbf{R}\|_\infty)$.

Note that by taking $\mathbf{A}$ uniform and $\mathbf{R}$ to be a high entropy small matrix, e.g. uniform in $\{-1, 0, 1\}$, and relying on the leftover hash lemma, Corollary 2.2 is in fact a special case of this one.

It is also possible to permute trapdoors in the following manner.

Corollary 2.6. *Given* $[\mathbf{A}_1 \| \dots \| \mathbf{A}_t]_\tau^{-1}$ *and a permutation* $\rho : \mathbb{Z}_t \to \mathbb{Z}_t$, *it is efficient to compute* $[\mathbf{A}_{\rho(1)} \| \dots \| \mathbf{A}_{\rho(t)}]_\tau^{-1}$.

2.4 Lattice Evaluation

The following is an abstraction of the evaluation procedure in recent LWE based FHE and ABE schemes, that developed in a long sequence of works [ABB10, MP12, GSW13, AP14, BGG+14, GVW15]. We use a similar formalism to [BV15, BCTW16] but slightly rename the functions.

Theorem 2.7. *There exist efficient deterministic algorithms* EvalF *and* EvalFX *such that for all* $n, q, \ell \in \mathbb{N}$, *and for any sequence of matrices* $\vec{\mathbf{A}} = (\mathbf{A}_1, \dots, \mathbf{A}_\ell) \in (\mathbb{Z}_q^{n \times n\lceil \log q \rceil})^\ell$, *for any depth* d *boolean circuit* $f : \{0,1\}^\ell \to \{0,1\}$ *and for every* $\mathbf{x} = (x_1, \dots, x_\ell) \in \{0,1\}^\ell$, *the outputs* $\mathbf{H}_f = \mathsf{EvalF}(f, \vec{\mathbf{A}})$ *and* $\mathbf{H}_{f,x} = \mathsf{EvalFX}(f, x, \vec{\mathbf{A}})$ *are both in* $\mathbb{Z}^{(\ell n\lceil \log q \rceil) \times n\lceil \log q \rceil}$ *and it holds that* $\|\mathbf{H}_f\|_\infty, \|\mathbf{H}_{f,x}\|_\infty \le (2n\lceil \log q \rceil)^d$ *and* $(\vec{\mathbf{A}} - \mathbf{x} \otimes \mathbf{G}) \cdot \mathbf{H}_{f,\mathbf{x}} = \vec{\mathbf{A}} \cdot \mathbf{H}_f - f(\mathbf{x})\mathbf{G} \pmod q$.

3 Definition of Constrained Signatures (CS)

We now define constrained signatures, along with a number of security notions that will be used throughout this work. The definitions are presented in the key-policy flavor. See Appendix A for definitions in the message-policy flavor. Lastly we define key delegation in the context of constrained signatures.

Definition 3.1 ((Key-Policy) Constrained Signatures)**.** *Let* $\mathcal{X}$ *be an attribute space and* $\mathcal{F}$ *be a function space of the form* $f \in \mathcal{F} \implies f : \mathcal{X}' \to \{0,1\}$ *where* $\mathcal{X}' \subseteq \mathcal{X}$. *A constrained signatures scheme for* $(\mathcal{X}, \mathcal{F})$ *is a tuple of algorithms:*

- $\mathsf{Setup}(1^\lambda) \to (\mathsf{msk}, \mathsf{vk})$ *takes as input the security parameter* λ *and possibly a description of* $(\mathcal{X}, \mathcal{F})$, *and outputs a master signing key* msk *and a public verification key* vk.
- $\mathsf{Keygen}(f, \mathsf{msk}) \to \mathsf{sk}_f$ *takes as input a function* $f \in \mathcal{F}$ *and the master signing key* msk, *and outputs a signing key* sk_f.
- $\mathsf{Sign}(x, \mathsf{sk}_f) \to \sigma_x$ *takes as input an attribute* $x \in \mathcal{X}$ *and a signing key* sk_f, *and outputs a signature* σ_x.
- $\mathsf{Ver}_{\mathsf{vk}}(x, \sigma_x) \to \{accept, reject\}$ *takes as input an attribute* $x \in \mathcal{X}$ *and a signature* σ_x, *and either accepts or rejects.*

Correctness. *The scheme is correct if for all* $x \in \mathcal{X}$ *and* $f \in \mathcal{F}$ *for which* $f(x) = 0$, *it holds that with all but negligible probability* $\mathsf{Ver}_{\mathsf{vk}}(x, \sigma_x) = accept$, *where* $(\mathsf{msk}, \mathsf{vk}) \leftarrow \mathsf{Setup}(1^\lambda)$ *and* $\sigma_x = \mathsf{Sign}(x, \mathsf{Keygen}(f, \mathsf{msk}))$.

Privacy. Privacy bounds the information revealed by a signature about the signing key that was used to produce it. We define two notions of privacy. In *constraint-hiding* privacy, a signature should not reveal the signing key's functionality f, however it might be possible to retrieve other information such as whether two signatures were produced using the same key. In *key-hiding* privacy, a signature should not reveal any information at all about the signing key.

Definition 3.2 (Privacy of (Key-Policy) Constrained Signatures)**.** *The scheme is* constraint-hiding *if any* PPT *adversary $\mathcal{A}$ has no more than negligible advantage in the following game.*

1. *The challenger computes and outputs* $(\mathsf{msk}, \mathsf{vk}) \leftarrow \mathsf{Setup}(1^\lambda)$.
2. *$\mathcal{A}$ sends* (f_0, f_1, x) *such that* $f_0(x) = f_1(x) = 0$.
3. *The challenger computes* $\mathsf{sk}_{f_0} = \mathsf{Keygen}(f_0, \mathsf{msk})$ *and* $\mathsf{sk}_{f_1} = \mathsf{Keygen}(f_1, \mathsf{msk})$. *It then samples* $b \xleftarrow{\$} \{0,1\}$ *and computes* $\sigma_{x,b} \leftarrow \mathsf{Sign}(x, \mathsf{sk}_{f_b})$. *It sends $\sigma_{x,b}$ to $\mathcal{A}$.*
4. *$\mathcal{A}$ outputs* $b' \in \{0,1\}$ *and wins if and only if* $b' = b$.

The scheme is key-hiding *if any* PPT *adversary $\mathcal{A}$ has no more than negligible advantage in the above game, where in step 3 the challenger sends* $(\mathsf{sk}_{f_0}, \mathsf{sk}_{f_1}, \sigma_{x,b})$ *to $\mathcal{A}$.*

Unforgeability. We consider *full unforgeability* vs. *message-selective unforgeability*. These notions are caputred by a security game between a challenger and an adversary. In the full unforgeability game, the adversary can adaptively make queries of three types: (1) query for constrained keys, (2) query for signatures under a specified constraint, and (3) query for signatures that are generated with an existing key from a type (2) query. In order to win the adversary has to forge a signature for an attribute x^* that is not authorized by any of the queried keys, and does not appear in any of the type (2) and (3) signature queries. In the message-selective game, the adversary has to announce x^* before seeing the verification key. The construction in Section 6 is message-selective unforgeable.

Definition 3.3 (Unforgeability of (Key-Policy) Constrained Signatures)**.** *The scheme is* fully unforgeable *if every PPTM adversary $\mathcal{A}$ has no more than negligible advantage in the following game:*

1. *The challenger computes* $(\mathsf{msk}, \mathsf{vk}) \leftarrow \mathsf{Setup}(1^\lambda)$ *and sends* vk *to $\mathcal{A}$.*
2. *$\mathcal{A}$ makes queries of three types:*
   - Key Queries. *$\mathcal{A}$ sends* $f \in \mathcal{F}$ *and gets back* $\mathsf{sk}_f \leftarrow \mathsf{Keygen}(f, \mathsf{msk})$.
   - Signature Queries. *$\mathcal{A}$ sends* $(f, x) \in \mathcal{F} \times \mathcal{X}$ *such that* $f(x) = 0$. *The challenger computes* $\mathsf{sk}_f \leftarrow \mathsf{Keygen}(f, \mathsf{msk})$ *and sends back* $\sigma_x \leftarrow \mathsf{Sign}(x, \mathsf{sk}_f)$.
   - Repeated Signature Queries. *$\mathcal{A}$ sends* $i \in \mathbb{N}$ *and* $x \in \mathcal{X}$. *If there were less than i signature queries at this point of the game, the challenger returns $\perp$. Otherwise, let f denote the constraint that was sent at the ith signature query and let sk_f denote the key that was generated by the challenger when answering this query. If* $f(x) \neq 0$, *the challenger returns $\perp$. Otherwise it returns* $\sigma_x \leftarrow \mathsf{Sign}(x, \mathsf{sk}_f)$.
3. *$\mathcal{A}$ wins if it manages to output* (x^*, σ_{x^*}) *such that* $\mathsf{Ver}_{\mathsf{vk}}(x^*, \sigma_{x^*}) =$ *accept and the following restrictions hold:*

- *For any key queried by $\mathcal{A}$ respective to $f \in \mathcal{F}$, it holds that $f(x^*) = 1$.*
- *For any signature σ_x queried by $\mathcal{A}$, it holds that $x \neq x^*$.*

The scheme is message-selective *unforgeable if any PPT $\mathcal{A}$ that announces x^* before seeing* vk *has no more than negligible advantage in the game.*

We also define a relaxed notion, *single-key-selective unforgeability*, which is useful as a building block towards full unforgeability, as shown in Section 4. In this security game, the adversary is restricted to a single key query and no signatures queries. It also has to announce the queried constraint at the beginning of the game.

Definition 3.4 (Single-Key-Selective Unforgeability of (Key-Policy) Constrained Signatures)**.** *The scheme is* single-key-selective *unforgeable if every PPTM adversary $\mathcal{A}$ has no more than negligible advantage in the following game:*

1. *$\mathcal{A}$ sends $f^* \in \mathcal{F}$ to the challenger.*
2. *The challenger computes $(\mathsf{msk}, \mathsf{vk}) \leftarrow \mathsf{Setup}(1^\lambda)$ and $\mathsf{sk}_{f^*} \leftarrow \mathsf{Keygen}(f^*, \mathsf{msk})$, and sends $(\mathsf{vk}, \mathsf{sk}_{f^*})$ to $\mathcal{A}$.*
3. *$\mathcal{A}$ wins if it manages to output $(x^*, \sigma_{(x^*)})$ such that $\mathsf{Ver}_{\mathsf{vk}}(x^*, \sigma_{x^*}) = accept$ and $f^*(x^*) = 1$.*

3.1 Key Delegation

Given a key sk_f for a constraint $f \in \mathcal{F}$, it might be useful to generate a key with limited capabilities, i.e. a key $\mathsf{sk}_{(f,g)}$ for a constraint that requires $f(x) = 0$ *and* $g(x) = 0$ for some function $g \in \mathcal{F}$. In this setting, any attribute $x \in \mathcal{X}$ that can be signed by $\mathsf{sk}_{(f,g)}$ can also be signed by sk_f, but the other direction is not guaranteed since it might be the case that $f(x) = 0$ but $g(x) = 1$. Key delegation can therefore be though of as restricting the signing permissions of a given key.

We now give a formal definition of the key delegation algorithm, along with definitions for correctness, privacy and unforgeability. Note that it captures multiple levels of delegation. The unforgeability game is analogouse to the non-delegatable unforgeability game, where the adversary can in addition query for delegated keys.

Definition 3.5 (Delegation of (Key-Policy) Constrained Signatures)**.** *A constrained signatures scheme* $\mathsf{CS} = (\mathsf{Setup}, \mathsf{Keygen}, \mathsf{Sign}, \mathsf{Ver})$ *with attribute space $\mathcal{X}$, function space $\mathcal{F}$ and key space $\mathcal{K}$ supports delegation if there exists a PPT algorithm* DelKey *with the syntax*

- $\mathsf{DelKey}(\mathsf{sk}_{(f_1,\ldots,f_t)}, f_{t+1}) \to \mathsf{sk}_{(f_1,\ldots,f_{t+1})}$*: takes as input a constrained key $\mathsf{sk}_{(f_1,\ldots,f_t)} \in \mathcal{K}$ and a function $f_{t+1} \in \mathcal{F}$, and outputs a delegated constrained key $\mathsf{sk}_{(f_1,\ldots,f_{t+1})} \in \mathcal{K}$.*

such that it satisfies correctness, privacy and unforgeability as defined below. For any $t \geq 1$ and $F = (f_1, \ldots, f_t) \in \mathcal{F}^t$, write $F(x) = 0$ to denote that $f \in F \Rightarrow f(x) = 0$. Moreover, denote $\mathsf{sk}_F = \mathsf{sk}_{(f_1,\ldots,f_t)}$, where $\forall i \in [2 \ldots t] : \mathsf{sk}_{(f_1,\ldots,f_i)} = \mathsf{DelKey}(\mathsf{sk}_{(f_1,\ldots,f_{i-1})}, f_i)$ and $\mathsf{sk}_{f_1} = \mathsf{Keygen}(f_1, \mathsf{msk})$ for some $(\mathsf{msk}, \mathsf{vk}) \leftarrow \mathsf{Setup}(1^\lambda)$ which is clear from the context.

Correctness. *Consider $(\mathsf{msk}, \mathsf{vk}) \leftarrow \mathsf{Setup}(1^\lambda)$. The scheme is correct for a function family $\mathcal{F}$ and attribute space $\mathcal{X}$, if for all $t \in \mathbb{N}$, $(x, F) \in \mathcal{X} \times \mathcal{F}^t$ for which $F(x) = 0$, it holds with all but negligible probability that $\mathsf{Ver}_{\mathsf{vk}}(x, \mathsf{Sign}(x, \mathsf{sk}_F)) = accept$.*

Privacy. *The scheme is* constraint-hiding *if any* PPT *adversary $\mathcal{A}$ has no more than negligible advantage in the following game.*

1. *The challenger computes and outputs* $(\mathsf{msk}, \mathsf{vk}) \leftarrow \mathsf{Setup}(1^\lambda)$.
2. *$\mathcal{A}$ sends* (t, F_0, F_1, x), *where* $\forall b \in \{0,1\} : F_b = (f_1^b, \ldots, f_t^b)$ *and* $F_b(x) = 0$.
3. *The challenger computes* sk_{F_0} *and* sk_{F_1}. *It then samples* $b \xleftarrow{\$} \{0,1\}$ *and computes* $\sigma_{x,b} \leftarrow \mathsf{Sign}(x, \mathsf{sk}_{F_b})$. *It sends* $\sigma_{x,b}$ *to* $\mathcal{A}$.
4. *$\mathcal{A}$ outputs* $b' \in \{0,1\}$ *and wins if and only if* $b' = b$.

The scheme is key-hiding *if any* PPT *adversary $\mathcal{A}$ has no more than negligible advantage in the above game, where in step 3 the challenger sends* $(\mathsf{sk}_{F_0}, \mathsf{sk}_{F_1}, \sigma_{x,b})$ *to* $\mathcal{A}$.

Full Unforgeability. *The scheme is* fully unforgeable *if every PPTM adversary $\mathcal{A}$ has no more than negligible advantage in the following game:*

1. *The challenger computes* $(\mathsf{msk}, \mathsf{vk}) \leftarrow \mathsf{Setup}(1^\lambda)$ *and sends* vk *to* $\mathcal{A}$.
2. *$\mathcal{A}$ makes queries of three types:*
   - Key Queries. *$\mathcal{A}$ sends* $t \in \mathbb{N}$, $F \in \mathcal{F}^t$ *and gets back* sk_F.
   - Signature Queries. *$\mathcal{A}$ sends* $t \in \mathbb{N}$, $(F, x) \in \mathcal{F}^t \times \mathcal{X}$ *such that* $F(x) = 0$. *The challenger computes* sk_F *as described above and returns* $\sigma_x \leftarrow \mathsf{Sign}(x, \mathsf{sk}_F)$.
   - Repeated Signature Queries. *$\mathcal{A}$ sends* $i \in \mathbb{N}$ *and* $x \in \mathcal{X}$. *If there were less than i signature queries at this point of the game, the challenger returns* $\perp$. *Otherwise, let F denote the set of constraints that was sent at the ith signature query and let* sk_F *denote the key that was generated by the challenger when answering this query. If* $\exists f \in F$ *s.t.* $f(x) \neq 0$, *the challenger returns* $\perp$. *Otherwise it returns* $\sigma_x \leftarrow \mathsf{Sign}(x, \mathsf{sk}_f)$.
3. *$\mathcal{A}$ wins if it manages to output* (x^*, σ_{x^*}) *such that* $\mathsf{Ver}_{\mathsf{vk}}(x^*, \sigma_{x^*}) =$ *accept and the following restrictions hold:*
   - *For any key queried by $\mathcal{A}$ respective to* $t \in \mathbb{N}$, $F \in \mathcal{F}^t$, *it holds that* $\exists f \in F$ *such that* $f(x^*) = 1$.
   - *For any signature* σ_x *queried by $\mathcal{A}$, it holds that* $x \neq x^*$.

Message-Selective Unforgeability. *The scheme maintains* message-selective *unforgeability if any PPT $\mathcal{A}$ that announces* x^* *before seeing* vk *has no more than negligible advantage in the game.*

Single-Key-Selective Unforgeability. *The scheme is* single-key-selective *unforgeable if every PPTM adversary $\mathcal{A}$ has no more than negligible advantage in the following game:*

1. *$\mathcal{A}$ sends* $t \in \mathbb{N}, F \in \mathcal{F}^t$ *to the challenger.*
2. *The challenger computes* $(\mathsf{msk}, \mathsf{vk}) \leftarrow \mathsf{Setup}(1^\lambda)$ *and* sk_F, *and sends* $(\mathsf{vk}, \mathsf{sk}_F)$ *to* $\mathcal{A}$.
3. *$\mathcal{A}$ wins if it manages to output* (x^*, σ_{x^*}) *such that* $\mathsf{Ver}_{\mathsf{vk}}(x^*, \sigma_{x^*}) =$ *accept and* $\exists f \in F$ *such that* $f(x^*) = 1$.

4 From Single-Key-Selective Unforgeability to Full Unforgeability

We show how any standard digital signatures scheme can be used to amplify the security guarantee of a (key-policy) CS scheme from *single-key-selective* to *full* unforgeability. This comes with a partial loss in key privacy – while the underlying scheme might be either key-hiding or solely constraint-hiding, the amplified scheme reveals key-specific information as part of the signature, and thus it is only constraint-hiding.

Let $\mathsf{CS} = (\mathsf{Setup}', \mathsf{Keygen}', \mathsf{Sign}', \mathsf{Ver}')$ be a single-key selectively unforgeable constraint-hiding constrained signature scheme with attribute space $\mathcal{X}'$, constraint space $\mathcal{F}'$ and verification-key space $\mathcal{VK}'$. Let $\mathsf{S} = (\mathsf{S.Setup}, \mathsf{S.Sign}, \mathsf{S.Ver})$ be a standard signature scheme with verification-key space $\mathcal{VK}$ and message space $\mathcal{X}$ such that $\mathcal{VK}' \times \mathcal{VK} \subseteq \mathcal{X}$. The construction is as follows.

- $\mathsf{Setup}(1^\lambda)$. Compute $(\mathsf{S.vk}, \mathsf{S.sk}) \leftarrow \mathsf{S.Setup}(1^\lambda)$. Output $\mathsf{vk} = \mathsf{S.vk}$ and $\mathsf{msk} = \mathsf{S.sk}$.
- $\mathsf{Keygen}(f, \mathsf{msk})$. Generate $(\mathsf{vk}', \mathsf{msk}') \leftarrow \mathsf{Setup}'(1^\lambda)$. Compute $k'_f \leftarrow \mathsf{Keygen}'(\mathsf{msk}', f)$. Generate $(\mathsf{vk}'', \mathsf{sk}'') \leftarrow \mathsf{S.Setup}(1^\lambda)$. Sign $(\mathsf{vk}', \mathsf{vk}'')$ using msk: $\sigma_{(\mathsf{vk}', \mathsf{vk}'')} \leftarrow \mathsf{S.Sign}(\mathsf{S.sk}, (\mathsf{vk}', \mathsf{vk}''))$. Output $k_f = (\mathsf{vk}', k'_f, \mathsf{vk}'', \mathsf{sk}'', \sigma_{(\mathsf{vk}', \mathsf{vk}'')})$.
- $\mathsf{Sign}(x, k_f)$. Compute $\sigma'_x = \mathsf{Sign}'(x, k'_f)$ and $\sigma''_x = \mathsf{S.Sign}(\mathsf{sk}'', x)$. Output $\sigma_x = (\mathsf{vk}', \sigma'_x, \mathsf{vk}'', \sigma''_x, \sigma_{(\mathsf{vk}', \mathsf{vk}'')})$.
- $\mathsf{Ver}_{\mathsf{vk}}(x, \sigma_x)$. Accept only if $\mathsf{S.Ver}(\sigma_{(\mathsf{vk}', \mathsf{vk}'')}, (\mathsf{vk}', \mathsf{vk}'')) = \mathit{accept}$, $\mathsf{Ver}'_{\mathsf{vk}'}(x, \sigma'_x) = \mathit{accept}$ and $\mathsf{S.Ver}_{\mathsf{vk}''}(x, \sigma''_x) = \mathit{accept}$.

Lemma 4.1 (Correctness)**.** *The scheme is correct for* $(\mathcal{F}', \mathcal{X}')$.

Proof. Fix $x \in \mathcal{X}'$ and $f \in \mathcal{F}'$ such that $f(x) = 0$, and consider $(\mathsf{msk}, \mathsf{vk}) \leftarrow \mathsf{Setup}(1^\lambda)$ and $\sigma_x = \mathsf{Sign}(x, \mathsf{Keygen}(f, \mathsf{msk}))$. Denote $\sigma_x = (\mathsf{vk}', \sigma'_x, \mathsf{vk}'', \sigma''_x, \sigma_{(\mathsf{vk}', \mathsf{vk}'')})$, then by Sign and Keygen it holds that $\sigma'_x = \mathsf{Sign}'(x, k'_f) = \mathsf{Sign}'(x, \mathsf{Keygen}'(\mathsf{msk}', f))$, and since $f(x) = 0$ it holds that $\mathsf{Ver}'_{\mathsf{vk}'}(\sigma'_x, x) = \mathit{accept}$ by the correctness of CS'. Moreover, $\mathsf{S.Ver}_{\mathsf{vk}''}(x, \sigma''_x) = \mathsf{S.Ver}_{\mathsf{vk}''}(x, \mathsf{S.Sign}(\mathsf{sk}'', x) = \mathit{accept}$ and $\mathsf{S.Ver}(\sigma_{(\mathsf{vk}', \mathsf{vk}'')}, (\mathsf{vk}', \mathsf{vk}'')) = \mathsf{S.Ver}(\mathsf{S.Sign}(\mathsf{S.sk}, (\mathsf{vk}', \mathsf{vk}'')), (\mathsf{vk}', \mathsf{vk}'')) = \mathit{accept}$ by the correctness of S. Therefore, $\mathsf{Ver}_{\mathsf{vk}}(x, m, \sigma_x)$ accepts. □

Lemma 4.2 (Privacy)**.** *The scheme is constraint-hiding for* $(\mathcal{F}', \mathcal{X}')$.

Proof. Assume towards contradiction an adversary $\mathcal{A}$ that wins the constraint-hiding privacy game with some significant probability, and use it to break the constraint-hiding privacy of CS as follows:

1. Receive $(\mathsf{vk}', \mathsf{msk}') \leftarrow \mathsf{Setup}'(1^\lambda)$ from the CS challenger.
2. Compute $(\mathsf{S.vk}, \mathsf{S.sk}) \leftarrow \mathsf{S.Setup}(1^\lambda)$ and send $(\mathsf{msk} = \mathsf{S.sk}, \mathsf{vk} = \mathsf{S.vk})$ to $\mathcal{A}$.
3. $\mathcal{A}$ returns (f_0, f_1, x) such that $f_0(x) = f_1(x) = 0$. Forward (f_0, f_1, x) to the CS challenger.
4. The CS challenger samples $b \xleftarrow{\$} \{0, 1\}$ and returns $\sigma'_{x,b}$. Now generate $(\mathsf{vk}'', \mathsf{sk}'') \leftarrow \mathsf{S.Setup}(1^\lambda)$, sign $(\mathsf{vk}', \mathsf{vk}'')$ with the standard signature scheme: $\sigma_{(\mathsf{vk}', \mathsf{vk}'')} \leftarrow \mathsf{S.Sign}(\mathsf{S.sk}, (\mathsf{vk}', \mathsf{vk}''))$, sign x with the standard signature scheme: $\sigma''_x \leftarrow \mathsf{S.Sign}(\mathsf{sk}'', x)$ and send to $\mathcal{A}$ the signature $\sigma_{x,b} = (\mathsf{vk}', \sigma'_{x,b}, \mathsf{vk}'', \sigma''_x, \sigma_{(\mathsf{vk}', \mathsf{vk}'')})$.
5. Get b' from $\mathcal{A}$ and forward it to the CS challenger. Clearly, any advantage of $\mathcal{A}$ induces an advantage of the reduction.

□

Lemma 4.3 (Unforgeability)**.** *The scheme is fully unforgeable for* $(\mathcal{F}', \mathcal{X}')$*.*

Proof. Assume towards contradiction an adversary $\mathcal{A}$ that wins the security game. We show that it can be used to break either S or CS. Let $\mathcal{Q}_{key}, \mathcal{Q}_{sig}, \mathcal{Q}_{rep}$ be the sets of key queries, signature queries and repeated signature queries made by $\mathcal{A}$ during the security game. Recall that $\mathcal{Q}_{key} \in \mathcal{F}'$, $\mathcal{Q}_{sig} \in \mathcal{F}' \times \mathcal{X}'$ and $\mathcal{Q}_{rep} \in \mathbb{N} \times \mathcal{X}'$. In particular, each query $q_i \in \mathcal{Q}_{key} \bigcup \mathcal{Q}_{sig}$ contains an element $f_i \in \mathcal{F}$. Moreover, every response of the challenger (whether it is a key or a signature) contains a pair $(\mathsf{vk}'_i, \mathsf{vk}''_i)$ that is generated during $\mathsf{Keygen}(f_i, \mathsf{msk})$. $\mathcal{A}$ wins the game, it therefore outputs a successful forgery (x^*, σ_{x^*}), where $\sigma_{x^*} = (\mathsf{vk}'_*, \sigma'_{x^*}, \mathsf{vk}''_*, \sigma''_{x^*}, \sigma_{(\mathsf{vk}'_*, \mathsf{vk}''_*)})$. Since $\mathsf{Ver}_{\mathsf{vk}}(x^*, \sigma_{(x^*,m^*)}) = \mathit{accept}$, it holds that $\mathsf{S.Ver}(\sigma_{\mathsf{vk}'_*}, \mathsf{vk}'_*)$ accepts, $\mathsf{Ver}'_{\mathsf{vk}'_*}(x^*, \sigma'_{x^*})$ accepts and $\mathsf{S.Ver}_{\mathsf{vk}''}(x^*, \sigma''_{x^*})$ accepts. Consider three cases:

- If $\exists q_i \in \mathcal{Q}_{key}$ such that $(\mathsf{vk}'_i, \mathsf{vk}''_i) = (\mathsf{vk}'_*, \mathsf{vk}''_*)$, then (x^*, σ'_{x^*}) is a valid forgery to the CS instance that was initialized during $\mathsf{Keygen}(f_i, \mathsf{msk})$. Note that since $q_i \in \mathcal{Q}_{key}$, $f_i(x^*) = 1$. We show a reduction from the selective-single-key security game of CS to this game:

  1. Initialize $(\mathsf{S.vk}, \mathsf{S.sk}) \leftarrow \mathsf{S.Setup}(1^\lambda)$ as in the real scheme and send $\mathsf{S.vk}$ to $\mathcal{A}$.
  2. Queries phase:
     - Answer all queries except of the ith as in the real unforgeability game.
     - Upon receiving form $\mathcal{A}$ the query $q_i \in \mathcal{Q}_{key}$, send f_i to the ith CS challenger and get back $(\mathsf{vk}'_i, k'_{f_i})$. Generate $(\mathsf{vk}''_i, \mathsf{sk}''_i) \leftarrow \mathsf{S.Setup}(1^\lambda)$, sign $(\mathsf{vk}'_i, \mathsf{vk}''_i)$ with the standard scheme: $\sigma_{(\mathsf{vk}'_i, \mathsf{vk}''_i)} \leftarrow \mathsf{S.Sign}(\mathsf{S.sk}, (\mathsf{vk}'_i, \mathsf{vk}''_i))$. Send to $\mathcal{A}$ the key $k_{f_i} = (\mathsf{vk}'_i, k'_{f_i}, \mathsf{vk}''_i, \mathsf{sk}''_i, \sigma_{(\mathsf{vk}'_i, \mathsf{vk}''_i)})$.
  3. When $\mathcal{A}$ sends the forgery (x^*, σ_{x^*}), send (x^*, σ'_{x^*}) to the ith CS challenger to win the selective-single-key game.

- If $\exists q_i \in \mathcal{Q}_{sig}$ such that $(\mathsf{vk}'_i, \mathsf{vk}''_i) = (\mathsf{vk}'_*, \mathsf{vk}''_*)$, then (x^*, σ''_{x^*}) is a valid forgery to the S instance that was initialized during $\mathsf{Keygen}(f_i, \mathsf{msk})$. Note that $\forall q_i \in \mathcal{Q}_{sig}$, where $q_i = (f_i, x_i)$, it holds that $x_i \neq x^*$. We show a reduction from the security game of $\mathcal{S}$ to this game:

  1. Initialize $(\mathsf{S.vk}, \mathsf{S.sk}) \leftarrow \mathsf{S.Setup}(1^\lambda)$ as in the real scheme and send $\mathsf{S.vk}$ to $\mathcal{A}$.
  2. Queries phase:
     - Answer all queries up to q_i as in the real unforgeability game.
     - Upon receiving form $\mathcal{A}$ the query $q_i \in \mathcal{Q}_{sig}$, instantiate a game against the ith S challenger and get vk''_i. Query a signature for x_i and get back σ''_{x_i}. Generate $(\mathsf{vk}'_i, \mathsf{msk}'_i) \leftarrow \mathsf{Setup}'(1^\lambda)$ and $k'_{f_i} \leftarrow \mathsf{Keygen}'(\mathsf{msk}'_i, f_i)$, sign $\sigma'_{x_i} \leftarrow \mathsf{Sign}'(x_i, k'_{f_i})$. Sign $(\mathsf{vk}'_i, \mathsf{vk}''_i)$ with the standard signature scheme: $\sigma_{(\mathsf{vk}'_i, \mathsf{vk}''_i)} \leftarrow \mathsf{S.Sign}(\mathsf{S.sk}, (\mathsf{vk}'_i, \mathsf{vk}''_i))$. Send $\mathcal{A}$ the signature $\sigma_{x_i} = (\mathsf{vk}'_i, \sigma'_{x_i}, \mathsf{vk}''_i, \sigma''_{x_i}, \sigma_{(\mathsf{vk}'_i, \mathsf{vk}''_i)})$.
     - Answer all queries as in the real game, except of repeated signature queries that reference q_i. For these, do as described above with the same vk'_i, $\mathsf{vk}''_i, \sigma_{(\mathsf{vk}'_i, \mathsf{vk}''_i)}, k'_{f_i}$.
  3. When $\mathcal{A}$ sends the forgery (x^*, σ_{x^*}), send (x^*, σ''_{x^*}) to the ith S challenger to win the game.

- Otherwise $\forall q_i \in \mathcal{Q}_{key} \bigcup \mathcal{Q}_{sig}\ (\mathsf{vk}'_i, \mathsf{vk}''_i) \neq (\mathsf{vk}'_*, \mathsf{vk}''_*)$, and thus $(\sigma_{(\mathsf{vk}'_*, \mathsf{vk}''_*)}, (\mathsf{vk}'_*, \mathsf{vk}''_*))$ is a valid forgery to S. We show a reduction from the security game of S to this game:

  1. Receive $\mathsf{S.vk}$ from the S challenger and send it to $\mathcal{A}$.
  2. Answer queries from $\mathcal{A}$ as in the real game, except the way $\sigma_{(\mathsf{vk}'_i, \mathsf{vk}''_i)}$ is computed: instead of signing $(\mathsf{vk}'_i, \mathsf{vk}''_i)$ with $\mathsf{msk} = \mathsf{S.sk}$ (which we don't have), query the S challenger and get $\sigma_{(\mathsf{vk}'_i, \mathsf{vk}''_i)}$.
  3. When $\mathcal{A}$ sends the forgery (x^*, σ_{x^*}), send $(\sigma_{(\mathsf{vk}'_*, \mathsf{vk}''_*)}, (\mathsf{vk}'_*, \mathsf{vk}''_*))$ to the S challenger to win the game.

□

4.1 Key Delegation

If the underlying scheme CS supports delegation, i.e. there exists an algorithm $\mathsf{DelKey}'(k'_{(f_1,\ldots,f_t)}, f_{t+1}) \to k'_{(f_1,\ldots,f_t,f_{t+1})}$ and CS is correct, constraint-hiding and single-key-selectively unforgeable as per Definition 3.5, then also the amplified construction is. The amplified delegation algorithm delegates the key of CS. It also initializes a new instance of S with each delegation, which is used either to sign x, when the key is used in Sign, or to sign the verification keys of every two neighboring delegation levels, when the key is delegated.

- $\mathsf{DelKey}(\mathsf{sk}_{(f_1,\ldots,f_t)}, f_{t+1})$ takes a key $\mathsf{sk}_{(f_1,\ldots,f_t)} = (\mathsf{vk}', k'_{(f_1,\ldots,f_t)}, \{\mathsf{vk}''_i\}_{i\in[t]}, \mathsf{sk}''_t, \sigma_{(\mathsf{vk}', \mathsf{vk}''_1)}, \{\sigma_{(\mathsf{vk}''_i, \mathsf{vk}''_{i+1})}\}_{i\in[t-1]})$ and a constraint $f_{t+1} \in \mathcal{F}'$. It computes $k'_{(f_1,\ldots,f_t,f_{t+1})} \leftarrow \mathsf{DelKey}'(k'_{(f_1,\ldots,f_t)}, f_{t+1})$. It then generates $(\mathsf{sk}''_{t+1}, \mathsf{vk}''_{t+1}) \leftarrow \mathsf{S.Setup}(1^\lambda)$, signs $\sigma_{(\mathsf{vk}''_t, \mathsf{vk}''_{t+1})} \leftarrow \mathsf{S.Sign}(\mathsf{sk}''_t, (\mathsf{vk}''_t, \mathsf{vk}''_{t+1}))$ and outputs $\mathsf{sk}_{(f_1,\ldots,f_t,f_{t+1})} = (\mathsf{vk}', k'_{(f_1,\ldots,f_{t+1})}, \{\mathsf{vk}''_i\}_{i\in[t+1]}, \mathsf{sk}''_{t+1}, \sigma_{(\mathsf{vk}', \mathsf{vk}''_1)}, \{\sigma_{(\mathsf{vk}''_i, \mathsf{vk}''_{i+1})}\}_{i\in[t]})$.
- $\mathsf{Sign}(x, \mathsf{sk}_{(f_1,\ldots,f_t)})$ takes a key $\mathsf{sk}_{(f_1,\ldots,f_t)} = (\mathsf{vk}', k'_{(f_1,\ldots,f_t)}, \{\mathsf{vk}''_i\}_{i\in[t]}, \mathsf{sk}''_t, \sigma_{(\mathsf{vk}', \mathsf{vk}''_1)}, \{\sigma_{(\mathsf{vk}''_i, \mathsf{vk}''_{i+1})}\}_{i\in[t-1]})$ and an attribute $x \in \mathcal{X}'$. It computes $\sigma'_x \leftarrow \mathsf{Sign}'(x, k'_{(f_1,\ldots,f_t)})$ and $\sigma''_x \leftarrow \mathsf{S.Sign}(\mathsf{sk}''_t, x)$.
  It outputs $\sigma_x = (\mathsf{vk}', \sigma'_x, \{\mathsf{vk}''_i\}_{i\in[t]}, \sigma''_x, \sigma_{(\mathsf{vk}', \mathsf{vk}''_1)}, \{\sigma_{(\mathsf{vk}''_i, \mathsf{vk}''_{i+1})}\}_{i\in[t-1]})$.
- $\mathsf{Ver}_{\mathsf{vk}}(x, \sigma_x)$ accepts only when all of the following conditions hold: $\mathsf{Ver}'_{\mathsf{vk}'}(x, \sigma'_x)$ accepts; $\mathsf{S.Ver}_{\mathsf{S.vk}}(\sigma_{(\mathsf{vk}', \mathsf{vk}''_1)}, (\mathsf{vk}', \mathsf{vk}''_1))$ accepts; $\forall i \in [t-1]$, $\mathsf{S.Ver}_{\mathsf{vk}''_i}(\sigma_{(\mathsf{vk}''_i, \mathsf{vk}''_{i+1})}, (\mathsf{vk}''_i, \mathsf{vk}''_{i+1}))$ accepts; $\mathsf{S.Ver}_{\mathsf{vk}''_t}(\sigma''_x, x)$ accepts.

See Appendix B for correctness, privacy and unforgeability proofs.

5 Equivalence of CS and Homomorphic Signatures

5.1 Recap on Homomorphic Signatures

Our starting point is a (single-data selectively secure) homomorphic signature scheme, which is also context hiding. We use a simplified version of the definition in [GVW15] that suffices for our needs.

Definition 5.1 (Single-Data Homomorphic Signature)**.** *A single-data homomorphic signature scheme is a 4-tuple of PPT algorithms* $(\mathsf{HS.Setup}, \mathsf{HS.Sign}, \mathsf{HS.Eval}, \mathsf{HS.Ver})$ *with the following syntax.*

- $\mathsf{HS.Setup}(1^\lambda) \to (\mathsf{vk}, \mathsf{sk})$ *takes as input the security parameter* λ *and possibly a description of the data-set space* $\mathcal{X}$ *and the functions space* $\mathcal{G}$*. It outputs a verification key* vk *and a signing key* sk*.*
- $\mathsf{HS.Sign}(\mathsf{sk}, x) \to \sigma_x$ *takes as input a signing key* sk *and a data-set* $x \in \mathcal{X}$*, and outputs a signature* σ_x*.*
- $\mathsf{HS.Eval}(g, x, \sigma_x) \to \sigma_{(g,g(x))}$ *takes as input a data-set* $x \in \mathcal{X}$ *and a function* $g \in \mathcal{G}$ *such that* $g(x)$ *is defined, and a signature* σ_x*. It outputs a signature for the pair* $(g, g(x))$*:* $\sigma_{(g,g(x))}$*.*
- $\mathsf{HS.Ver}_{\mathsf{vk}}(g, y, \sigma_{(g,y)})$ *takes as input a function* $g \in \mathcal{G}$*, a value* y *and a signature* $\sigma_{(g,y)}$*, and either accepts or rejects.*

Correctness. *The scheme is correct for a function family* $\mathcal{G}$ *and data-set space* $\mathcal{X}$ *if for all* $x \in \mathcal{X}$ *and* $g \in \mathcal{G}$ *such that* $g(x)$ *is defined, it holds that* $\mathsf{HS.Ver}_{\mathsf{vk}}(g, g(x), \sigma_{(g,g(x))}) = accept$*, where* $\sigma_{(g,g(x))} = \mathsf{HS.Eval}(g, x, \sigma_x)$*,* $\sigma_x = \mathsf{HS.Sign}(\mathsf{sk}, x)$ *and* $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{HS.Setup}(1^\lambda)$*.*

Single-Data Selective Unforgeability. *Fix* $\mathcal{X}, \mathcal{G}$ *and consider the following game between an adversary* $\mathcal{A}$ *and a challenger:*

- $\mathcal{A}$ *sends* $x \in \mathcal{X}$ *to the challenger.*
- *The challenger computes* $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{HS.Setup}(1^\lambda)$ *and* $\sigma_x \leftarrow \mathsf{HS.Sign}_{vk}(\mathsf{sk}, x)$*. It sends to* $\mathcal{A}$ *the values* (vk, σ_x)*.*
- $\mathcal{A}$ *outputs* $(g, y, \sigma_{(g,y)})$*. It wins if* $g \in \mathcal{G}$*,* $\mathsf{HS.Ver}_{\mathsf{vk}}(g, y, \sigma_{(g,y)}) = accept$ *and* $y \neq g(x)$*.*

The scheme is secure for $\mathcal{X}, \mathcal{G}$ *if any PPT* $\mathcal{A}$ *has no more than negligible advantage in this game.*

Context Hiding. *The scheme is context hiding for* $\mathcal{X}, \mathcal{G}$ *if any* PPT *adversary* $\mathcal{A}$ *has no more than negligible advantage in the following game.*

1. *The challenger computes and outputs* $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{HS.Setup}(1^\lambda)$*.*
2. $\mathcal{A}$ *sends* $(g, x_0, x_1) \in \mathcal{G} \times \mathcal{X} \times \mathcal{X}$ *such that* $g(x_0) = g(x_1)$*. Denote this value by* y*.*
3. *The challenger computes* $\sigma_{x_0} \leftarrow \mathsf{HS.Sign}(\mathsf{sk}, x_0)$ *and* $\sigma_{x_1} \leftarrow \mathsf{HS.Sign}(\mathsf{sk}, x_1)$*. It then samples* $b \xleftarrow{\$} \{0,1\}$ *and computes* $\sigma_{(g,y)} \leftarrow \mathsf{HS.Eval}(g, x_b, \sigma_{x_b})$*. It sends* $(\sigma_{x_0}, \sigma_{x_1}, \sigma_{(g,y)})$ *to* $\mathcal{A}$*.*
4. $\mathcal{A}$ *outputs* $b' \in \{0,1\}$ *and wins if and only if* $b' = b$*.*

Note that the correctness requirement also captures the validity of a non-evaluated signature: a signature σ_x for a data-set $x \in \mathcal{X}$ can be verified bit-by-bit using the functions $\{g_i\}_{i \in [|x|]}$, where $g_i(x)$ outputs the ith bit of x. The context hiding property requires that an evaluated signature will not reveal anything about the original (pre-evaluated) signature, other than the evaluation result along with a signature for it.

5.2 Constrained Signatures from Homomorphic Signatures

In this section we construct a (message-policy) CS scheme from context-hiding homomorphic signatures. We assume that the underlying HS scheme is context-hiding and single data-set unforgeable, and show that the resulting CS scheme is single-key-selective unforgeable and key-hiding. Combined with the security amplification from Section 4 (which downgrades the key privacy), this results in a scheme that is fully unforgeable and attribute-hiding.

Let $\mathsf{HS} = (\mathsf{Setup}, \mathsf{Sign}, \mathsf{Eval}, \mathsf{Ver})$ be a homomorphic signature scheme with data-space $\mathcal{X}$ and functions space $\mathcal{F}$. We construct $\mathsf{CS} = (\mathsf{Setup}, \mathsf{Keygen}, \mathsf{Sign}, \mathsf{Ver})$ for $(\mathcal{X}, \mathcal{F})$.

- $\mathsf{CS.Setup}(1^\lambda)$. Initialize the HS scheme $(\mathsf{HS.sk}, \mathsf{HS.vk}) \leftarrow \mathsf{HS.Setup}(1^\lambda)$ and output $\mathsf{vk} = \mathsf{HS.vk}$ and $\mathsf{msk} = \mathsf{HS.sk}$.
- $\mathsf{CS.Keygen}(x, \mathsf{msk})$. Sign x using HS: $\mathsf{HS}.\sigma_x \leftarrow \mathsf{HS.Sign}(\mathsf{HS.sk}, x)$. Output $\mathsf{sk}_x = \mathsf{HS}.\sigma_x$.
- $\mathsf{CS.Sign}(f, \mathsf{sk}_x)$. Use σ_x to homomorphically compute a context-hiding signature for $y = f(x)$. That is, compute and output $\sigma_f = \mathsf{HS}.\sigma_{(f,f(x))} \leftarrow \mathsf{HS.Eval}(f, x, \sigma_x)$.
- $\mathsf{CS.Ver}_{\mathsf{vk}}(f, \sigma_f)$. Accept if and only if $\mathsf{HS.Ver}_{\mathsf{vk}}(f, 0, \sigma_f)$ accepts.

Lemma 5.1 (Correctness)**.** *The scheme is correct for* $(\mathcal{F}, \mathcal{X})$.

Proof. Fix $(x, f) \in \mathcal{X} \times \mathcal{F}$ such that $f(x) = 0$. Consider $(\mathsf{msk}, \mathsf{vk}) \leftarrow \mathsf{CS.Setup}(1^\lambda)$, $\mathsf{sk}_x \leftarrow \mathsf{CS.Keygen}(x, \mathsf{msk})$ and $\sigma_f = \mathsf{CS.Sign}(f, \mathsf{sk}_x)$. Then it holds that $\sigma_f \leftarrow \mathsf{HS.Eval}(f, x, \mathsf{HS.Sign}(\mathsf{HS.sk}, x))$. We need to show that $\mathsf{Ver}_{\mathsf{vk}}(f, \sigma_f) = \mathit{accept}$, i.e. that $\mathsf{HS.Ver}_{\mathsf{vk}}(f, 0, \mathsf{HS}.\sigma_{(f,f(x))})$ accepts. Indeed, $f(x) = 0$ by assumption, thus the result follows by the correctness of HS. □

Lemma 5.2 (Privacy)**.** *The scheme is key-hiding for* $(\mathcal{F}, \mathcal{X})$.

Proof. Assume towards contradiction an adversary $\mathcal{A}_c$ that wins the privacy game with noticeable advantage and use it to break the context hiding property of the underlying HS scheme as follows:

1. Receive $(\mathsf{HS.sk}, \mathsf{HS.vk}) \leftarrow \mathsf{HS.Setup}(1^\lambda)$ from the HS challenger and forward it to $\mathcal{A}_c$ as $(\mathsf{msk}, \mathsf{vk})$.
2. Receive from $\mathcal{A}_c$ a tuple (x_0, x_1, f) such that $f(x_0) = f(x_1) = 0$. Forward (x_0, x_1, f) to the HS challenger.
3. Receive from the HS challenger the challenge $(\mathsf{HS}.\sigma_{x_0}, \mathsf{HS}.\sigma_{x_1}, \mathsf{HS}.\sigma_{(f,0)})$ and forward it to $\mathcal{A}_c$ as $(\mathsf{sk}_{x_0}, \mathsf{sk}_{x_1}, \sigma_f)$.
4. Get b' from $\mathcal{A}_c$ and forward it to the HS challenger. Clearly, any advantage of $\mathcal{A}_c$ induces an advantage of the reduction.

□

Lemma 5.3 (Unforgeability)**.** *The scheme is single-key-selectively unforgeable for* $(\mathcal{F}, \mathcal{X})$.

Proof. Consider the CS single-key selective security game against an adversary $\mathcal{A}_c$. Let $x \in \mathcal{X}$ be the attribute sent by $\mathcal{A}_c$, and assume towards contradiction that it wins the game. Then $\mathcal{A}_c$ outputs (f, σ_f) such that $\mathsf{CS.Ver}_{\mathsf{vk}}(f, \sigma_f) = \mathit{accept}$ and $f(x) \neq 0$. Such adversary can be used to break the unforgeability of HS:

1. Upon receiving x from $\mathcal{A}_c$, send it to the HS challenger.

2. The HS challenger sends back $\mathsf{HS.vk}$ and $\mathsf{HS}.\sigma_x = \mathsf{HS.Sign}(\mathsf{HS.sk}, x)$, which is exactly $(\mathsf{vk}, \mathsf{sk}_x)$ that we have to send to $\mathcal{A}_c$.

3. $\mathcal{A}_c$ sends back (f, σ_f) such that $\mathsf{Ver}_{\mathsf{vk}}(f, \sigma_f) = \mathit{accept}$ and $f(x) \neq 0$. Denoting $\sigma_f = \mathsf{HS}.\sigma_{(f,f(x))}$, it means that $\mathsf{HS.Ver}_{\mathsf{vk}}(f, 0, \mathsf{HS}.\sigma_{(f,f(x))}) = \mathit{accept}$ while $f(x) \neq 0$, therefore $\mathsf{HS}.\sigma_{(f_m,f_m(x))}$ is a successful forgery against HS.

□

5.3 Homomorphic Signatures from Constrained Signatures

We show how to construct a context-hiding 1-hop homomorphic signatures scheme from (message-policy) CS. We assume that the underlying CS scheme is single-key-selective unforgeable and key-hiding, and show that the resulting HS scheme is context-hiding and selectively unforgeable. As shown in [GVW15], it is possible to construct an adaptively unforgeable HS scheme from a selectively unforgeable HS scheme.

Let $\mathsf{CS} = (\mathsf{Setup}, \mathsf{Keygen}, \mathsf{Sign}, \mathsf{Ver})$ be a constrained signatures scheme with attribute space $\mathcal{X}$ and constraint space $\mathcal{F}$. We construct $\mathsf{HS} = (\mathsf{Setup}, \mathsf{Sign}, \mathsf{Eval}, \mathsf{Ver})$ for data-set space $\mathcal{X}$ and functions space $\mathcal{G} : \mathcal{X} \to \mathcal{Y}$, where the requirement is that for any $(g, y) \in \mathcal{G} \times \mathcal{Y}$, it holds that $f_{(g,y)} \in \mathcal{F}$, where $f_{(g,y)} : \mathcal{X} \to \{0, 1\}$ is a function that on input x returns 0 if and only if $g(x) = y$.

- $\mathsf{HS.Setup}(1^\lambda)$. Initialize the CS scheme: compute $(\mathsf{CS.msk}, \mathsf{CS.vk}) \leftarrow \mathsf{CS.Setup}(1^\lambda)$. Output $\mathsf{vk} = \mathsf{CS.vk}$ and $\mathsf{sk} = \mathsf{CS.msk}$.

- $\mathsf{HS.Sign}(x, \mathsf{sk})$. Compute and output $\sigma_x = \mathsf{CS.sk}_x \leftarrow \mathsf{CS.Keygen}(x, \mathsf{CS.msk})$.

- $\mathsf{HS.Eval}(g, \sigma_x)$. Let $y = g(x)$. Define the circuit $f_{(g,y)} : \mathcal{X} \to \{0, 1\}$ that on input x returns 0 if and only if $g(x) = y$. Use $\mathsf{CS.sk}_x$ to sign the policy $f_{(g,y)}$. That is, compute and output $\sigma_{(g,y)} = \mathsf{CS}.\sigma_{f_{(g,y)}} \leftarrow \mathsf{CS.Sign}(f_{(g,y)}, \mathsf{CS.sk}_x)$.

- $\mathsf{HS.Ver}_{\mathsf{vk}}(g, y, \sigma_{(g,y)})$. Accept if and only if $\mathsf{CS.Ver}_{\mathsf{vk}}(f_{(g,y)}, \mathsf{CS}.\sigma_{f_{(g,y)}})$ accepts.

Lemma 5.4 (Correctness)**.** *The scheme is correct for* $(\mathcal{G}, \mathcal{X})$.

Proof. Fix $(x, g) \in \mathcal{X} \times \mathcal{G}$. Consider $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{HS.Setup}(1^\lambda)$, $\sigma_x \leftarrow \mathsf{HS.Sign}(x, \mathsf{sk})$ and $\sigma_{(g,y)} = \mathsf{HS.Eval}(g, \sigma_x)$, where $y = g(x)$. Then it holds that $\sigma_{(g,y)} = \mathsf{CS}.\sigma_{f_{(g,y)}} = \mathsf{CS.Sign}(f_{(g,y)}, \mathsf{CS.sk}_x)$, where $\mathsf{CS.sk}_x \leftarrow \mathsf{CS.Keygen}(x, \mathsf{CS.msk})$. We need to show that $\mathsf{HS.Ver}_{\mathsf{vk}}(g, y, \sigma_{(g,y)}) = \mathit{accept}$, i.e. that $\mathsf{CS.Ver}_{\mathsf{vk}}(f_{(g,y)}, \mathsf{CS}.\sigma_{f_{(g,y)}})$ accepts. Indeed, $g(x) = y$ and therefore $f_{(g,y)}(x) = 0$, and thus $\mathsf{CS.Ver}_{\mathsf{vk}}(f_{(g,y)}, \mathsf{CS}.\sigma_{f_{(g,y)}})$ accepts by the correctness of CS. □

Lemma 5.5 (Privacy)**.** *The scheme is context-hiding for* $(\mathcal{G}, \mathcal{X})$.

Proof. Assume towards contradiction an adversary $\mathcal{A}_h$ that wins the context-hiding game with noticeable advantage, and use it to break the key-privacy the underlying MPCS scheme as follows:

1. Receive $(\mathsf{CS.msk}, \mathsf{CS.vk}) \leftarrow \mathsf{CS.Setup}(1^\lambda)$ from the CS challenger and forward it to $\mathcal{A}_h$ as $(\mathsf{sk}, \mathsf{vk})$.

2. Receive from $\mathcal{A}_h$ a tuple (g, x_0, x_1) such that $g(x_0) = g(x_1)$ and denote this value by y. Forward $(x_0, x_1, f_{(g,y)})$ to the CS challenger.

3. Receive from the CS challenge the challenge $(\mathsf{CS.sk}_{x_0}, \mathsf{CS.sk}_{x_1}, \mathsf{CS}.\sigma_{f_{(g,y)}})$ and forward it to $\mathcal{A}_h$ as $(\sigma_{x_0}, \sigma_{x_1}, \sigma_{(g,y)})$.

4. Get b' from $\mathcal{A}_h$ and forward it to the CS^y challenger. Clearly, any advantage of $\mathcal{A}_h$ induces an advantage of the reduction.

□

Lemma 5.6 (Unforgeability)**.** *The scheme is single-data selectively unforgeable for* $(\mathcal{G}, \mathcal{X})$*.*

Proof. Consider the HS single-data selective unforgeability game against an adversary $\mathcal{A}_h$. Let $x \in \mathcal{X}$ be the data-set sent by $\mathcal{A}_h$, and assume towards contradiction that it wins the game. Then $\mathcal{A}_h$ outputs $(g, y, \sigma_{(g,y)})$ such that $\mathsf{HS.Ver}_{\mathsf{vk}}(g, y, \sigma_{(g,y)}) = \mathit{accept}$ and $g(x) \neq y$. Such adversary can be used to break the unforgeability of MPCS:

1. Upon receiving x from $\mathcal{A}_h$, send it to the CS challenger.

2. The CS challenger sends back $\mathsf{CS.sk}_x = \mathsf{CS.Keygen}(\mathsf{CS.msk}, x)$ and $\mathsf{CS.vk}$, which is exactly (σ_x, vk) that we have to send to $\mathcal{A}_h$.

3. $\mathcal{A}_h$ sends back $(g, y, \sigma_{(g,y)})$ such that $\mathsf{HS.Ver}_{\mathsf{vk}}(g, y, \sigma_{(g,y)}) = \mathit{accept}$ and $g(x) \neq y$. Denoting $\sigma_{(g,y)} = \mathsf{CS}.\sigma_{f_{(g,y)}}$, it means that $\mathsf{CS.Ver}_{\mathsf{vk}}(f_{(g,y)}, \mathsf{CS}.\sigma_{f_{(g,y)}}) = \mathit{accept}$, however $g(x) \neq y$ and therefore $f_{(g,y)}(x) \neq 0$, thus $\mathsf{CS}.\sigma_{f_{(g,y)}}$ is a successful forgery against CS.

□

6 CS Construction from Lattice Trapdoors

In this section we construct a (key-policy) CS scheme from lattices trapdoors, using techniques that were developed in [GVW13, BGG+14]. The resulting scheme supports a fixed attribute space, and the constraint space consists of boolean circuits with a bound on depth. We prove message-selective unforgeability based on the SIS hardness assumption, and statistical key-hiding. Lastly we show how to extend the scheme to support key delegation.

The initialization parameters are (ℓ, d), where the attribute space is $\mathcal{X} = \{0,1\}^\ell$ and the constraint space is all d-depth bounded circuits $\mathcal{F}_d = \{f : \{0,1\}^\ell \to \{0,1\}\}$.

6.1 The Scheme

Initialize the parameters $n, m, m', q, B, \tau_0, \tau_k, \tau_s$ respective to λ, d, ℓ as described below.

- $\mathsf{Setup}(1^\lambda) \to (\mathsf{msk}, \mathsf{vk})$: Generate a matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m'}$ with its trapdoor $\mathbf{A}_{\tau_0}^{-1}$ (see Corollary 2.2). Sample uniformly a matrix $\vec{\mathbf{A}} \xleftarrow{\$} \mathbb{Z}_q^{n \times (m \times \ell)}$. Output $\mathsf{vk} = (\mathbf{A}, \vec{\mathbf{A}})$ and $\mathsf{msk} = \mathbf{A}_{\tau_0}^{-1}$.

- $\mathsf{Keygen}_{\mathsf{vk}}(f, \mathsf{msk}) \to \mathsf{sk}_f$: Compute $\mathbf{H}_f = \mathsf{EvalF}(f, \vec{\mathbf{A}})$ (see Theorem 2.7) and $\mathbf{A}_f = \vec{\mathbf{A}} \cdot \mathbf{H}_f$, then use $\mathbf{A}_{\tau_0}^{-1}$ to compute $\mathsf{sk}_f = [\mathbf{A} \| \mathbf{A}_f]_{\tau_k}^{-1}$ (see Corollary 2.4).

- $\mathsf{Sign}_{\mathsf{pp}}(x, \mathsf{sk}_f) \to \sigma_x$: If $f(x) \neq 0$ return $\perp$. Otherwise, compute $\mathbf{H}_{f,x} = \mathsf{EvalFX}(f, x, \vec{\mathbf{A}})$ (see Theorem 2.7). Note that by this theorem, $[\vec{\mathbf{A}} - x \otimes \mathbf{G}] \cdot \mathbf{H}_{f,x} = \mathbf{A}_f - f(x)\mathbf{G} = \mathbf{A}_f$. Now apply trapdoor extension (see Theorem 2.3) with

$$\bar{\mathbf{A}} = [\mathbf{A} \| \mathbf{A}_f], \qquad \bar{\mathbf{B}} = [\mathbf{A} \| \vec{\mathbf{A}} - x \otimes \mathbf{G}], \qquad \mathbf{S} = \begin{bmatrix} \mathbf{I}_{m'} & \mathbf{0} \\ \mathbf{0} & \mathbf{H}_{f,x} \end{bmatrix}$$

  (using $\mathsf{sk}_f = [\mathbf{A} \| \mathbf{A}_f]^{-1}_{\tau_k} = \bar{\mathbf{A}}^{-1}_{\tau_k}$), and achieve $\bar{\mathbf{B}}^{-1}_{\tau_s} = [\mathbf{A} \| \vec{\mathbf{A}} - x \otimes \mathbf{G}]^{-1}_{\tau_s}$. Sample $\sigma_x \xleftarrow{\$} [\mathbf{A} \| \vec{\mathbf{A}} - x \otimes \mathbf{G}]^{-1}_{\tau_s}(\mathbf{0})$ and output σ_x.

  Note that by Theorem 2.7, $\mathbf{H}_{f,x} \in \mathbb{Z}^{\ell m \times m}$ and $\|\mathbf{H}_{f,x}\|_\infty \leq (2m)^d$, and thus the largest singular value $s_1(\mathbf{S}) = \max\{1, s_1(\mathbf{H}_{f,x})\} \leq \sqrt{\ell} 2^d m^{d+1}$. Hence $\tau_k \cdot s_1(\mathbf{S}) \leq \tau_s = \tau_k \cdot \sqrt{\ell} 2^d m^{d+1}$, as required by the conditions of Theorem 2.3.

- $\mathsf{Ver}_{\mathsf{pp}}(x, \sigma_x) \to \{accept, reject\}$: Output *accept* if and only if the following conditions hold: $\sigma_x \neq \perp$, $\sigma_x \neq \mathbf{0}$, $[\mathbf{A} \| \vec{\mathbf{A}} - x \otimes \mathbf{G}] \cdot \sigma_x = \mathbf{0}$ and $\|\sigma_x\|_\infty \leq B$.

Choice of Parameters. The SIS parameters n, q, B' are chosen according to constraints from the correctness and security analyses that follow. We require that $n \geq \lambda$, $q \leq 2^n$ and recall that $\ell = \mathrm{poly}(\lambda) \leq 2^n$. We set $m = n\lceil \log q \rceil$, $m' = \max\{m_0, (n+1)\lceil \log q \rceil + 2\lambda\}$, where m_0 is as required by $\mathsf{TrapGen}$ (see 2.2), $\tau_0 = O(\sqrt{n \lceil \log q \rceil \log n})$ as required by $\mathsf{TrapGen}$ (see 2.2), $\tau_k = \max\{\sqrt{m'} \ell 2^d m^{1.5+d}, \tau_0\}$, $\tau_s = \tau_k \cdot \sqrt{\ell} 2^d m^{d+1}$, $B = \tau_s \sqrt{m' + \ell \cdot m}$, and we require that $(\ell m + 1)B \leq B'$, i.e. that $(\ell m + 1)\sqrt{m'} \ell^{1.5} 2^{2d} m^{2d+2.5} \sqrt{m' + \ell \cdot m} \leq B'$, while keeping $\mathrm{SIS}_{n,q,B',m'}$ hard as per Theorem 2.1. These constraints can be met by setting $n = d^{\frac{1}{\epsilon}} + \ell$, $B' = 2^{n^\epsilon}$ and then choosing q accordingly based on Theorem 2.1. Note that it guarantees that indeed $q \leq 2^n$ and $(\ell m + 1)\sqrt{m'} \ell^{1.5} 2^{2d} m^{2d+2.5} \sqrt{m' + \ell \cdot m} \leq B'$.

Correctness and Security. We prove correctness and security for the attribute space $\mathcal{X} = \{0,1\}^\ell$ and function family $\mathcal{F}_d = \{f : \{0,1\}^\ell \to \{0,1\}\}$ of circuits with depth at most d.

Lemma 6.1 (Correctness)**.** *The scheme is correct for* $(\mathcal{X}, \mathcal{F})$.

Proof. Fix $x \in \mathcal{X}$ and $f \in \mathcal{F}$ for which $f(x) = 0$, and consider $(\mathsf{msk}, \mathsf{vk}) \leftarrow \mathsf{Setup}(1^\lambda)$ and $\sigma_x = \mathsf{Sign}_{\mathsf{vk}}(x, \mathsf{Keygen}_{\mathsf{vk}}(f, \mathsf{msk}))$. Then since $f(x) = 0$, $\sigma_x \in [\mathbf{A} \| \vec{\mathbf{A}} - x \otimes \mathbf{G}]^{-1}_{\tau_s}(\mathbf{0})$ and therefore $[\mathbf{A} \| \vec{\mathbf{A}} - x \otimes \mathbf{G}] \cdot \sigma_x = 0$. By the properties of lattice trapdoors, samples from $[\mathbf{A} \| \vec{\mathbf{A}} - x \otimes \mathbf{G}]^{-1}_{\tau_s}(\mathbf{0})$ are within 2^{-n} statistical distance from a discrete Gaussian distribution over $\mathbb{Z}_q^{m' + \ell \cdot m}$ with parameter τ_s. Therefore, with all but $2^{-(m' + \ell \cdot m)} = \mathrm{negl}(\lambda)$ probability, $\left\|\sigma_{(x,m)}\right\|_\infty \leq \tau_s \sqrt{m' + \ell \cdot m} = B$ and hence $\mathsf{Ver}_{\mathsf{vk}}(x, \sigma_x) = accept$. □

Lemma 6.2 (Privacy)**.** *The scheme is statistically key-hiding for* $(\mathcal{X}, \mathcal{F})$.

Proof. Consider the key-hiding privacy game from Definition 3.2. Change the way that $\sigma_{x,b}$ is generated in the challenge: use $\mathsf{msk} = \mathbf{A}^{-1}_{\tau_0}$ to compute $[\mathbf{A} \| \vec{\mathbf{A}} - x \otimes \mathbf{G}]^{-1}_{\tau_s}$ (note that $\tau_s \geq \tau_0$ and see Corollary 2.4), then sample and output $\sigma_{x,b} \xleftarrow{\$} [\mathbf{A} \| \vec{\mathbf{A}} - x \otimes \mathbf{G}]^{-1}_{\tau_s}(\mathbf{0})$. The distribution from which $\sigma_{x,b}$ is sampled remains the same, therefore this change is statistically indistinguishable. In this setting, the challenge is independent of b and thus any adversary has no advantage in the game. □

Lemma 6.3 (Unforgeability)**.** *The scheme is message-selective unforgeable for* $(\mathcal{X}, \mathcal{F})$.

Proof. The proof proceeds with a sequence of hybrids and follows similar lines to [BGG+14].

Hybrid $\mathcal{H}_0$. The message-selective unforgeability game from Definition 3.3.

Hybrid $\mathcal{H}_1$. Upon receiving x^*, the challenger generates vk as follows: it generates $\mathbf{A}$ along with $\mathbf{A}_{\tau_0}^{-1}$ as before, then it samples a matrix $\vec{\mathbf{R}}_A \xleftarrow{\$} \{0,1\}^{m' \times \ell_{\mathcal{X}} m}$ and computes $\vec{\mathbf{A}} = \mathbf{A}\vec{\mathbf{R}}_A + x^* \otimes \mathbf{G}$. Indistinguishability follows from the extended leftover hash lemma , since $m' \geq (n+1)\lceil \log q \rceil + 2\lambda$ and $\mathbf{A}$ is statistically-close to uniform by Corollary 2.2.

Hybrid $\mathcal{H}_2$. Change the way that the challenger answers key queries. Let f be a query, then $f(x^*) = 1$ and thus $f(x^*) = 1$, thus by Theorem 2.7

$$[\mathbf{A} \| \mathbf{A}_f - \mathbf{G}] = [\mathbf{A} \| \mathbf{A}_f - f(x^*)\mathbf{G}] = [\mathbf{A} \| [\vec{\mathbf{A}} - x^* \otimes \mathbf{G}] \cdot \mathbf{H}_{f,x^*}] = [\mathbf{A} \| \mathbf{A} \cdot \vec{\mathbf{R}}_A \cdot \mathbf{H}_{f,x^*}] \ .$$

Hence $[\mathbf{A} \| \mathbf{A}_f] = [\mathbf{A} \| \mathbf{A} \cdot \vec{\mathbf{R}}_A \cdot \mathbf{H}_{f,x^*} + \mathbf{G}]$, and by Corollary 2.5 it is possible to compute $\mathsf{sk}_f = [\mathbf{A} \| \mathbf{A}_f]_{\tau_k}^{-1} = [\mathbf{A} \| \mathbf{A} \cdot \vec{\mathbf{R}}_A \cdot \mathbf{H}_{f,x^*} + \mathbf{G}]_{\tau_k}^{-1}$ given $\mathbf{A}, \vec{\mathbf{R}}_A$ and $\mathbf{H}_{f,x^*}$, since $\|\mathbf{H}_{f,x}\|_\infty \leq (2m)^d$ and thus

$$\sqrt{m'm} \left\| \vec{\mathbf{R}}_A \cdot \mathbf{H}_{f,x} \right\|_\infty \leq \sqrt{m'} \ell m^{1.5} \cdot \left\| \vec{\mathbf{R}}_A \right\|_\infty \cdot \|\mathbf{H}_{f,x}\|_\infty \leq \sqrt{m'} \ell 2^d m^{1.5+d} \leq \tau_k \ .$$

The distribution of sk_f remains the same, thus the hybrids are statistically indistinguishable.

Hybrid $\mathcal{H}_3$. Change the way that the challenger answers signature queries. Let (f, x) be a query, then $x \neq x^*$ and $f(x) = 0$. Consider the function $f_x : \{0,1\}^\ell \to \{0,1\}$ that returns 0 if the input is x, and 1 otherwise. Then since $x \neq x^*$, $f_x(x^*) = 1$, hence we can generate a sk_{f_x} respective to the function f_x as described in the previous hybrid. In this hybrid we compute a signature for x using this sk_{f_x}, i.e. output $\mathsf{Sign}_{\mathsf{vk}}(x, \mathsf{sk}_{f_x})$. Since $f_x(x) = 0$ and the scheme is statistically constraint-hiding, this change is statistically indistinguishable.

Hybrid $\mathcal{H}_4$. Change the way that the challenger answers repeated signature queries. Let $(i, x) \in \mathbb{N} \times \mathcal{X}$ be a query. Compute and output $\mathsf{Sign}_{\mathsf{vk}}(x, \mathsf{sk}_{f_x})$, where sk_{f_x} is as described above. Since the scheme is statistically key-hiding, this change is statistically indistinguishable.

Hybrid $\mathcal{H}_5$. At this point the challenger does not use $\mathbf{A}_{\tau_0}^{-1}$ anymore. We switch to sampling $\mathbf{A}$ uniformly without $\mathbf{A}_{\tau_0}^{-1}$, which is statistically indistinguishable by Corollary 2.2.

Finally we show that if $\mathcal{A}$ wins the game in this hybrid then it breaks $\mathrm{SIS}_{n,q,B'}$: Let $\mathbf{A}$ be a $\mathrm{SIS}_{n,q,B',m'}$ challenge. Initialize a game against $\mathcal{A}$ as in this hybrid using the matrix $\mathbf{A}$. Assume that $\mathcal{A}$ produces a valid forgery σ_{x^*} for x^*. Then $\sigma_{x^*} \neq \mathbf{0}$, $\|\sigma_{x^*}\|_\infty \leq B$ and

$$\mathbf{0} = [\mathbf{A} \| \vec{\mathbf{A}} - x^* \otimes \mathbf{G}] \cdot \sigma_{x^*} = [\mathbf{A} \| \mathbf{A}\vec{\mathbf{R}}_A] \cdot \sigma_{x^*} = \mathbf{A} \cdot [\mathbf{I} \| \vec{\mathbf{R}}_A] \cdot \sigma_{x^*} \ .$$

Since

$$\left\| [\mathbf{I} \| \vec{\mathbf{R}}_A] \cdot \sigma_{x^*} \right\|_\infty \leq (\ell m + 1) \|\sigma_{x^*}\|_\infty = (\ell m + 1) B \leq B' \ ,$$

$[\mathbf{I} \| \vec{\mathbf{R}}_A] \cdot \sigma_{x^*}$ is a valid solution to $\mathrm{SIS}_{n,q,B',m'}$. □

6.2 Adding Key Delegation

It is possible to extend the construction to support key delegation as per Definition 3.5. We define an alternative Sign^{del} algorithm along with a new DelKey algorithm. Note that by definition each key maintains its delegation history: an ordered list of constraints which define the permissions of the key. Upon computing $\mathsf{DelKey}_{\mathsf{vk}}(\mathsf{sk}_{(f_1,\ldots,f_t)}, f_{t+1}) \to \mathsf{sk}_{(f_1,\ldots,f_{t+1})}$, the delegated key $\mathsf{sk}_{(f_1,\ldots,f_{t+1})}$ contains the constraints list of $\mathsf{sk}_{(f_1,\ldots,f_t)}$ and the new constraint f_{t+1}. The scheme should be parameterized with an upper bound t' to the delegation depth (i.e. the list length). The other parameters are initialized as before, with the only differences $\tau_s = \tau_k \cdot \sqrt{\ell t'} 2^d m^{d+1}$ and $n = d^{\frac{1}{\epsilon}} + \ell t'$. Hence the scheme can be initializes with any $t' = \mathrm{poly}(\lambda)$.

- $\mathsf{DelKey}_{\mathsf{vk}}(\mathsf{sk}_{(f_1,\ldots,f_t)}, f_{t+1}) \to \mathsf{sk}_{(f_1,\ldots,f_{t+1})}$: Recall that when $t = 1$, $\mathsf{sk}_f = [\mathbf{A}\|\mathbf{A}_f]_{\tau_k}^{-1}$. Assume that for any $t \geq 1$, $\mathsf{sk}_{f_1,\ldots,f_t} = [\mathbf{A}\|\mathbf{A}_{f_1}\|\ldots\|\mathbf{A}_{f_t}]_{\tau_k}^{-1}$, and compute the new key as follows: Compute $\mathbf{H}_{f_{t+1}} = \mathsf{EvalF}(f_{t+1}, \vec{\mathbf{A}})$ (see Theorem 2.7) and $\mathbf{A}_{f_{t+1}} = \vec{\mathbf{A}} \cdot \mathbf{H}_{f_{t+1}}$, then use $[\mathbf{A}\|\mathbf{A}_{f_1}\|\ldots\|\mathbf{A}_{f_t}]_{\tau_k}^{-1}$ to compute and output $sk_{f_1,\ldots,f_{t+1}} = [\mathbf{A}\|\mathbf{A}_{f_1}\|\ldots\|\mathbf{A}_{f_{t+1}}]_{\tau_k}^{-1}$ (see Corollary 2.4).

- $\mathsf{Sign}_{\mathsf{pp}}(x, sk_{f_1,\ldots,f_t}) \to \sigma_x$: If $\exists i \in [t]$ s.t. $f_i(x) \neq 0$, return $\perp$. Otherwise, for $i \in [t]$ compute $\mathbf{H}_{f_i,x} = \mathsf{EvalFX}(f_i, x, \vec{\mathbf{A}})$ (see Theorem 2.7). Note that by this theorem, $[\vec{\mathbf{A}} - x \otimes \mathbf{G}] \cdot \mathbf{H}_{f_i,x} = \mathbf{A}_{f_i} - f_i(x)\mathbf{G} = \mathbf{A}_{f_i} - f_i(x)\mathbf{G} = \mathbf{A}_{f_i}$. Now apply the Trapdoor Extension Theorem (2.3) with

$$\bar{\mathbf{A}} = [\mathbf{A}\|\mathbf{A}_{f_1}\|\ldots\|\mathbf{A}_{f_t}], \qquad \bar{\mathbf{B}} = [\mathbf{A}\|\vec{\mathbf{A}} - x \otimes \mathbf{G}], \qquad \mathbf{S} = \begin{bmatrix} \mathbf{I}_{m'} & \mathbf{0} & \ldots & \mathbf{0} \\ \mathbf{0} & \mathbf{H}_{f_1,x} & \ldots & \mathbf{H}_{f_t,x} \end{bmatrix}$$

  (using $\mathsf{sk}_f = \bar{\mathbf{A}}_{\tau_k}^{-1}$), and achieve $\bar{\mathbf{B}}_{\tau_s}^{-1} = [\mathbf{A}\|\vec{\mathbf{A}} - x \otimes \mathbf{G}]_{\tau_s}^{-1}$. Finally sample and output $\sigma_x \xleftarrow{\$} [\mathbf{A}\|\vec{\mathbf{A}} - x \otimes \mathbf{G}]_{\tau_s}^{-1}(\mathbf{0})$.

  Note that by Theorem 2.7, $\forall i \in [t] : \mathbf{H}_{f_i,x} \in \mathbb{Z}^{\ell m \times m}$ and $\|\mathbf{H}_{f_i,x}\|_\infty \leq (2m)^d$, and thus the largest singular value $s_1(\mathbf{S}) \leq \sqrt{\ell t} 2^d m^{d+1}$. Hence $\tau_k \cdot s_1(\mathbf{S}) \leq \tau_k \cdot \sqrt{\ell t'} 2^d m^{d+1} = \tau_s$, as required by the conditions of Theorem 2.3.

Correctness and Security. Correctness and statistical key-hiding can be proved the same way as in the non-delegatable scheme, since for each x the valid signatures distribution remains the same: $[\mathbf{A}\|\vec{\mathbf{A}} - x \otimes \mathbf{G}]_{\tau_s}^{-1}(\mathbf{0})$. We now prove message-selective unforgeability as per Definition 3.5.

Lemma 6.4. *The scheme is message-selective unforgeable for $(\mathcal{X}, \mathcal{F})$.*

Proof. We first define the procedure $\mathsf{PermuteKey}(\mathsf{sk}_{f_1,\ldots,f_t}, \rho) \to \mathsf{sk}_{f_{\rho(1)},\ldots,f_{\rho(t)}}$ that takes as input a signing key $\mathsf{sk}_{f_1,\ldots,f_t}$ and a permutation $\rho : \mathbb{Z}_t \to \mathbb{Z}_t$, and outputs a key of the permuted constraints $\mathsf{sk}_{f_{\rho(1)},\ldots,f_{\rho(t)}}$. $\mathsf{PermuteKey}$ works as follows: Recall that $\mathsf{sk}_{f_1,\ldots,f_t} = [\mathbf{A}\|\mathbf{A}_{f_1}\|\ldots\|\mathbf{A}_{f_t}]_{\tau_k}^{-1}$, thus by Corollary 2.6, it is efficient to compute $\mathsf{sk}_{f_{\rho(1)},\ldots,f_{\rho(t)}} = [\mathbf{A}\|\mathbf{A}_{f_{\rho(1)}}\|\ldots\|\mathbf{A}_{f_{\rho(t)}}]_{\tau_k}^{-1}$.

The security proof goes by reduction to the security of the non-delegatable scheme. Assume an adversary $\mathcal{A}_{del}$ that wins the delegation security game, and use it to win the security game without delegation against a challenger $\mathsf{Challenger}$ as follows:

1. Receive x^* from $\mathcal{A}_{del}$ and forward it to $\mathsf{Challenger}$.
2. Receive vk from $\mathsf{Challenger}$ and forward it to $\mathcal{A}_{del}$.

3. Answer $\mathcal{A}_{del}$'s queries as follows:

   - If the query is a key query, i.e. it is of the form $t \in \mathbb{N}$, $F \in \mathcal{F}^t$ such that $\exists f \in F$ for which $f(x^*) = 1$, request sk_{f_i} from $\mathsf{Challenger}$. Then compute $\mathsf{sk}_{(f,F/f)}$ using DelKey $|F| - 1$ times and sk_f. Finally compute sk_F using $\mathsf{PermuteKey}$ and $\mathsf{sk}_{(f,F/f)}$, and send it to $\mathcal{A}_{del}$.
   - If the query is a signature query, i.e. it is of the form $t \in \mathbb{N}$, $(F, x) \in \mathcal{F}^t \times \mathcal{X}$ such that $x \neq x^*$ and $\forall f \in F : f(x) = 0$, request σ_x from $\mathsf{Challenger}$ using an arbitrary $f \in F$, i.e. send (x, f) and get back σ_x. Forward the signature to $\mathcal{A}_{del}$. Recall that in the unforgeability game, those queries should be answered by computing $\sigma_x \leftarrow \mathsf{Sign}(x, \mathsf{sk}_F)$. Since the construction is key-hiding, this is indistinguishable to $\mathcal{A}_{del}$.
   - If the query is a repeated signature query, i.e. it is of the form $i \in \mathbb{N}$, $x \in \mathcal{X}$ such that $x \neq x^*$ and the ith signature query (F_i, x_i) satisfies $\forall f \in F_i : f(x) = 0$, answer it as described above as if it were a signature query of the form (F_i, x). Recall that in the unforgeability game, those queries should be answered by computing $\sigma_x \leftarrow \mathsf{Sign}(x, \mathsf{sk}_{F_i})$, where sk_{F_i} is a key that was generated when the ith signature query was answered. Since the construction is key-hiding, this is indistinguishable to $\mathcal{A}_{del}$.

4. Get a forgery σ_{x^*} from $\mathcal{A}_{del}$ and forward it to $\mathsf{Challenger}$.

If $\mathcal{A}_{del}$ wins the game then also the reduction does, with contradiction to the security of the non-delegatable scheme. □

References

[ABB10] Shweta Agrawal, Dan Boneh, and Xavier Boyen. Lattice basis delegation in fixed dimension and shorter-ciphertext hierarchical IBE. In Tal Rabin, editor, *Advances in Cryptology - CRYPTO 2010, 30th Annual Cryptology Conference, Santa Barbara, CA, USA, August 15-19, 2010. Proceedings*, volume 6223 of *Lecture Notes in Computer Science*, pages 98–115. Springer, 2010.

[Ajt96] Miklós Ajtai. Generating hard instances of lattice problems (extended abstract). In Gary L. Miller, editor, *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing, Philadelphia, Pennsylvania, USA, May 22-24, 1996*, pages 99–108. ACM, 1996.

[AP14] Jacob Alperin-Sheriff and Chris Peikert. Faster bootstrapping with polynomial error. In Juan A. Garay and Rosario Gennaro, editors, *Advances in Cryptology - CRYPTO 2014 - 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part I*, volume 8616 of *Lecture Notes in Computer Science*, pages 297–314. Springer, 2014.

[BCTW16] Zvika Brakerski, David Cash, Rotem Tsabary, and Hoeteck Wee. Targeted homomorphic attribute-based encryption. In *TCC*, pages 330–360, 2016.

[BF11] Dan Boneh and David Mandell Freeman. Linearly homomorphic signatures over binary fields and new tools for lattice-based signature. In *PKC*, pages 1–16, 2011.

[BF14] Mihir Bellare and Georg Fuchsbauer. Policy-based signatures. In *PKC*, pages 520–537, 2014.

[BGG+14] Dan Boneh, Craig Gentry, Sergey Gorbunov, Shai Halevi, Valeria Nikolaenko, Gil Segev, Vinod Vaikuntanathan, and Dhinakaran Vinayagamurthy. Fully key-homomorphic encryption, arithmetic circuit ABE and compact garbled circuits. In *Advances in Cryptology - EUROCRYPT 2014 - 33rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Copenhagen, Denmark, May 11-15, 2014. Proceedings*, pages 533–556, 2014.

[BGI13] Elette Boyle, Shafi Goldwasser, and Ioana Ivan. Functional signatures and pseudorandom functions. Cryptology ePrint Archive, Report 2013/401, 2013. http://eprint.iacr.org/2013/401.

[BK16] Rachid El Bansarkhani and Ali El Kaafarani. Post-quantum attribute-based signatures from lattice assumptions. Cryptology ePrint Archive, Report 2016/823, 2016. http://eprint.iacr.org/2016/823.

[BMS16] Michael Backes, Sebastian Meiser, and Dominique Schrder. Delegatable functional signatures. In *PKC*, pages 357–386, 2016.

[BV15] Zvika Brakerski and Vinod Vaikuntanathan. Constrained key-homomorphic prfs from standard lattice assumptions - or: How to secretly embed a circuit in your PRF. In *Theory of Cryptography - 12th Theory of Cryptography Conference, TCC 2015, Warsaw, Poland, March 23-25, 2015, Proceedings, Part II*, pages 1–30, 2015.

[BZ14] Dan Boneh and Mark Zhandry. Multiparty key exchange, efficient traitor tracing, and more from indistinguishability obfuscation. In *Crypto 2014*, pages 480–499, 2014.

[CFW14] Dario Catalano, Dario Fiore, and Bogdan Warinschi. Homomorphic signatures with efficient verification for polynomial functions. In *CRYPTO*, pages 371–389, 2014.

[CHKP12] David Cash, Dennis Hofheinz, Eike Kiltz, and Chris Peikert. Bonsai trees, or how to delegate a lattice basis. *J. Cryptology*, 25(4):601–639, 2012.

[CNW16] Shantian Cheng, Khoa Nguyen, and Huaxiong Wang. Policy-based signature scheme from lattices. In *Designs, Codes and Cryptography, Volume 81*, pages 43–74, 2016.

[CRV14] Nishanth Chandran, Srinivasan Raghuraman, and Dhinakaran Vinayagamurthy. Constrained pseudorandom functions: Verifiable and delegatable. 2014. http://eprint.iacr.org/2014/522.

[FMNP16] Dario Fiore, Aikaterini Mitrokotsa, Luca Nizzardo, and Elena Pagnin. Multi-key homomorphic authenticators. In *ASIACRYPT (2) 2016*, pages 499–530, 2016.

[Fuc14] Georg Fuchsbauer. Constrained verifiable random functions. In *SCN 2014*, pages 95–114, 2014.

[GPV08] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In Cynthia Dwork, editor, *Proceedings of the*

40th Annual ACM Symposium on Theory of Computing, Victoria, British Columbia, Canada, May 17-20, 2008, pages 197–206. ACM, 2008.

[GSW13] Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part I*, volume 8042 of *Lecture Notes in Computer Science*, pages 75–92. Springer, 2013.

[GVW13] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Attribute-based encryption for circuits. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*, pages 545–554. ACM, 2013.

[GVW15] Sergey Gorbunov, Vinod Vaikuntanathan, and Daniel Wichs. Leveled fully homomorphic signatures from standard lattices. In Rocco A. Servedio and Ronitt Rubinfeld, editors, *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015*, pages 469–477. ACM, 2015.

[LTWC16] Russell W. F. Lai, Raymond K. H. Tai, Harry W. H. Wong, and Sherman S. M. Chow. A zoo of homomorphic signatures: Multi-key and key-homomorphism. 2016. http://eprint.iacr.org/2016/834.

[Mic04] Daniele Micciancio. Almost perfect lattices, the covering radius problem, and applications to ajtais connection factor. In *SIAM J. Comput., 34(1)*, pages 118–169, 2004.

[MP12] Daniele Micciancio and Chris Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In *Advances in Cryptology - EUROCRYPT 2012 - 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cambridge, UK, April 15-19, 2012. Proceedings*, pages 700–718, 2012.

[MP13] Daniele Micciancio and Chris Peikert. Hardness of sis and lwe with small parameters. In *CRYPTO 2013*, pages 21–39, 2013.

[MPR11] Hemanta K. Maji, Manoj Prabhakaran, and Mike Rosulek. Attribute-based signatures. In *CT-RSA*, pages 376–392, 2011.

[MR07] Daniele Micciancio and Oded Regev. Worst-case to average-case reductions based on gaussian measures. In *SIAM J. Comput., 37(1)*, pages 267–302, 2007.

[Pei16] Chris Peikert. A decade of lattice cryptography. In *Foundations and Trends in Theoretical Computer Science 10*, pages 283–424, 2016.

[SAH16] Yusuke Sakai, Nuttapong Attrapadung, and Goichiro Hanaoka. Attribute-based signatures for circuits from bilinear map. In *Public-Key Cryptography PKC 2016*, pages 283–300, 2016.

A Definitions of Message-Policy CS

Definition A.1 ((Message-Policy) Constrained Signatures)**.** *Let $\mathcal{X}$ be an attribute space and $\mathcal{F}$ be a function space of the form $f \in \mathcal{F} \implies f : \mathcal{X}' \to \{0,1\}$ where $\mathcal{X}' \subseteq \mathcal{X}$. A constrained signatures scheme for $(\mathcal{X}, \mathcal{F})$ is a tuple of algorithms:*

- $\mathsf{Setup}(1^\lambda) \to (\mathsf{msk}, \mathsf{vk})$ *takes as input the security parameter λ and possibly a description of $(\mathcal{X}, \mathcal{F})$, and outputs a master signing key msk and a public verification key vk.*
- $\mathsf{Keygen}(x, \mathsf{msk}) \to \mathsf{sk}_x$ *takes as input an attribute $x \in \mathcal{X}$ and the master signing key msk, and outputs a signing key sk_x.*
- $\mathsf{Sign}(f, \mathsf{sk}_f) \to \sigma_f$ *takes as input a policy $f \in \mathcal{F}$ and a signing key sk_x, and outputs a signature σ_f.*
- $\mathsf{Ver}_{\mathsf{vk}}(f, \sigma_f) \to \{accept, reject\}$ *takes as input a policy $f \in \mathcal{F}$ and a signature σ_f, and either accepts or rejects.*

Correctness. *The scheme is correct if for all $x \in \mathcal{X}$ and $f \in \mathcal{F}$ for which $f(x) = 0$, it holds that with all but negligible probability $\mathsf{Ver}_{\mathsf{vk}}(f, \sigma_f) = accept$, where $(\mathsf{msk}, \mathsf{vk}) \leftarrow \mathsf{Setup}(1^\lambda)$ and $\sigma_f = \mathsf{Sign}(f, \mathsf{Keygen}(x, \mathsf{msk}))$.*

Definition A.2 (Privacy of (Message-Policy) Constrained Signatures)**.** *The scheme is* attribute-hiding *if any* PPT *adversary $\mathcal{A}$ has no more than negligible advantage in the following game.*

1. *The challenger computes and outputs $(\mathsf{msk}, \mathsf{vk}) \leftarrow \mathsf{Setup}(1^\lambda)$.*
2. *$\mathcal{A}$ sends (x_0, x_1, f) such that $f(x_0) = f(x_1) = 0$.*
3. *The challenger computes $\mathsf{sk}_{x_0} = \mathsf{Keygen}(x_0, \mathsf{msk})$ and $\mathsf{sk}_{x_1} = \mathsf{Keygen}(x_1, \mathsf{msk})$. It then samples $b \xleftarrow{\$} \{0,1\}$ and computes $\sigma_{f,b} \leftarrow \mathsf{Sign}(f, \mathsf{sk}_{x_b})$. It sends $\sigma_{f,b}$ to $\mathcal{A}$.*
4. *$\mathcal{A}$ outputs $b' \in \{0,1\}$ and wins if and only if $b' = b$.*

The scheme is key-hiding *if any* PPT *adversary $\mathcal{A}$ has no more than negligible advantage in the above game, where in step 2 the challenger sends $(\mathsf{sk}_{x_0}, \mathsf{sk}_{x_1}, \sigma_{f,b})$ to $\mathcal{A}$.*

Definition A.3 (Unforgeability of (Message-Policy) Constrained Signatures)**.** *The scheme is* fully unforgeable *if every PPTM adversary $\mathcal{A}$ has no more than negligible advantage in the following game:*

1. *The challenger computes $(\mathsf{msk}, \mathsf{vk}) \leftarrow \mathsf{Setup}(1^\lambda)$ and sends vk to $\mathcal{A}$.*
2. *$\mathcal{A}$ makes queries of three types:*
   - Key Queries. *$\mathcal{A}$ sends $x \in \mathcal{X}$ and gets back $\mathsf{sk}_x \leftarrow \mathsf{Keygen}(x, \mathsf{msk})$.*
   - Signature Queries. *$\mathcal{A}$ sends $(f, x) \in \mathcal{F} \times \mathcal{X}$ such that $f(x) = 0$. The challenger computes $\mathsf{sk}_x \leftarrow \mathsf{Keygen}(x, \mathsf{msk})$ and sends back $\sigma_f \leftarrow \mathsf{Sign}(f, \mathsf{sk}_x)$.*

- Repeated Signature Queries. *$\mathcal{A}$ sends $i \in \mathbb{N}$ and $f \in \mathcal{F} \times \mathcal{M}$. If there were less than i signature queries at this point of the game, the challenger returns $\perp$. Otherwise, let x denote the attribute that was sent at the ith signature query and let sk_x denote the key that was generated by the challenger when answering this query. If $f(x) \neq 0$, the challenger returns $\perp$. Otherwise it returns $\sigma_f \leftarrow \mathsf{Sign}(f, \mathsf{sk}_x)$.*

3. *$\mathcal{A}$ wins if it manages to output (f^*, σ_{f^*}) such that $\mathsf{Ver}_{\mathsf{vk}}(f^*, \sigma_{f^*}) = accept$ and the following restrictions hold:*

   - *For any key queried by $\mathcal{A}$ respective to $x \in \mathcal{X}$, it holds that $f^*(x) = 1$.*
   - *For any signature σ_f queried by $\mathcal{A}$, it holds that $f \neq f^*$.*

The scheme maintains message-selective *unforgeability if any PPT $\mathcal{A}$ that announces f^* before seeing vk has no more than negligible advantage in the game.*

Definition A.4 (Single-Key-Selective Unforgeability of (Message-Policy) Constrained Signatures)**.** *The scheme is single-key selectively unforgeable if every PPTM adversary $\mathcal{A}$ has no more than negligible advantage in the following game:*

1. *$\mathcal{A}$ sends $x^* \in \mathcal{F}$ to the challenger.*
2. *The challenger computes $(\mathsf{msk}, \mathsf{vk}) \leftarrow \mathsf{Setup}(1^\lambda)$ and $\mathsf{sk}_{x^*} \leftarrow \mathsf{Keygen}(x^*, \mathsf{msk})$, and sends $(\mathsf{vk}, \mathsf{sk}_{x^*})$ to $\mathcal{A}$.*
3. *$\mathcal{A}$ wins if it manages to output (f^*, σ_{f^*}) such that $\mathsf{Ver}_{\mathsf{vk}}(f^*, \sigma_{f^*}) = accept$ and $f^*(x^*) = 1$.*

B Proofs for Section 4.1

For any $t \geq 1$ and $F = (f_1, \ldots, f_t) \in \mathcal{F}^t$, write $F(x) = 0$ to denote that $f \in F \Rightarrow f(x) = 0$. Moreover, denote $\mathsf{sk}_F = \mathsf{sk}_{(f_1,\ldots,f_t)}$, where $\forall i \in [2 \ldots t] : \mathsf{sk}_{(f_1,\ldots,f_i)} = \mathsf{DelKey}(\mathsf{sk}_{(f_1,\ldots,f_{i-1})}, f_i)$ and $\mathsf{sk}_{f_1} = \mathsf{Keygen}(f_1, \mathsf{msk})$ for some $(\mathsf{msk}, \mathsf{vk}) \leftarrow \mathsf{Setup}(1^\lambda)$ which is clear from the context.

Lemma B.1 (Correctness)**.** *The scheme from section 4.1 is correct for $(\mathcal{F}', \mathcal{X}')$.*

Proof. Fix $x \in \mathcal{X}'$, $t \in \mathbb{N}$ and $F \in \mathcal{F}'^t$ such that $F(x) = 0$, and consider $(\mathsf{msk}, \mathsf{vk}) \leftarrow \mathsf{Setup}(1^\lambda)$. Consider sk_F as described above and $\sigma_x = \mathsf{Sign}(x, \mathsf{sk}_F)$. Denote

$$\sigma_x = (\mathsf{vk}', \sigma'_x, \{\mathsf{vk}''_i\}_{i\in[t]}, \sigma''_x, \sigma_{(\mathsf{vk}', \mathsf{vk}''_1)}, \{\sigma_{(\mathsf{vk}''_i, \mathsf{vk}''_{i+1})}\}_{i\in[t-1]}) \ ,$$

then by Sign, Keygen and DelKey it holds that $\sigma'_x = \mathsf{Sign}'(x, \mathsf{sk}'_F)$, and since $F(x) = 0$ it holds that $\mathsf{Ver}'_{\mathsf{vk}'}(\sigma'_x, x) = accept$ by the correctness of CS'. Moreover,

$$\mathsf{S.Ver}_{\mathsf{vk}''_t}(x, \sigma''_x) = \mathsf{S.Ver}_{\mathsf{vk}''_t}(x, \mathsf{S.Sign}(\mathsf{sk}''_t, x)) = accept \ ,$$

$$\mathsf{S.Ver}_{\mathsf{S.vk}}(\sigma_{(\mathsf{vk}', \mathsf{vk}''_1)}, (\mathsf{vk}', \mathsf{vk}''_1)) = \mathsf{S.Ver}_{\mathsf{S.vk}}(\mathsf{S.Sign}(\mathsf{S.sk}, (\mathsf{vk}', \mathsf{vk}''_1)), (\mathsf{vk}', \mathsf{vk}''_1)) = accept \ ,$$

and for all $i = 1 \ldots . t - 1$,

$$\mathsf{S.Ver}_{\mathsf{vk}''_i}(\sigma_{(\mathsf{vk}''_i, \mathsf{vk}''_{i+1})}, (\mathsf{vk}''_i, \mathsf{vk}''_{i+1})) = \mathsf{S.Ver}_{\mathsf{vk}''_i}(\mathsf{S.Sign}(\mathsf{sk}''_i, (\mathsf{vk}''_i, \mathsf{vk}''_{i+1})), (\mathsf{vk}''_i, \mathsf{vk}''_{i+1})) = accept \ .$$

by the correctness of S. Therefore, $\mathsf{Ver}_{\mathsf{vk}}(x, \sigma_x)$ accepts. □

Lemma B.2 (Privacy)**.** *The scheme from section 4.1 is constraint-hiding for* $(\mathcal{F}', \mathcal{X}')$.

Proof. Assume towards contradiction an adversary $\mathcal{A}$ that wins the constraint-hiding privacy game with some significant probability, and use it to break the constraint-hiding privacy of CS as follows:

1. Receive $(\mathsf{vk}', \mathsf{msk}') \leftarrow \mathsf{Setup}'(1^\lambda)$ from the CS challenger.
2. Compute $(\mathsf{S.vk}, \mathsf{S.sk}) \leftarrow \mathsf{S.Setup}(1^\lambda)$ and send $(\mathsf{msk} = \mathsf{S.sk}, \mathsf{vk} = \mathsf{S.vk})$ to $\mathcal{A}$.
3. $\mathcal{A}$ returns (t, F_0, F_1, x), where $\forall b \in \{0,1\} : F_b = (f_1^b, \ldots, f_t^b)$ and $F_b(x) = 0$. Forward (t, F_0, F_1, x) to the CS challenger.
4. The CS challenger samples $b \xleftarrow{\$} \{0,1\}$ and returns $\sigma'_{x,b}$.

   Now for $i \in [t]$ generate $(\mathsf{vk}''_1, \mathsf{sk}''_t) \leftarrow \mathsf{S.Setup}(1^\lambda)$, sign $(\mathsf{vk}', \mathsf{vk}''_1)$ with the standard signature scheme: $\sigma_{(\mathsf{vk}', \mathsf{vk}''_1)} \leftarrow \mathsf{S.Sign}(\mathsf{S.sk}, (\mathsf{vk}', \mathsf{vk}''_1))$ and for each $i \in [t-1]$ sign $\sigma_{(\mathsf{vk}''_i, \mathsf{vk}''_{i+1})} \leftarrow \mathsf{S.Sign}(\mathsf{S.sk}''_i, (\mathsf{vk}''_i, \mathsf{vk}''_{i+1}))$. Finally sign $\sigma''_x \leftarrow \mathsf{S.Sign}(\mathsf{sk}''_t, x)$ and send to $\mathcal{A}$ the signature $\sigma_{x,b} = (\mathsf{vk}', \sigma'_{x,b}, \{\mathsf{vk}''_i\}_{i\in[t]}, \sigma''_x, \sigma_{(\mathsf{vk}', \mathsf{vk}''_1)}, \{\sigma_{(\mathsf{vk}''_i, \mathsf{vk}''_{i+1})}\}_{i\in[t-1]})$.
5. Get b' from $\mathcal{A}$ and forward it to the CS challenger. Clearly, any advantage of $\mathcal{A}$ induces an advantage of the reduction.

□

Lemma B.3 (Unforgeability)**.** *The scheme from section 4.1 is fully unforgeable for* $(\mathcal{F}', \mathcal{X}')$.

Proof. Assume towards contradiction an adversary $\mathcal{A}$ that wins the security game. We show that it can be used to break either S or CS. Let $\mathcal{Q}_{key}, \mathcal{Q}_{sig}, \mathcal{Q}_{rep}$ be the sets of key queries, signature queries and repeated signature queries made by $\mathcal{A}$ during the security game. Recall that each query $q_i \in \mathcal{Q}_{key}$ is of the form $(t_i, (f_1^i, \ldots, f_{t_i}^i))$ and each query $q_i \in \mathcal{Q}_{sig}$ is of the form $(t_i, (f_1^i, \ldots, f_{t_i}^i, x_i))$, where $t_i \in \mathbb{Z}$, $f_j^i \in \mathcal{F}'$, $x_i \in \mathcal{X}'$. In particular, each query $q_i \in \mathcal{Q}_{key} \bigcup \mathcal{Q}_{sig}$ contains a set $(f_1^i, \ldots, f_{t_i}^i) \in \mathcal{F}'^{t_i}$. Moreover, every response of the challenger (whether it is a key or a signature) contains a tuple $(\mathsf{vk}'^i, \{\mathsf{vk}''^i_j\}_{j\in[t_i]})$ that is generated during Keygen and DelKey. $\mathcal{A}$ wins the game, it therefore outputs a successful forgery (x^*, σ_{x^*}), where $\sigma_{x^*} = (\mathsf{vk}'^*, \sigma'_{x^*}, \{\mathsf{vk}''^*_j\}_{j=1\ldots t}, \sigma''_{x^*}, \sigma_{(\mathsf{vk}'^*, \mathsf{vk}''^*_1)}, \{\sigma_{(\mathsf{vk}''^*_j, \mathsf{vk}''^*_{j+1})}\}_{j=1\ldots t-1})$. Consider three cases:

- If $\exists q_i \in \mathcal{Q}_{key}$ such that $(\mathsf{vk}'^i, \{\mathsf{vk}''^i_j\}_{j\in[t_i]}) = (\mathsf{vk}'^*, \{\mathsf{vk}''^*_j\}_{j\in[t_i]})$, then (x^*, σ'_{x^*}) is a valid forgery to the delegatable CS instance that was initialized during $\mathsf{Keygen}(f_i, \mathsf{msk})$. Note that since $q_i \in \mathcal{Q}_{key}$, $\exists j \in [1 \ldots t_i]$ such that $f_j^i(x^*) = 1$, therefore $(t_i, f_1^i, \ldots, f_{t_i}^i)$ is a valid delegated-key query to the underlying CS challenger. We show a reduction from the selective-single-key security game of CS to this game:
  1. Initialize $(\mathsf{S.vk}, \mathsf{S.sk}) \leftarrow \mathsf{S.Setup}(1^\lambda)$ as in the real scheme and send $\mathsf{S.vk}$ to $\mathcal{A}$.
  2. Queries phase:
     - Answer all queries except of the ith as in the real unforgeability game.
     - Upon receiving form $\mathcal{A}$ the query $q_i \in \mathcal{Q}_{key}$, send $(t_i, f_1^i, \ldots, f_{t_i}^i)$ to the ith CS challenger and get back $(\mathsf{vk}'_i, k'_{(f_1^i, \ldots, f_{t_i}^i)})$. For $j \in [t_i]$, generate $(\mathsf{vk}''^i_j, \mathsf{sk}''^i_j) \leftarrow \mathsf{S.Setup}(1^\lambda)$.

Compute $\sigma_{(\mathsf{vk}'_i,\mathsf{vk}''^i_1)} \leftarrow \mathsf{S.Sign}(\mathsf{S.sk},(\mathsf{vk}'_i,\mathsf{vk}''^i_1))$ and for each $j \in [1 \dots t_i - 1]$ compute $\sigma_{(\mathsf{vk}''^i_j,\mathsf{vk}''^i_{j+1})} \leftarrow \mathsf{S.Sign}(\mathsf{sk}''^i_j,(\mathsf{vk}''^i_j,\mathsf{vk}''^i_{j+1}))$. Send to $\mathcal{A}$ the key $\mathsf{sk}_{(f^i_1,\dots,f^i_{t_i})} = (\mathsf{vk}'_i, k'_{(f^i_1,\dots,f^i_{t_i})}, \{\mathsf{vk}''^i_j\}_{j\in[t_i]}, \mathsf{sk}''^i_{t_i}, \sigma_{(\mathsf{vk}'_i,\mathsf{vk}''^i_1)}, \{\sigma_{(\mathsf{vk}''^i_j,\mathsf{vk}''^i_{j+1})}\}_{j\in[t_i-1]})$.

3. When $\mathcal{A}$ sends the forgery (x^*,σ_{x^*}), send (x^*,σ'_{x^*}) to the ith CS challenger to win the selective-single-key game.

- If $\exists q_i \in \mathcal{Q}_{sig}$ such that $(\mathsf{vk}'^i,\{\mathsf{vk}''^i_j\}_{j\in[t_i]}) = (\mathsf{vk}'^*,\{\mathsf{vk}''^*_j\}_{j\in[t_i]})$, then (x^*,σ''_{x^*}) is a valid forgery to the S instance that was initialized during $\mathsf{DelKey}(k'_{(f^i_1,\dots,f^i_{t_i-1})}, f^i_{t_i})$. Note that $\forall q_i \in \mathcal{Q}_{sig}$, it holds that $x_i \neq x^*$. We show a reduction from the security game of $\mathcal{S}$ to this game:

  1. Initialize $(\mathsf{S.vk},\mathsf{S.sk}) \leftarrow \mathsf{S.Setup}(1^\lambda)$ as in the real scheme and send $\mathsf{S.vk}$ to $\mathcal{A}$.
  2. Queries phase:
     - Answer all queries up to q_i as in the real unforgeability game.
     - Upon receiving form $\mathcal{A}$ the query $q_i \in \mathcal{Q}_{sig}$, compute $k'_{(f^i_1,\dots,f^i_{t_i-1})}$ as in the real game, then instantiate a game against the S challenger and get $\mathsf{vk}''^i_{t_i}$. Query a signature for (x_i,m_i) and get back $\sigma''_{(x_i,m_i)}$. Sign $\sigma_{(\mathsf{vk}''^i_{t_i-1},\mathsf{vk}''^i_{t_i})} \leftarrow \mathsf{S.Sign}(\mathsf{sk}''^i_{t_i-1},(\mathsf{vk}''^i_{t_i-1},\mathsf{vk}''^i_{t_i}))$. Compute $k'_{(f^i_1,\dots,f^i_{t_i})} \leftarrow \mathsf{DelKey}'(k'_{(f^i_1,\dots,f^i_{t_i-1})}, f^i_{t_i})$ and $\sigma'_{(x_i,m_i)} \leftarrow \mathsf{Sign}'(x_i,m_i,k'_{(f^i_1,\dots,f^i_{t_i})})$. Send to $\mathcal{A}$: $\sigma_{x_i} = (\mathsf{vk}'_i,\sigma'_{x_i},\{\mathsf{vk}''^i_j\}_{j\in[t_i]},\sigma''_{(x_i,m_i)},\sigma_{(\mathsf{vk}'_i,\mathsf{vk}''^i_1)},\{\sigma_{(\mathsf{vk}''^i_j,\mathsf{vk}''^i_{j+1})}\}_{j\in[t_i-1]})$.
     - Answer all queries as in the real game, except of repeated signature queries that reference q_i. For these, do as described above with the values vk'_i, $\{\mathsf{vk}''^i_j\}_{j\in[t_i]}$, $\sigma_{(\mathsf{vk}'_i,\mathsf{vk}''^i_1)}$, $\{\sigma_{(\mathsf{vk}''^i_j,\mathsf{vk}''^i_{j+1})}\}_{j\in[t_i-1]}$, $k'_{(f^i_1,\dots,f^i_{t_i})}$ that were generated when q_i was answered.
  3. When $\mathcal{A}$ sends the forgery (x^*,σ_{x^*}), send (x^*,σ''_{x^*}) to the ith S challenger to win the game.

- If $\forall d \in [1 \dots t_i]$ and $\forall q_i \in \mathcal{Q}^*_d = \{q_i \in \mathcal{Q}_{key} \bigcup \mathcal{Q}_{sig} : (\mathsf{vk}'_i,\{\mathsf{vk}''^i_j\}_{j=1\dots d-1}) = (\mathsf{vk}'_*,\{\mathsf{vk}''^*_j\}_{j=1\dots d-1})\}$ it holds that $\mathsf{vk}''^*_d \neq \mathsf{vk}''^i_d$, then $(\sigma_{(\mathsf{vk}''^*_{d-1},\mathsf{vk}''^*_d)},((\mathsf{vk}''^*_{d-1},\mathsf{vk}''^*_d))$ is a valid forgery to the S instance with the verification key $\mathsf{vk}''^*_{d-1} = \mathsf{vk}''^i_{d-1}$. The reduction follows similar lines to the reduction from the previous case.

- Otherwise $\forall q_i \in \mathcal{Q}_{key} \bigcup \mathcal{Q}_{sig}$ $(\mathsf{vk}'_i,\mathsf{vk}''^i_1) \neq (\mathsf{vk}'_*,\mathsf{vk}''^*_1)$, and thus $(\sigma_{(\mathsf{vk}'_*,\mathsf{vk}''^*_1)},(\mathsf{vk}'_*,\mathsf{vk}''^*_1))$ is a valid forgery to S. We show a reduction from the security game of S to this game:

  1. Receive $\mathsf{S.vk}$ from the S challenger and send it to $\mathcal{A}$.
  2. Answer queries from $\mathcal{A}$ as in the real game, except the way $\sigma_{(\mathsf{vk}'_*,\mathsf{vk}''^*_1)}$ is computed: instead of signing $(\mathsf{vk}'_*,\mathsf{vk}''^*_1)$ with $\mathsf{msk} = \mathsf{S.sk}$, query the S challenger and get $\sigma_{(\mathsf{vk}'_*,\mathsf{vk}''^*_1)}$.
  3. When $\mathcal{A}$ sends the forgery (x^*,σ_{x^*}), send $(\sigma_{(\mathsf{vk}'_*,\mathsf{vk}''^*_1)},(\mathsf{vk}'_*,\mathsf{vk}''^*_1))$ to the S challenger to win the game.

□