International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:14, No:7, 2020

A Watermarking Signature Scheme with Hidden Watermarks and Constraint Functions in the Symmetric Key Setting

Yanmin Zhao, Siu Ming Yiu

***Abstract*—**To claim the ownership for an executable program is a non-trivial task. An emerging direction is to add a watermark to the program such that the watermarked program preserves the original program's functionality and removing the watermark would heavily destroy the functionality of the watermarked program. In this paper, the first watermarking signature scheme with the watermark and the constraint function hidden in the symmetric key setting is constructed. The scheme uses well-known techniques of lattice trapdoors and a lattice evaluation. The watermarking signature scheme is unforgeable under the Short Integer Solution (SIS) assumption and satisfies other security requirements such as the unremovability security property.

***Keywords*—**Short integer solution problem, signatures, the symmetric-key setting, watermarking schemes.

I. Introduction

THE notion of watermarking is influential for digital rights management, such as tracing information leaks or resolving ownership disputes. There are numerous works, for example [1], [2], on watermarking pictures, music files, movies, or other perceptual objects. On the other hand, it is not trivial how to watermark an executable program. The difficulty was formally illustrated in [3], [4]. They showed that no watermarking scheme can be constructed for any program such that the watermarked program behaves *identically* to the original program even assuming the existence of a powerful indistinguishable obfuscator (the IO assumption). But if we can sacrifice the *perfect correctness*, [5] showed that assuming the existence of an indistinguishable obfuscator, watermarking pseudorandom functions, signatures and encryption schemes is possible if we allow the watermarked programs behave differently from the original programs with a negligible fraction of the whole domain.

Based on Cohen et al.'s work [5] and other previous studies [9]-[12], we we can have three security requirements for a watermarked program. The first one is correctness and function-preserving, which requires that the watermarked program must behave *almost identical* to the original one except for a few cases with negligible probability, which should be unnoticeable to the users. The second one is unremovability. Once a program is watermarked, an adversary cannot easily remove the watermark while keeping the correctness and function-preserving property[1]. The third one is unforgeability, which means that an adversary cannot forge a valid watermarked program which behaves significantly different from any original watermarked programs he/she is able to access to.

Most of existing works for watermarking a program were done for pseudorandom functions (PRFs) [5], [9]-[12]. Only limited results exist on how to watermark other cryptographic schemes such as signatures and encryptions. In particular for signatures[2], there are only two results. Reference [5] proved the existence of a watermarking scheme, but the scheme is based on the IO assumption and no concrete construction was given. The only concrete scheme we can find for signatures is [14]. They constructed a watermarking scheme for constrained signature schemes (i.e., these signing schemes only sign correctly for messages satisfying certain constraints and cannot output a correct signature for other messages), but the watermark needs to be exposed. Both results allow public extraction of the watermark.

"Public" extraction and "exposed" watermark are useful in certain application scenarios such as when a client tries to purchase a software (program), it is easy for him to check if the copyright owner is the seller or an authorized agent to sell him the software. On the other hand, there are also applications that we prefer having "private" extraction and "hidden" watermark (i.e., no user except the signer can tell if the software has been watermarked). For example, hidden watermark can help to trace and identify the copyright infringement chain without being noticed by the suspects and in some cases, signers may want to embed private information into the watermark for which they do not want to reveal unless it is really necessary such as in a court case to prove the ownership of the program. However, to the best of our knowledge, currently there is no result in the symmetric key setting [3].

In this paper, we consider watermarking a signature scheme and propose the first watermarking signature scheme in the symmetric key setting with a hidden watermark and a hidden

Yanmin Zhao is with The University of Hong Kong, China (e-mail: ymzhao@cs.hku.hk).

[1] Note that in some earlier works [6]-[8], the unremovability property is only guaranteed when an adversary is restricted to certain modification requirements.

[2] For encryption, there are some results. For example, [13] gave constructions for watermarking public-key encryption with unremovability and unforgeability security properties.

[3] We also remark that this "watermarking a program" is an important and emerging direction. Although no concrete and real applications yet, we believe that real applications will follow after the research community has come up with effective schemes for different settings.

constraint function. Our scheme works for any constrained signature schemes. We prove that our watermarking scheme is secure against chosen programs attack in which the adversary can access the watermarking oracle while the watermarked signature scheme is unforgeable.

Technique overview: Our scheme is under the symmetric key setting, in which the key used to watermark the signature scheme will be the key for extracting the watermark from the scheme, i.e., only the owner who has the correct key can show the watermark to prove its ownership.

For all we know, there is no previous scheme for watermarking a signature scheme under the symmetry key setting. It is easy to transform the scheme in [14] from "public" extraction to "private" extraction as follows. To hide the embedded message (which means that the embedded message cannot be obtained directly from the signature), we first encrypt the embedded message and then encode the resulting ciphertext into the signing key. This construction can enable the private extraction. But on the other hand, the "exposed" message still provides hints to everyone that the program has been watermarked.

Technically, in order to hide both the message and also the constraint function, we first borrow the idea of the message-embedding method in [9] to create and embed our watermark message into the signature scheme. The underlying idea is as follows. To encode a message msg, we first generate a set of points $\{x_i^0, x_i^1\}_{i\in[t]}$ where t is the length of the embedded message msg. The subset $\{x_i^{msg_i}\}_{i\in[t]}$, which is set of the messages used to encode the embedded message, is set of the messages that cannot be correctly signed by a constrained signing key. If the set $\{x_i^0, x_i^1\}_{i\in[t]}$ is shared by the watermarking procedure and the extraction procedure, then for an embedded message msg, the watermarking procedure can change the signatures for messages belonging to the subset $\{x_i^{msg_i}\}_{i\in[t]}$ to encode the message msg into the signature scheme. This explains why both this paper and [14] choose a constrained signature scheme as an underlying component to construct a watermarking signature scheme. And the extraction procedure can test every pair of $\{x_i^0, x_i^1\}$. If the signature for x_i^0 is not correct, then $msg_i = 0$; otherwise, $msg_i = 1$.

Once we know how to encode a message, another difficulty is to hide the point set. Otherwise, the adversary can just change the signatures for the messages in the point set to remove the embedded message. The difficulty is that the hidden points must be found later when extracting the embedded message. Our solution is to randomize the set $\{x_i^{msg_i}\}_{i\in[t]}$ by multiplying a secret matrix **B** and a public matrix **A** and define the new set $\{\mathbf{A}\mathbf{B}x_i^{msg_i}\}_{i\in[t]}$ as part of the public verification key. We also remark that we cannot use the same point set for two different signature schemes as explained in the following. Since the point set used for encoding the embedded message must be linked to the secret signing key, the point set must be unique for every signing key. Suppose that we use the same point set encoding messages for signature schemes Sig_1 and Sig_2. If some points in the set are determined by noticing incorrect signatures for a watermarked signature scheme Sig_1, then the embedded message cannot be extracted correctly for the signature scheme Sig_2 since the adversary knows which message's signature should be changed.

For how we make use of these techniques in the concrete construction, please refer to Section IV. The rest of the paper is organized as follows: Section II provides some technical background knowledge. Section III talks about the constrained signature schemes. Section V gives the formal proofs of the security of the proposed scheme and Section VI concludes the paper.

Remarks: Another formal definition for watermarking schemes can be found in [15]. They formulated the rigorous and general security definition for watermarking schemes. Their security definition implies other former definitions, while also no concrete constructions are given.

Roughly speaking, our proposed definition is for the case of "private" marking and "private" extraction in a symmetric-key setting. All previous definitions are defined for the public-key setting, either with "public" marking or "public" extraction. More precisely, in [5], Cohen et al. define a watermarking scheme for a signature scheme by generating a public extraction key xk and a secret marking key mk. In other words, any signature scheme is watermarked by a marker who possesses the secret marking key while anyone else can extract the embedded mark or message. Afterwards, Goyal et al. define a different watermarking scheme for signature scheme. In their work [14], a signature scheme can be marked by anyone (public marking) and the embedded mark or message can be extracted from the watermarked signature circuit by anyone (public extraction). In their extensions, they briefly state how to construct a secret marking and public extraction scheme. On the other hand, our watermarking scheme definition is different. Ours only permits the person who is aware of the secret marking or extraction key can execute the marking or extraction algorithm.

Existing works focus on watermarking cryptographic programs. A major reason is discussed in [5], in which they showed that if a program is *learnable*, i.e., it is possible for an adversary to consider the watermarked program as a virtual black box and recover the description of the original program, then it is possible for the adversary to construct a program with the same functionality without the watermark (for details, please refer to [5]), then it is not watermarkable. Most of the cryptographic programs are not learnable while many other programs belong to the category of learnable programs, thus researchers focus on how to watermark a cryptographic program.

II. Preliminaries

A. Background on Lattices and the SIS Problem

Notation. For any integer $q \geq 2$, the ring of integers modulo q is denoted by $\mathbb{Z}_q$. Elements of $\mathbb{Z}_q$ are represented in the range $(-q/2, q/2]$ and take their representatives in the range as absolute values. For a vector $\mathbf{v} \in \mathbb{Z}_q^n$, $\| \mathbf{v} \|_\infty \leq \beta$ if each component $v_i \leq \beta$. For a matrix $\mathbf{V} \in \mathbb{Z}_q^{n\times m}$, $\| \mathbf{V} \|_\infty \leq \beta$ if each

entry $V_{i,j} \leq \beta$.

The SIS Problem. Let n, m, q, B be integer parameters. In the SIS(n, m, q, B) problem, given a uniformly random matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$, the attacker targets to find a vector $\mathbf{u} \in \mathbb{Z}_q^m$ with $\mathbf{u} \neq 0$ and $\| \mathbf{u} \|_\infty \leq B$[4] such that $\mathbf{A} \cdot \mathbf{u} = 0$. For parameters $n = n(\lambda), m = m(\lambda), q = q(\lambda), B = B(\lambda)$ defined in terms of the security parameter λ, the hardness assumption of the SIS(n, m, q, B) problem states that for any PPT (probabilistic polynomial time) attacker $\mathcal{A}$,

$$\Pr[\mathbf{A} \cdot \mathbf{u} = 0 \wedge \| \mathbf{u} \|_\infty \leq B \wedge \mathbf{u} \neq 0 : \mathbf{A} \leftarrow \mathbb{Z}_q^{n\times m}, \mathbf{u} \leftarrow \mathcal{A}(1^\lambda, \mathbf{A})] \leq \mathrm{negl}(\lambda).$$

Theorem 1 [16]-[20]. For any $m = \mathrm{poly}(n)$, $B > 0$ and sufficiently large $q > B \times \mathrm{poly}(n)$, solving $\mathrm{SIS}_{n,m,q,B}$ with non-negligible possibility 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 any n-dimensional lattice with overwhelming possibility, for some $\gamma = B \times \mathrm{poly}(n)$.

Lattice Trapdoors. Let $n, q \in \mathbb{Z}$, $\mathrm{g} = (1,2,4,\ldots,2^{\lceil \log q \rceil - 1}) \in \mathbb{Z}_q^{\lceil \log q \rceil}$ and $m = n\lceil \log q \rceil$. The gadget matrix $\mathbf{G} = \mathbf{g} \otimes \mathbf{I}_n$, where $\otimes$ is the tensor product of two matrices. For any $t \in \mathbb{Z}$, define $\mathbf{G}^{-1}: \mathbb{Z}^{n\times t} \rightarrow \{0,1\}^{m\times t}$ mapping each entry of the input matrix into a column vector which is the bit decomposition of the corresponding entry. For any $\mathbf{A} \in \mathbb{Z}^{n\times t}$, it holds that $\mathbf{G} \cdot \mathbf{G}^{-1}(\mathbf{A}) = \mathbf{A}$.

As stated in Theorem 1, the $\mathrm{SIS}_{n,m,q,B}$ problem is hard. However, with some trapdoor, solving the $\mathrm{SIS}_{n,m,q,B}$ problem is tractable for any random matrix $\mathbf{A}$. Moreover, $\mathrm{SIS}_{n,m,q,B}$ problem is easy to solve for a gadget matrix $\mathbf{G}$.

Theorem 2 (Trapdoor Generation [16], [21]). Given integers $n \geq 1$, $q \geq 2$, and $m_0 = O(n \log q)$, there exists an efficient algorithm $(\mathbf{A}, \mathrm{T_A}) \leftarrow \mathrm{TrapGen}(1^n, 1^m, q)$, where $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ is 2^{-n}-uniform for all $m \geq m_0$. Meanwhile, there exists an efficient algorithm $\mathbf{u} \leftarrow \mathrm{SamplePre}(\mathbf{A}, \mathrm{T_A}, \mathbf{v})$ such that $\mathbf{Au} = \mathbf{v}$ and $\mathbf{u}$ follows Discrete Gaussian distribution $\mathcal{D}_{\mathbb{Z}^m, \tau_0}$ with $\tau_0 = O(\sqrt{n \log q \log n})$.

Theorem 3 (Trapdoor Extension [21], [22]). Given $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ with a trapdoor $\mathrm{T_A}$, if there exist two matrices $\mathbf{B} \in \mathbb{Z}_q^{n\times m'}$ and $\mathbf{S} \in \mathbb{Z}^{m'\times m}$ with the largest singular value $s_1(\mathrm{S})$ such that $\mathbf{A} = \mathbf{BS} (\mathrm{mod} q)$, then $\mathrm{T_A}$ and $\mathbf{S}$ can be used to generate a trapdoor $\mathrm{T_B}$ for the matrix $\mathbf{B}$. If the output of $\mathrm{SamplePre}(\mathbf{A}, \mathrm{T_A}, \mathbf{v})$ follows a distribution that is not far-from the Discrete Gaussian distribution $\mathcal{D}_{\mathbb{Z}^m, \tau}$, then the output $\mathrm{SamplePre}(\mathbf{B}, \mathrm{T_B}, \mathbf{v})$ follows a distribution that is not far-from the Discrete Gaussian distribution $\mathcal{D}_{\mathbb{Z}^{m'}, \tau'}$, in which $\tau' \geq \tau' s_1(\mathbf{S})$.

Corollary 1. As stated in Theorem 3, set $\mathbf{S} = [\mathrm{I}_m | 0_{m'}]^T$ specifically. Given $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ with a trapdoor $\mathrm{T_A}$, for any $\mathbf{B} \in \mathbb{Z}^{n\times m'}$, there exists an efficient algorithm $([\mathbf{A}|\mathbf{B}], \mathrm{T}_{[\mathbf{A}|\mathbf{B}]}) \leftarrow \mathrm{ExtensionLeft}(\mathbf{A}, \mathrm{T_A})$. If the output of $\mathrm{SamplePre}(\mathbf{A}, \mathrm{T_A}, \mathbf{v})$ follows a distribution that is not far-from the Discrete Gaussian distribution $\mathcal{D}_{\mathbb{Z}^m, \tau}$, then $\mathbf{u} \leftarrow \mathrm{SamplePre}([\mathbf{A}|\mathbf{B}], \mathrm{T}_{[\mathbf{A}|\mathbf{B}]}, \mathbf{v})$ follows a distribution that is not far-from the Discrete Gaussian distribution $\mathcal{D}_{\mathbb{Z}^{m'+m}, \tau'}$, and $\tau' > \tau$.

Before proceeding to the following corollary, we recall that the trapdoor for a gadget matrix G is trivial.

Corollary 2. As stated in Theorem 3, set $\mathbf{S} = [-\mathbf{R}^T | \mathbf{I}_m]^T$ specifically. Given matrices $\mathbf{A} \in \mathbb{Z}_q^{n\times m'}$ and $\mathbf{R} \in \mathbb{Z}^{m'\times m}$, there exists an efficient algorithm $(\mathbf{M}, \mathrm{T_M}) \leftarrow \mathrm{ExtensionRight}(\mathbf{A}, \mathbf{R}, \mathbf{G})$ where $\mathbf{M} = [\mathbf{A}|\mathbf{AR}+\mathbf{G}]$. And $\mathbf{u} \leftarrow \mathrm{SamplePre}(\mathbf{M}, \mathrm{T_M}, \mathbf{v})$ follows a distribution that is not far-from the Discrete Gaussian distribution $\mathcal{D}_{\mathbb{Z}^{m'+m}, \tau}$ for $\tau = O(\sqrt{mm'} \| \mathbf{R} \|_\infty)$.

In particular, $\mathrm{ExtensionRight}(\mathbf{A}, \mathbf{R}, \mathbf{G})$ and $\mathrm{SamplePre}([\mathbf{A}|\mathbf{AR}+\mathrm{G}], \mathrm{T}_{[\mathbf{A}|\mathbf{AR}+\mathbf{G}]}, \mathbf{v})$ are renamed as $\mathrm{SampleRight}(\mathbf{A}, \mathbf{R}, \mathbf{v})$ in the following.

Discrete Gaussian Distribution. Let $q \geq 2$ be a prime, $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ and $\mathbf{u} \in \mathbb{Z}_q^n$. Define $\Lambda_q^{\mathbf{u}}(\mathbf{A}) = \{\mathbf{e} \in \mathbb{Z}^m, s.t.\ \mathbf{Ae} = \mathbf{u} (\mathrm{mod} q)\}$. Suppose $\mathbf{S}^{n\times m}$ is any matrix, and $\tilde{\mathbf{S}}$ is the Gram-Schmidt orthogonalization of $\mathbf{S}$, define $\| \tilde{\mathbf{S}} \|_2 = \max\{\| \tilde{\mathbf{s}}_1 \|_2, \ldots, \| \tilde{\mathbf{s}}_m \|_2\}$.

For any vector $\mathbf{x} \in \mathbb{Z}^n$ and any $s > 0$, a Gaussian function scaled by a factor s is defined by

$$\rho_s(\mathbf{x}) = \exp(-\pi \| \frac{\mathbf{x}}{s} \|^2).$$

For any countable set A and a parameter $s > 0$, the Discrete Gaussian probability distribution $\mathcal{D}_{A,s}(\mathbf{x})$ is defined as

$$\mathcal{D}_{A,s}(\mathbf{x}) = \frac{\rho_s(\mathbf{x})}{\rho_s(A)} = \frac{\rho_s(\mathbf{x})}{\sum_{\mathbf{y}\in A} \rho_s(\mathbf{y})}.$$

Lemma 1 [18], [23]. Assume that $q \geq 2$ and $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ with $m > n$. Let $\mathrm{T_A}$ is a basis for $\Lambda_q^\perp(\mathbf{A})$ and $s \geq \| \widetilde{\mathrm{T_A}} \| \omega(\sqrt{\log m})$. Then for $\mathbf{u} \in \mathbb{R}^n$:

$$\Pr[\mathbf{x} \sim \mathcal{D}_{\Lambda_q^{\mathbf{u}}(\mathbf{A}),s} : \| \mathbf{x} \| \geq \sqrt{m} s] \leq \mathrm{negl}(n).$$

Lattice Evaluation. For the following, we cite a theorem from [20] directly. The evaluation procedure has been studied in many works [21], [22], [24]-[27] relating to LWE-based FHE and ABE. The description from [20] is similar to one in [28], [29].

Theorem 4 (Theorem 3 [20]). For any $n, q, t \in \mathbb{Z}$ and $m = n\lceil \log q \rceil$, let $\vec{\mathbf{A}} \in \mathbb{Z}_q^{n\times mt}$, f be a d-depth Boolean circuit $f: \{0,1\}^t \rightarrow \{0,1\}$, and $\mathbf{x} = (x_1, x_2, \ldots, x_t) \in \{0,1\}$. There exist two efficient algorithms EvalF and EvalFX s.t. $\mathbf{H}_f = \mathrm{EvalF}(f, \vec{\mathbf{A}})$ and $\mathbf{H}_{f,x} = \mathrm{EvalFX}(f, \mathbf{x}, \vec{\mathbf{A}})$ where $\mathbf{H}_f, \mathbf{H}_{f,x} \in \mathbb{Z}^{tm\times m}$. It holds that $\| \mathbf{H}_f \|_\infty, \| \mathbf{H}_{f,x} \|_\infty \leq (2m)^d$ and $(\vec{\mathbf{A}} - \mathbf{x} \otimes \mathbf{G}) \cdot \mathbf{H}_{f,x} = \vec{\mathbf{A}} \mathbf{H}_f - f(\mathbf{x}) \mathbf{G} (\mathrm{mod} q)$.

Lemma 2 [30]. Suppose that $m > (n+1)\log q + \omega(\log n)$ and that $q > 2$ is prime. Let $\mathbf{R}$ be an $m \times k$ matrix chosen uniformly in $\{-1,1\}^{m\times k} (\mathrm{mod} q)$ where $k = k(n)$ is

[4] Usually, the SIS Problem is stated with ℓ_2 norm rather than ℓ_∞ norm. The two statements are equivalent up to some small losses of parameters. ℓ_∞ norm is chosen for simplicity.

polynomial in n. Let **A** and **B** be matrices chosen uniformly in $\mathbb{Z}_q^{n\times m}$ and $\mathbb{Z}_q^{n\times k}$ respectively. Then, for all vectors $\boldsymbol{w}$ in $\mathbb{Z}_q^m$, the distribution $(\mathbf{A}, \mathbf{AR}, \mathbf{R}^T\boldsymbol{w})$ is statistically close to the distribution $(\mathbf{A}, \mathbf{B}, \mathbf{R}^T\boldsymbol{w})$.

III. Constrained Signature Schemes

Definition 1 (Constrained Signatures [20]). A constrained signature scheme consists of four algorithms $\Pi_{CS} = ($ CS.Setup, CS.KeyGen, CS.Sign, CS.Verify) over an attribute space $\mathcal{X}$, a signature space $\mathcal{Y}$, and a circuit class $\mathcal{C}$ is defined as:

- CS.Setup$(1^\lambda) \to (vk, msk)$. CS.Setup is the initialization algorithm that on input the security parameter λ, output a master secret key msk and a verification key vk.
- CS.KeyGen$(msk, f) \to sk_f$. On input the master secret key msk and the constraint circuit $f \in \mathcal{C}$, the CS.KeyGen$(\cdot)$ outputs a constrained key sk_f.
- CS.Sign$(x, sk_f) \to \sigma_x$. On input the attribute $x \in \mathcal{X}$ and a constrained key sk_f, the CS.Sign$(\cdot)$ outputs a signature $\sigma_x \in \mathcal{Y}$. The signature is correct if and only if $f(x) = 0$.
- CS.Verify$(x, \sigma_x, vk) \to \{0,1\}$. On input the attribute x, the signature σ_x and the verification key vk, CS.Verify$(\cdot)$ outputs 1 for accepting the signature or 0 for rejecting the signature.

The constrained signature scheme satisfies requirements for correctness, unforgeability and privacy which are elaborated as follows.

Correctness. The scheme is correct if for all $x \in \mathcal{X}$ and $f \in \mathcal{C}$ with $f(x) = 0$, the following holds:

$$\Pr\left[\begin{array}{c}\text{CS.Verify}(x, \sigma_x, vk) \\ = 0\end{array}\middle|\begin{array}{c}(vk, msk) \leftarrow \text{CS.Setup}(1^\lambda) \\ sk_f \leftarrow \text{CS.KeyGen}(msk, f) \\ \sigma_x \leftarrow \text{CS.Sign}(x, sk_f)\end{array}\right] \leq \text{negl}(\lambda).$$

Unforgeability. We describe a message-selective unforgeability experiment in which an adversary sends a message before seeing any public parameters. In the experiment, an adversary can make three kinds of queries: 1. a query for a constrained key; 2. a query for a signature under a specific constraint; 3. a query for a signature that is signed under an existing key.

We define a security game between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$ who can be described as a probabilistic polynomial time machine (PPTM) as follows:

- The adversary $\mathcal{A}$ sends x^* that is the message for which it intends to forge a signature to the challenger $\mathcal{C}$;
- The challenger $\mathcal{C}$ runs CS.Setup(1^λ) to obtain (msk, vk) and sends vk to the adversary $\mathcal{A}$;
- Query phase:
  - Key Queries. The adversary $\mathcal{A}$ sends f to the challenger, and receives $sk_f \leftarrow \text{CS.KeyGen}(msk, f)$.
  - Signature Queries. The adversary $\mathcal{A}$ sends (f, x) with $f(x) = 0$ to the challenger. CS.KeyGen(msk, f) is invoked by the challenger to get sk_f. Finally, the challenger returns $\sigma_x \leftarrow \text{CS.Sign}(x, sk_f)$.
  - Repeated Signature Queries. Adversary $\mathcal{A}$ sends (i, x) to the challenger. If there are less than i signature queries, then the challenger returns $\perp$. Assume the constraint in the i-th query is f and the corresponding constrained key is sk_f. If $f(x) = 1$, then the challenger returns $\perp$. Otherwise, the challenger returns $\sigma_x \leftarrow \text{CS.Sign}(x, sk_f)$
- $\mathcal{A}$ outputs a (x^*, σ_{x^*}) such that CS.Verify$(vk, x^*, \sigma_{x^*}) = 1$. For all $f \in \mathcal{C}$ queried by $\mathcal{A}$, $f(x^*) = 1$ and $\mathcal{A}$ never asks a signature for x^*. If all requirements are met, then $\mathcal{A}$ wins the experiment.

A constrained signature scheme is message-selectively unforgeable if Pr[$\mathcal{A}$ wins]≤negl(λ).

Privacy. Privacy guarantees the security of the signing key after releasing the signed signatures. In a constrained signature scheme, the privacy guarantees that the constraint function f is hidden.

Firstly, define a privacy game between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$ as follows:

- The challenger $\mathcal{C}$ runs $(vk, msk) \leftarrow \text{CS.Setup}(1^\lambda)$ and sends vk to the adversary $\mathcal{A}$;
- $\mathcal{A}$ sends (f_0, f_1, x) such that $f_0(x) = f_1(x) = 0$;
- The challenger computes $sk_0 = \text{CS.KeyGen}(msk, f_0)$ and $sk_1 = \text{CS.KeyGen}(msk, f_1)$. Then, it samples uniformly at random $b \leftarrow \{0,1\}$ and computes $\sigma_{x,b} \leftarrow \text{CS.Sign}(x, sk_b)$. Finally, it sends $(sk_0, sk_1, \sigma_{x,b})$ to $\mathcal{A}$;
- $\mathcal{A}$ outputs a bit b' and wins the game if $b = b'$.

A constrained signature scheme is strongly-hiding if Pr[$\mathcal{A}$ wins]≤negl(λ).

In [20], Tsabary constructs a constrained signature scheme from lattice trapdoors. The security of his constrained signature scheme is based on the SIS assumption. Our watermarking signature scheme takes this constrained signature scheme as an underlying component. For completeness, we describe Tsabary's scheme here.

The initialization parameters are t, d. The attribute space is $\{0,1\}^t$ and the constraint space is all d-depth circuits $\mathcal{F}_d = \{f: \{0,1\}^t \to \{0,1\}\}$. Define the constrained signature scheme $\Pi_{CS} =$ (CS.Setup, CS.KeyGen, CS.Sign, CS.Verify) as:

- CS.Setup$(1^\lambda) \to (msk, vk)$. Run $(\mathbf{A} \in \mathbb{Z}_q^{n\times m'}, \mathrm{T}_\mathbf{A}) \leftarrow \text{TrapGen}(1^\lambda)$; sample uniformly a matrix $\vec{\mathbf{A}} \xleftarrow{\$} \mathbb{Z}_q^{n\times(m\times t)}$. Output $msk = \mathrm{T}_\mathbf{A}$ and $vk = (\mathbf{A}, \vec{\mathbf{A}})$.
- CS.KeyGen$(msk, f) \to sk_f$. Compute $\mathbf{H}_f = \text{EvalF}(f, \vec{\mathbf{A}})$ and $\mathbf{A}_f = \vec{\mathbf{A}} \cdot \mathbf{H}_f$. Then, obtain $sk_f = \mathrm{T}_{\mathbf{A}||\mathbf{A}_f} \leftarrow \text{ExtensionLeft}(\mathbf{A}, \mathrm{T}_\mathbf{A})$.
- CS.Sign$(x, sk_f) \to \sigma_x$. If $f(x) \neq 0$, then return $\perp$. Otherwise, generate a trapdoor T for $[\mathbf{A}||\vec{\mathbf{A}} - \boldsymbol{x} \otimes \mathbf{G}]$ using the constraint secret signing key sk_f. Then, run $\sigma_x \leftarrow \text{SamplePre}([\mathbf{A}||\vec{\mathbf{A}} - \boldsymbol{x} \otimes \mathbf{G}], \mathrm{T}, 0)$ to generate a signature.
- CS.Verify$(x, \sigma_x, vk) \to \{0,1\}$. Output 1 if $\sigma_x \neq 0$, $\sigma_x \neq \perp$, $[\mathbf{A}||\vec{\mathbf{A}} - \boldsymbol{x} \otimes \mathbf{G}] \cdot \sigma_x = 0$ and $\| \sigma_x \|_\infty < B$. Otherwise, output 0.

An explanation for CS.Sign$(\cdot)$ procedure. Compute

$\mathbf{H}_{f,x} = \text{EvalFX}(f, x, \vec{\mathbf{A}})$. According to Theorem 4, $[\vec{\mathbf{A}} - x \otimes \mathbf{G}]\mathbf{H}_{f,x} = \mathbf{A}_f - f(x)\mathbf{G} = \mathbf{A}_f$ if $f(x) = 0$. For the following matrices:

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

It holds that $\bar{\mathbf{A}} = \bar{\mathbf{B}}\mathbf{S}$. According to Theorem 3, there exists an algorithm generating a trapdoor $\mathrm{T}_{\bar{\mathrm{B}}}$ with knowledge of sk_f. Thus, it is practical to invoke SamplePre(·) to compute a signature for x.

Theorem 5 (Correctness, privacy, and unforgeability [20]). The construction is correct, statistically key-hiding and message-selectively unforgeable for $(\mathcal{X}, \mathcal{F})$.

IV. A Watermarking Scheme for Constrained Signatures

Definition 4. A watermarking signature scheme is a tuple of probabilistic polynomial time algorithms (WM.Gen, SIG.Gen^0, SIG.Gen, Sign, Verify, Extract) which is correct, existentially unforgeable under the chosen message attack, and ϵ-unremovability.

- WM.Gen$(1^\lambda) \to ek$. On input the security parameter λ, WM.Gen(·) outputs an extraction key ek.
- SIG.Gen$^0(1^\lambda) \to (vk^0, sk^0)$. On input the security parameter λ, SIG.Gen^0 outputs a verification key vk^0 and a secret key sk^0.
- SIG.Gen$(1^\lambda, ek, msg, vk^0, sk^0) \to (vk, sk)$. On input the security parameter λ, an extraction key ek, and an embedded message msg, SIG.Gen(·) outputs a pair of a verification key vk and a secret key sk.
- Sign$(vk, sk, x) \to \Sigma_x$. On input a verification key vk, a secret key sk and a message x, Sign(·) outputs a signature Σ_x.
- Verify$(vk, x, \Sigma_x) \to \{0,1\}$. On input a verification key vk, a message x and a signature Σ_x, Verify(·) outputs 0 for rejecting the signature or 1 for accepting the signature.
- Extract$(ek, vk, C) \to \{msg, \perp\}$. On input an extraction key ek, the verification key vk and a circuit C, Extract(·) outputs an embedded message or a symbol $\perp$.

Correctness. For every message x and embedded message *msg*, the following holds:

$$\Pr\left[\begin{array}{c|c} \text{Verify}(vk, x, \Sigma_x) \neq 1 \vee \\ \text{Extract}\left(ek, vk, \text{Sign}_{sk}(\cdot)\right) \\ \neq msg \end{array} \middle| \begin{array}{c} ek \leftarrow \text{WM.Gen}(1^\lambda) \\ (vk^0, sk^0) \leftarrow \text{SIG.Gen}^0(1^\lambda) \\ (vk, sk) \leftarrow \text{SIG.Gen} \\ (1^\lambda, ek, msg, vk^0, sk^0) \\ \Sigma_x \leftarrow \text{Sign}(vk, sk, x) \end{array}\right] \leq \text{negl}(\lambda).$$

(Selective-)Existential Unforgeability Under the Chosen Message Attack. This notion is captured by an experiment between a challenger and an attacker $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ as: $\text{Exp}_{\mathcal{A}}^{wm-sel}(\lambda)$:

1. $x^* \leftarrow \mathcal{A}_0(1^\lambda)$
2. $ek \leftarrow \text{WM.Gen}(1^\lambda)$
3. $(vk^0, sk^0) \leftarrow \text{SIG.Gen}^0(1^\lambda)$
4. $(vk, sk) \leftarrow \text{SIG.Gen}(1^\lambda, ek, msg, vk^0, sk^0)$
5. $\sigma^* \leftarrow \mathcal{A}_1^{\text{sign}(sk,\cdot)}(vk)$. When the attacker requires a signature for x^*, the challenger returns $\perp$.
6. If Verify$(vk, x^*, \sigma^*) = 1$, output 1. Otherwise, output 0.

The watermarkable signature is (selective-)existential unforgeable under the chosen message attack if $\Pr[\text{Exp}_{\mathcal{A}}^{wm-sel}(\lambda)=1] \leq \text{negl}(\lambda)$.

A. ϵ-Unremovability

Definition 5. Fix a security parameter λ. For a watermarking signature scheme Π_{wmsign} = (WM.Gen, SIG.Gen^0, SIG.Gen, Sign, Verify, Extract), we say an adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ is unremoving-admissible if the following conditions hold:

- The adversary $\mathcal{A}$ makes exactly one challenge query.
- The circuit $\tilde{C}$ output by the adversary $\mathcal{A}$ possesses almost the same distribution as the circuit C^* output by the challenger. Except ϵ of the whole message space, $\tilde{C}$ must output a correct corresponding signature. Denote by $\tilde{C} \approx_\epsilon C^*$.

For every unremoving-admissible probabilistic polynomial time adversary $\mathcal{A}$ and an embedded message *msg*,

$$\Pr\left[\begin{array}{c|c} \tilde{C} \approx_\epsilon \text{Sign}_{sk^*}(\cdot) \wedge \\ \text{Extract}\left(ek, vk^*, \tilde{C}\right) \\ \neq msg^* \end{array} \middle| \begin{array}{c} ek \leftarrow \text{WM.Gen}(1^\lambda) \\ msg^* \leftarrow \mathcal{A}_1(1^\lambda) \\ (vk^{0,*}, sk^{0,*}) \leftarrow \text{SIG.Gen}^0(1^\lambda) \\ (vk^*, sk^*) \leftarrow \text{SIG.Gen} \\ \left(1^\lambda, ek, msg, vk^{0,*}, sk^{0,*}\right) \\ \tilde{C} \leftarrow \mathcal{A}_2^{\text{SIG.Gen}()}\left(1^\lambda, vk^*, \text{Sign}_{sk^*}(\cdot)\right) \end{array}\right] \leq \text{negl}(\lambda).$$

Remark. Definition 4 formulates a private-key watermarking signature scheme. The extractor must possess the private key to extract the message from the marked circuit. A stronger notion for watermarking is publicly-extractable watermarking. The extractor can extract the message without knowledge of the private key. In [5], the definition for a watermarking signature scheme is in the setting of public-key setting. In this work, we consider a watermarking signature scheme in the private-key setting and the marked circuit is embedded with a message instead of merely a $mark$ or $unmark$ label.

B. A Concrete Scheme

To construct a watermarking signature scheme, a secure PRF and a secure constrained signature scheme are required. We suppose that a secure PRF Π_{PRF} = (PRF.Gen, PRF.Eval) maps $(\{0,1\}^{m\prime})^d$ to $(\{0,1\}^{m\prime})^{2t}$. The watermarking signature scheme Π_{wmsign} = (WM.Gen, SIG.Gen^0, SIG.Gen, Sign, Verify, Extract) is defined as follows:

WM.Gen(1^λ): Sample $k^* \leftarrow \text{PRF.Gen}(1^\lambda)$. Choose a set of short matrices $\mathbf{U}_i \in \mathbb{Z}_q^{m\prime \times m}$ with $\| \mathbf{U}_i \|_\infty < \beta$ for $i = 1, 2, \ldots, t$. Uniformly sample $\mathbf{h}_j \in \{0,1\}^h$ for $j = 1, 2 \ldots, d$ and a matrix $\mathbf{B} \xleftarrow{\$} \{-1, 1\}^{m\prime \times m\prime}$. Set an extraction key as $ek =$

$(k^*, \mathbf{B}, \mathbf{U}_1, \dots, \mathbf{U}_t, \mathbf{h}_1, \dots, \mathbf{h}_d)$.

SIG.Gen $^0(1^\lambda)$: $(\mathbf{A} \in \mathbb{Z}_q^{n\times m'}, \mathrm{T_A}) \leftarrow TrapGen(1^n, 1^{m'}, q)$; sample uniformly a matrix $\overrightarrow{\mathbf{A}} \xleftarrow{\$} \mathbb{Z}_q^{n\times(m\times t)}$. And set $vk^0 = (\mathbf{A}, \overrightarrow{\mathbf{A}})$ and $sk^0 = \mathrm{T_A}$.

SIG.Gen(1^λ, ek, msg, vk^0, sk^0)**:** Compute $\mathbf{c} = (\mathbf{c}_1^0, \mathbf{c}_1^1, \mathbf{c}_2^0, \mathbf{c}_2^1, \dots \mathbf{c}_t^0, \mathbf{c}_t^1) \leftarrow$ PRF.Eval($k^*, \mathbf{Ah}_1, \dots, \mathbf{Ah}_d$) and $\mathbf{c}_i = \mathbf{ABc}_i^{msg_i}$ for $i = 1,2, \dots, t$.

- We suppose that f is the function to determine whether the signed message is equal to one element of a t-size set. Specifically, $f = \bigvee_{i=1}^{t} eq(x, y_i)$, where $eq(x, y) = 1$ if $x = y$ and $eq(x, y) = 0$ if $x \neq y$. The function f is defined with respect to the marked message *msg*, namely, $y_i = \mathbf{c}_i^{msg_i}$ for $i = 1, 2 \dots, t$.
- We compute $\mathbf{H}_f \leftarrow$ EvalF($f, \overrightarrow{\mathbf{A}}$). And $\mathbf{A}_f = \overrightarrow{\mathbf{A}} \cdot \mathbf{H}_f$. Then, we obtain $sk_f \leftarrow$ ExtensionTrap($sk^0, [\mathbf{A}||\mathbf{A}_f]$) with $sk^0 = \mathrm{T_A}$.

Set $vk = (\mathbf{A}, \overrightarrow{\mathbf{A}}, \{\mathbf{c}_i\}_{i=1}^t)$, $sk = (sk_f, \mathbf{U}_1, \dots, \mathbf{U}_t)$.

Sign(vk, sk, x): Procedure Sign($\cdot$) signs almost all x's from domain except several points that are relative to the marked message and the outputs of the PRF.

- If $f(x) \neq 0$, return $\perp$.
- Otherwise, generate a trapdoor T for $[\mathbf{A}||\overrightarrow{\mathbf{A}} - x \otimes \mathbf{G}]$ using the constraint signing key sk_f. Then, generate the signature $\sigma_x \leftarrow$ SamplePre($[\mathbf{A}||\overrightarrow{\mathbf{A}} - x \otimes \mathbf{G}]$, T, 0).
- $\mathbf{V}_i = \mathbf{AU}_i + x_i\mathbf{G}$, and $\mathbf{H}_i = [\mathbf{A}|\mathbf{V}_i + (x_i - 1)\mathbf{G}]$, where x_i is the i-th bit of x. Set σ_i = SampleRight ($\mathbf{H}_i$, $\mathbf{U}_i$, $\mathbf{c}_i$) for $i = 1, \dots, t$. $\sigma_i \in \mathbb{Z}_q^{m'+m}$. $\| \sigma_i \|_\infty < \beta\sqrt{mm'}\sqrt{m + m'}$.
- Output $\Sigma = (\sigma_x, \{\mathbf{V}_i, \sigma_i\}_{i=1}^t)$ as the signature for x.

Verify(vk, x, Σ): Express $\Sigma = (\sigma_x, \{\mathbf{V}_i, \sigma_i\}_{z=1}^t)$ as the signature for x. If the following three conditions are satisfied, output is 1; otherwise, 0.

- $\sigma_i \neq \perp$and $\| \sigma_i \|_\infty < \beta\sqrt{mm'}\sqrt{m + m'}$ for all $i = 1, 2, \dots, t$.
- $[\mathbf{A}|\mathbf{V}_i + (x_i - 1)\mathbf{G}] \cdot \sigma_i = \mathbf{c}_i$ for all $i = 1, 2, \dots, t$.
- $[\mathbf{A}||\overrightarrow{\mathbf{A}} - x \otimes \mathbf{G}] \cdot \sigma_x = 0$ and $\| \sigma_x \|_\infty < B$, $\sigma_x \neq \perp$, $\sigma_x \neq 0$.

Extract($ek, vk, \mathrm{Sign}_{sk}(\cdot)$): For an arbitrary valid signature, parse a signature Σ for a message x into $(\sigma_x, \{\mathbf{V}_i, \sigma_i\}_{i=1}^t)$.

- Compute $\mathbf{c} = (\mathbf{c}_1^0, \mathbf{c}_1^1, \mathbf{c}_2^0, \mathbf{c}_2^1, \dots \mathbf{c}_t^0, \mathbf{c}_t^1) \leftarrow$ PRF.Eval($k^*, \mathbf{Ah}_1, \dots, \mathbf{Ah}_d$).
- If $[\mathbf{A}|\mathbf{V}_i + (x_i - 1)\mathbf{G}] \cdot \sigma_i = \mathbf{ABc}_i^0$ or $\mathrm{Sign}_{sk}(\mathbf{c}_i^0) = \perp$, then $msg_i = 0$; if $[\mathbf{A}|\mathbf{V}_i + (x_i - 1)\mathbf{G}] \cdot \sigma_i = \mathbf{ABc}_i^1$ or $\mathrm{Sign}_{sk}(\mathbf{c}_i^1) = \perp$, $msg_i = 1$, for all $i = 1, 2, \dots, t$. Otherwise, output $\perp$.

C. Parameter Choices

For the above scheme, we require that $\beta = q$, $n > \lambda$, $q \leq 2^n$, $t = poly(\lambda) \leq 2^n$, $m = n\lceil \log q \rceil$ and $m' = \max\{m_0, (n + 1)\lceil \log q \rceil + 2\lambda\}$, where m_0 is required in Theorem 2. $B = \tau\sqrt{t}2^k m^{k+1}\sqrt{m' + tm}$, where $\tau$is set to be $\max\{\sqrt{m'}t2^k m^{1.5+k}, O(\sqrt{n\lceil \log q \rceil \log n})\}$ and k is the depth of circuit f. All parameters are required identically to those in [20].

V. Security Proofs

In this section, we prove that the watermarking scheme for constrained signatures in Section IV.*B* satisfies properties of verification correctness, extraction correctness, existential unforgeability, privacy, and ϵ-unremovability.

A. Correctness

Theorem 6. If parameters are chosen as above, then the construction of watermarking signature scheme is correct.

Proof. For convenience, we use the same set of notations from Section IV.*B*. Suppose that a signature Σ for a message x is output by the procedure Sign($\cdot$) which is a watermarked circuit with a message " msg ". Next, we will prove that $\mathrm{Verify}(vk, x, \Sigma) = 1$ and $\mathrm{Extract}(ek, vk, \mathrm{Sign}_{sk}(\cdot)) = msg$ where msg is the correct embedded message with non-negligible probability.

First, we prove that $\mathrm{Verify}(vk, x, \Sigma) = 1$. Suppose that $\Sigma = (\sigma_x, \{\mathbf{V}_i, \sigma_i\}_{i=1}^t)$. According to **Corollary 2**, σ_i follows a distribution that is not far from the Discrete Gaussian Distribution $\mathcal{D}_{\mathbb{Z}^{m'+m}, \tau}$ $(\tau = O(\sqrt{mm'}\beta)$.

Lemma 1 states that σ_i is bounded by $\tau\sqrt{m' + m}$ with non-negligible probability. Thus, the first condition holds. The equation $[\mathbf{A}|\mathbf{V}_i + (x_i - 1)\mathbf{G}] \cdot \sigma_i = \mathbf{c}_i$ holds for all $i = 1, 2, \dots, t$. This is guaranteed by the procedure SampleRight(). Thus, the second condition holds. The remaining verification holds due to the correctness of underlying constrained signature scheme in [20].

Secondly, we prove that $\mathrm{Extract}(ek, vk, \mathrm{Sign}_{sk}(\cdot)) = msg$. Since the secret key k^* is shared by both SIG.Gen($\cdot$) and Extract($\cdot$), the vectors **c**'s are identical in these two procedures. Moreover, the procedure SampleRight() guarantees the correctness of verification equations in Extract($\cdot$) . The $\mathrm{Sign}_{sk}(\cdot)$ condition is used additionally to choose the correct message bit since the $\mathrm{Sign}_{sk}(\cdot)$ can not sign a signature for $\mathbf{c}_i^{msg_i}$ for all $i = 1, 2 \dots, t$.

B. (Selective) Existential Unforgeability under Chosen Message Attack

Theorem 7. If the PRF is secure and the underlying constrained signature scheme is selectively existential unforgeable under chosen message attack, under the $\mathrm{SIS}_{n,m',q,D}$ assumption where $D = (m + 1)\beta\sqrt{mm'}\sqrt{m + m'} + m'$, then the watermarking signature scheme in the above section is selectively existential unforgeable under chosen message attack.

Proof. In our scheme, the signature Σ for a message x is composed of two parts: one is generated by the underlying constrained signature; the other is generated by SampleRight($\cdot$) algorithm. Thus, we prove these two parts are unforgeable separately.

We assume that there exists an adversary $\mathcal{A}$ who can successfully forge a signature in the selective setting for the watermarking signature scheme with non-negligible probability, then there exists an adversary $\mathcal{C}$ who can break $\mathrm{SIS}_{n,m',q,D}$. At the beginning of the unforgeability experiment, the adversary sends a message x^* as its challenge.

We firstly prove that under the $\mathrm{SIS}_{n,m\prime,q,D}$ assumption, the second part of the signature Σ is unforgeable. By Theorem 2, the matrix A is sampled approximately uniformly at random. By our construction, matrices $\{\mathrm{U}_i\}_{i=1}^{t}$ are sampled uniformly at random. Thus, the matrix AU_i is sampled uniformly at random. Uniformly at random sample a matrix R_i from $\{-1,1\}^{m\prime\times m}$. According to the leftover hash lemma (Lemma 2), if we set $\mathrm{AU}_i = \mathrm{AR}_i - (2x_i^* - 1)\mathrm{G}$, then the distribution of signature Σ will not change. If the adversary successfully forges a signature component σ_i^*, then according to our construction, we have:

$$[\mathbf{A}|\mathbf{V}_i + (x^*{}_i - 1)\mathbf{G}] \times \sigma_i^* = \mathbf{A}\mathbf{B}\mathbf{c}_i^{msg_i}.$$

If we write $\sigma_i^* = (\sigma_{1i}^*, \sigma_{2i}^*)$, then the above equation implies:

$$\mathbf{A}(\sigma_{1i}^* + \mathbf{R}_i\sigma_{2i}^* - \mathbf{B}\mathbf{c}_i^{msg_i}) = 0.$$

Since

$$\|\sigma_{1i}^* + \mathbf{R}_i\sigma_{2i}^* - \mathbf{B}\mathbf{c}_i^{msg_i}\| \leq \beta\sqrt{mm'}\sqrt{m+m'} + m\beta\sqrt{mm'}\sqrt{m+m'} + m',$$

$(\sigma_{1i}^* + \mathbf{R}_i\sigma_{2i}^* - \mathbf{B}\mathbf{c}_i^{msg_i})$ is a valid solution to $\mathrm{SIS}_{n,m\prime,q,D}$ where $D = (m+1)\beta\sqrt{mm'}\sqrt{m+m'} + m'$.

Next, we prove that the first part is unforgeable. Assume that there exists an adversary $\mathcal{A}$ who can successfully forge a signature in the selective setting for the watermarking signature scheme with non-negligible probability, then there exists an adversary $\mathcal{B}$ who can forge a signature for the constrained signature scheme with overwhelming probability.

When $\mathcal{A}$ makes a challenge query to require the signature for x^*, adversary $\mathcal{B}$ trivially send x^* to the challenger. The challenger sends $(\mathbf{A}, \overrightarrow{\mathbf{A}})$ chosen as in the constrained signature scheme as the verification key to $\mathcal{B}$. Before answering any signature query from the adversary $\mathcal{A}$ for $x \neq x^*$, adversary $\mathcal{B}$ randomly chooses $\{\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_t\} \in \mathbb{Z}_q^n$ and output $(\mathbf{A}, \overrightarrow{\mathbf{A}}, \{\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_t\})$ as the verification key to adversary $\mathcal{A}$. Adversary $\mathcal{B}$ chooses a set of short matrices $\{\mathbf{U}_i\}_{i=1}^{t}$ as a secret. Since the underlying PRF is secure, this modification is indistinguishable from the real SIG.Gen(·) of the watermarking signature scheme except the generation of the constrained secret signing key. When adversary $\mathcal{A}$ requires a signature for x, adversary $\mathcal{B}$ trivially sends x and a function $f_x(z) = z - x$ to the challenger. Meanwhile, adversary $\mathcal{B}$ runs $\sigma_i \leftarrow \mathrm{SampleRight}([\mathbf{A}|\mathbf{A}\mathbf{U}_i + (2x_i - 1)\mathbf{G}], \mathbf{c}_i)$ and sends $\{\mathbf{A}\mathbf{U}_i + x_i\mathbf{G}, \sigma_i\}_{i=1}^{t}$ along with the signature σ_x from the challenger to adversary $\mathcal{A}$ as the final signature for x. Obviously, adversary $\mathcal{B}$ perfectly simulates the watermarking signature scheme for adversary $\mathcal{A}$. When adversary $\mathcal{A}$ aborts the experiment, it would send a signature Σ_{x^*} as the forge to $\mathcal{B}$. Adversary $\mathcal{B}$ just extracts the corresponding part as the signature forge for x^* in the experiment for breaking the underlying constrained signature scheme. Hence, $\mathcal{B}$ has at least the same advantage as $\mathcal{A}$ to win the experiment. Since the underlying constrained signature scheme is selectively existential unforgeable under chosen message attack, this is a contradiction.

C. Privacy

In this section, we treat our construction in Section IV.*B* as a constrained signature scheme. In the following, we prove our constrained signature scheme satisfies privacy.

Theorem 8. If the PRF is secure, the underlying constrained signature scheme is strongly-hiding, then the watermarking scheme for signatures is strongly-hiding.

Proof. Since the PRF is secure, $(\mathbf{c}_1^0, \mathbf{c}_1^1, \mathbf{c}_2^0, \mathbf{c}_2^1, \ldots \mathbf{c}_t^0, \mathbf{c}_t^1)$ is indistinguishable from uniform random variables. By the leftover hash lemma, AB is sampled approximately uniformly at random. Thus, for all $b \in \{0,1\}$ and $i \in [t]$, ABc_i^b follows a distribution which is not far from uniform sampling. In other words, for all $b \in \{0,1\}$ and $i \in [t]$, c_i^b is hidden from ABc_i^b. By Theorem 5, the underlying constrained signature scheme is strongly-hiding. Therefore, the property of hiding the constraint functions is implied by the privacy of the underlying constrained signature scheme.

D. ϵ-Unremovability

Before proving the unremovability of the constructed watermarking signature scheme, the experiment game between an adversary and a challenger is defined in the following. An adversary is allowed to make queries to a marking oracle for more than once while make queries to a challenge oracle merely once.

Experiment I.

- **Setup Phase**. Sample $k^* \leftarrow \mathrm{PRF.Gen}(1^\lambda)$. Choose a set of short matrices $\mathbf{U}_i \in \mathbb{Z}_q^{m\prime\times m}$ with $\| \mathbf{U}_i \|_\infty < \beta$ for $i = 1, 2 \ldots, t$. Uniformly sample $\mathbf{h}_j \in \{0,1\}^h$ for $j = 1, 2 \ldots, d$ and a matrix $\mathbf{B} \xleftarrow{\$} \{-1,1\}^{m\prime\times m\prime}$. Set $ek = (k^*, \mathbf{B}, \mathbf{U}_1, \ldots, \mathbf{U}_t, \mathbf{h}_1, \ldots, \mathbf{h}_d)$.
- **Query Phase**.
  - **Marking Oracle**. The adversary uniformly samples a matirx $\overrightarrow{\mathbf{A}} \xleftarrow{\$} \mathbb{Z}_q^{n\times(m\times t)}$ and runs $(\mathbf{A} \in \mathbb{Z}_q^{n\times m\prime}, \mathrm{T_A}) \leftarrow \mathrm{TrapGen}(1^n, 1^{m\prime}, q)$, then sends $(\mathbf{A}, \mathrm{T_A})$ and a message msg that it would like to embed. The challenger does the subsequent steps as in SIG.Gen(·). Then, the challenger sends a Sign(·) circuit to the adversary.
  - **Challenge Oracle**. The adversary chooses a challenge message msg^* sent to the challenger. The challenger runs $(vk^0, sk^0) \leftarrow \mathrm{SIG.Gen}^0(1^\lambda)$ and runs $(vk^*, sk^*) \leftarrow \mathrm{SIG.Gen}(1^\lambda, ek, msg^*, vk^0, sk^0)$. Then, it sends a challenge circuit $\widetilde{\mathrm{Sign}}_{\widehat{sk}^*}(\cdot)$ and the public verification key vk^* to the adversary.
- **Challenge Phase.** The adversary outputs a circuit $\widetilde{\mathrm{Sign}}_{\widehat{sk}^*}(\cdot)$.
- **Extraction Phase.** For an arbitrary valid signature $\tilde{\Sigma}$ output by the circuit $\widetilde{\mathrm{Sign}}_{\widehat{sk}^*}(\cdot)$, the extractor parses $\tilde{\Sigma}$ into $(\widetilde{\sigma_x}, \{\widetilde{\mathbf{V}_\iota}, \widetilde{\sigma_\iota}\}_{i=1}^{t})$. Then, it does the following calculation:
  - Compute $\hat{\mathbf{c}} = (\widehat{\mathbf{c}_1^0}, \widehat{\mathbf{c}_1^1}, \widehat{\mathbf{c}_2^0}, \widehat{\mathbf{c}\mathbf{c}_2^1}, \ldots \widehat{\mathbf{c}_t^0}, \widehat{\mathbf{c}_t^1}) \leftarrow \mathrm{PRF.Eval}(k^*, \widehat{\mathbf{A}}\mathbf{h}_1, \ldots, \widehat{\mathbf{A}}\mathbf{h}_d)$.
  - For all $i = 1, 2, \ldots, t$, if $[\widehat{\mathbf{A}}||\widetilde{\mathbf{V}_\iota} + (x_i - 1)\mathbf{G}] \cdot \widetilde{\sigma_\iota} = \widehat{\mathbf{A}}\mathbf{B} \cdot \widehat{\mathbf{c}_\iota^0}$ or $\widetilde{\mathrm{Sign}}_{\widehat{sk}^*}(\widehat{\mathbf{c}_\iota^0}) = \perp$, then $msg_i = 0$; if $[\widehat{\mathbf{A}}||\widetilde{\mathbf{V}_\iota} + (x_i -$

$1)\mathbf{G}] \cdot \tilde{\sigma}_i = \widehat{\mathbf{A}}\mathbf{B} \cdot \hat{\mathbf{c}}_i^1$ or $\widetilde{\mathrm{Sign}}_{\widehat{sk}^*}(\hat{\mathbf{c}}_i^1) = \perp$, then $msg_i = 1$. Otherwise, output $\perp$.

Here, $\widehat{\mathbf{A}}$ is part of the verification key for the challenge signature circuit.

Theorem 9. For any efficient and unremoving-admissible adversary $\mathcal{A}$, the constructed watermarking signature scheme possesses unremovability property.

Proof. The unremovability of our scheme is obvious. Removing an embedded message is equivalent to modifying part of the signature verification key.

Remark. When a watermarked signature scheme is distributed, the verification key is considered to be hard to change.

VI. Conclusion

In this work, we construct a watermarking scheme for constrained signatures under a standard lattice assumption in the symmetric key setting. This is the first concrete watermarking scheme for signatures with hidden watermarks and constraint functions in the symmetric key setting.

Our construction adopts the methods in [9] and [5] to embed a message in the original program. We adopt a constrained signature scheme described in [20] as part of the whole signature scheme. To hide the watermark, we use randomization. One shortcoming is that we expand the size of a signature. Possible future works include how to reduce the signature size, how to design a watermarking scheme for signatures with both hidden watermarks and constraint functions in the public-key setting. We also believe that the techniques we used in the paper may provide insights for other researchers in constructing watermarking schemes for other cryptographic constructions.

References

[1] A. Adelsbach, S. Katzenbeisser, and H. Veith, "Watermarking schemes provably secure against copy and ambiguity attacks," in *Proceedings of the 3rd ACM workshop on Digital rights management*. ACM, 2003, pp. 111–119.

[2] C. I. Podilchuk and E. J. Delp, "Digital watermarking: algorithms and applications," *IEEE signal processing Magazine*, vol. 18, no. 4, pp. 33–46, 2001.

[3] B. Barak, O. Goldreich, R. Impagliazzo, S. Rudich, A. Sahai, S. Vadhan, and K. Yang, "On the (im) possibility of obfuscating programs," in *Annual International Cryptology Conference*. Springer, 2001, pp. 1–18.

[4] B. Barak, O. Goldreich, R. Impagliazzo, S. Rudich, A. Sahai, S. Vadhan, and K. Yang, "On the (im) possibility of obfuscating programs," Journal of the ACM (JACM), vol. 59, no. 2, p. 6, 2012.

[5] A. Cohen, J. Holmgren, R. Nishimaki, V. Vaikuntanathan, and D. Wichs, "Watermarking cryptographic capabilities," in *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*. ACM, 2016, pp. 1115–1127.

[6] D. Naccache, A. Shamir, and J. P. Stern, "How to copyright a function?" in *International Workshop on Public Key Cryptography*. Springer, 1999, pp.188–196.

[7] M. Yoshida and T. Fujiwara, "Toward digital watermarking for cryp-tographic data," *IEICE transactions on fundamentals of electronics, communications and computer sciences*, vol. 94, no. 1, pp. 270–272, 2011.

[8] R. Nishimaki, "How to watermark cryptographic functions," in *Annual International Conference on the Theory and Applications of Crypto-graphic Techniques*. Springer, 2013, pp. 111–125.

[9] S. Kim and D. J. Wu, "Watermarking cryptographic functionalities from standard lattice assumptions," in *Annual International Cryptology Conference*. Springer, 2017, pp. 503–536.

[10] W. Quach, D. Wichs, and G. Zirdelis, "Watermarking prfs under standard assumptions: Public marking and security with extraction queries," in *Theory of Cryptography Conference*. Springer, 2018, pp. 669–698.

[11] R. Yang, M. H. Au, J. Lai, Q. Xu, and Z. Yu, "Collusion resistant watermarking schemes for cryptographic functionalities," *in Interna-tional Conference on the Theory and Application of Cryptology and Information Security*. Springer, 2019, pp. 371–398.

[12] S. Kim and D. J. Wu, "Watermarking prfs from lattices: Stronger security via extractable prfs," in *Annual International Cryptology Conference*. Springer, 2019, pp. 335–366.

[13] F. Baldimtsi, A. Kiayias, and K. Samari, "Watermarking public-key cryptographic functionalities and implementations," in *International Conference on Information Security*. Springer, 2017, pp. 173–191.

[14] R. Goyal, S. Kim, N. Manohar, B. Waters, and D. J. Wu, "Watermarking public-key cryptographic primitives," in *Annual International Cryptolo-gy Conference*. Springer, 2019, pp. 367–398.

[15] N. Hopper, D. Molnar, and D. Wagner, "From weak to strong water-marking," in *Theory of Cryptography Conference*. Springer, 2007, pp. 362–382.

[16] M. Ajtai, "Generating hard instances of lattice problems," in *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*. ACM, 1996, pp. 99–108.

[17] D. Micciancio, "Almost perfect lattices, the covering radius problem, and applications to ajtai's connection factor," *SIAM Journal on Computing*, vol. 34, no. 1, pp. 118–169, 2004.

[18] D. Micciancio and O. Regev, "Worst-case to average-case reductions based on gaussian measures," *SIAM Journal on Computing*, vol. 37, no. 1, pp. 267–302, 2007.

[19] D. Micciancio and C. Peikert, "Hardness of sis and lwe with small parameters," in *Advances in Cryptology–CRYPTO 2013*. Springer, 2013, pp. 21–39.

[20] R. Tsabary, "An equivalence between attribute-based signatures and homomorphic signatures, and new constructions for both," in *Theory of Cryptography Conference*. Springer, 2017, pp. 489–518.

[21] D. Micciancio and C. Peikert, "Trapdoors for lattices: Simpler, tighter, faster, smaller," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2012, pp. 700– 718.

[22] S. Agrawal, D. Boneh, and X. Boyen, "Lattice basis delegation in fixed dimension and shorter-ciphertext hierarchical ibe," in *Annual Cryptology Conference*. Springer, 2010, pp. 98–115.

[23] S. Agrawal, D. Boneh, and X. Boyen, "Efficient lattice (h) ibe in the standard model," in *Annual Inter-national Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2010, pp. 553–572.

[24] C. Gentry, A. Sahai, and B. Waters, "Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based," in *Advances in Cryptology–CRYPTO 2013*. Springer, 2013, pp. 75–92.

[25] J. Alperin-Sheriff and C. Peikert, "Faster bootstrapping with polynomial error," in *International Cryptology Conference*. Springer, 2014, pp. 297–314.

[26] D. Boneh, C. Gentry, S. Gorbunov, S. Halevi, V. Nikolaenko, G. Segev, V. Vaikuntanathan, and D. Vinayagamurthy, "Fully key-homomorphic encryption, arithmetic circuit abe and compact garbled circuits," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2014, pp. 533–556.

[27] S. Gorbunov, V. Vaikuntanathan, and D. Wichs, "Leveled fully homo-morphic signatures from standard lattices," in *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*. ACM, 2015, pp.469–477.

[28] Z. Brakerski and V. Vaikuntanathan, "Constrained key-homomorphic prfs from standard lattice assumptions," in *Theory of Cryptography Conference*. Springer, 2015, pp. 1–30.

[29] Z. Brakerski, D. Cash, R. Tsabary, and H. Wee, "Targeted homomorphic attribute-based encryption," in *Theory of Cryptography Conference*. Springer, 2016, pp. 330-360.

[30] Y. Dodis, R. Ostrovsky, L. Reyzin, and A. Smith, "Fuzzy extractors: How to generate strong keys from biometrics and other noisy data," *SIAM journal on computing*, vol. 38, no. 1, pp. 97-139, 2008.