ITC 2/46
Journal of Information Technology and Control
Vol. 46 / No. 2 / 2017
pp. 274-286
DOI 10.5755/j01.itc.46.2.14320
© Kaunas University of Technology

Homomorphic Signature from Chameleon Hash Functions

Received 2016/08/31 | Accepted after revision 2017/04/12

http://dx.doi.org/10.5755/j01.itc.46.2.14320

Homomorphic Signature from Chameleon Hash Functions

Dong Xie, Haipeng Peng, Lixiang Li
Information Security Center, State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing, 100876, China

Yixian Yang
Guizhou Provincial Key Laboratory of Public Big Data, Guizhou University, Guiyang, 550025, China
e-mail: penghaipeng@bupt.edu.cn
Information Security Center, State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing, 100876, China

Corresponding author: penghaipeng@bupt.edu.cn

Homomorphic signature schemes provide a feasible solution to the authenticity of computations on an untrusted server (e.g. cloud). In a homomorphic signature scheme, given a k -length message set $\mu = \{\mu_1, \mu_2, \cdots, \mu_k\}$ and its corresponding signed dataset $\delta = \{\delta_1, \delta_2, \cdots, \delta_k\}$, anyone can publicly perform homomorphic computations and produce a new signature δ' for the messages $\mu' = f(\mu_1, \mu_2, \cdots, \mu_k)$, where f is a function or a circuit. If the generated homomorphic signature δ' is valid, then the owner of the dataset (e.g. cloud users) convinces that μ' is indeed the correct output of the function f over the original messages even if he/she forgets them. In this work, the main contribution is to build a bridge between the leveled Fully Homomorphic Signature Scheme (FHSS) and Homomorphic Chameleon Hash Function (HCHF), which is a new cryptographic primitive introduced by us based on prior works. We first present the definition and specific construction of HCHF and then use this forceful technique to construct leveled fully homomorphic signature schemes for any polynomial-depth circuit. In our standard model scheme, the size of evaluated homomorphic signature grows polynomially in the depth of the circuit. The security of our scheme is based on the property of collision resistance of HCHF, which can be reduced to the Small Integer Solution (SIS) in hard random lattices.

KEYWORDS: homomorphic signature schemes, chameleon hash functions, small integer solution, lattice.

Introduction

Compared to some traditional number-theoretic primitives (e.g., factoring problem, discrete logarithm problem), the lattice-based cryptography has the following advantages: i) It is conceptual simple and can be efficient implemented; ii) It can resist so far to quantum cryptanalysis; iii) The lattice-based scheme enjoys the worst case complexity, i.e., any random instance is indeed asymptotically hard [4,22]. Due to these attractive and distinguishing features, lattice has been widely used to construct a large number of cryptographic schemes. Lattice-based cryptography can be used for constructing versatile theoretical applications ranging from functional encryption [2-3, 6, 9], to fully homomorphic encryption [11, 17-18, 25], and much more [7, 8, 19, 21].

Cloud computing enables users to store sensitive data in the untrusted sever and sometimes the untrusted cloud requires to perform computations on them. The privacy of data and the authentication of computation are two key secure challenges in this field. Homomorphic encryption schemes [11, 17-18,25] can maintain the privacy of user's data by encrypting them and the server can also homomorphic perform computations over the ciphertexts. In this paper, we only focus on the authenticity of homomorphic computation through the notion of homomorphic signatures. In a homomorphic signature scheme, given a signed dataset vector $\boldsymbol{\delta}$ and its corresponding message vector $\boldsymbol{\mu}$, anyone can homomorphically compute and produce a new signature $\boldsymbol{\delta}'$ for a message $\boldsymbol{\mu}'$and a circuit C. Given the public parameters and the tuple (C, $\boldsymbol{\mu}'$, $\boldsymbol{\delta}'$), anyone can verify that $\boldsymbol{\delta}'$ is indeed the signature of the message $\boldsymbol{\mu}'$. Note that the verification procedure can be performed without knowing the original dataset $\boldsymbol{\mu}$. In recent years, some homomorphic signature schemes have been proposed [7, 8, 10, 16, 26]. However, many prior works have many drawbacks. In particular, some of them are only homomorphic for linear functions [7, 16, 26] and the security proofs of several schemes are in the random oracle model [7, 16]. In 2011, Boneh and Freeman [7] introduced a linearly homomorphic signature scheme that authenticates vector subspaces of a given ambient space. In the same year, they presented a general definition of homomorphic signatures, and constructed the first homomorphic signature scheme which can compute arbitrary polynomial functions over signed data [8]. In fact, if we translate these functions to the circuits, then the size of evaluated signatures can grow exponentially in the depth of the circuits. Furthermore, the construction is based on the SIS problem in ideal lattice. Recently, Boyen et al. presented the first adaptively secure fully homomorphic signature scheme that can evaluate any circuit over signed data [10].

Chameleon hash function, related to the notion of non-interactive chameleon commitment schemes, was originally introduced by Brassard et al. [12]. Roughly speaking, a chameleon trapdoor hash function is a collision-resistance function with chameleon property, i.e., the holder of the trapdoor can easily find collisions for every input. In addition, anyone can compute the hash function using public parameters and the resulting probability distribution is statistically close to uniform over the range. Chameleon hash functions have been proven to be an extremely useful tool in many scenarios, especially in signature schemes. Mohassel showed a general construction for transforming any chameleon hash function to a strongly unforgeable one-time signature scheme [23]. Recently, Micciancio and Peikert [21] proposed a signature scheme with short parameters and proved its security with strong unforgeability under static chosen-message attack (su-scma). Krawczyk and Rabin [20] showed that there is a generic transformation from su-scma to su-acma (strong unforgeability under adaptive chosen-message attack) security using a family of chameleon hash functions.

The main contribution of this work is to build a bridge between FHSS and Homomorphic Chameleon Hash Function (HCHF). In [13], Cash et al. straightforwardly presented a simple chameleon hash function using the preimage sampleable function under standard lattice assumption. Along this line of work, we give the definition of HCHF and present a family of HCHFs, which is based on the SIS problem in hard random lattices. After that, we construct a leveled fully homomorphic signature scheme using the HCHF tool. Similar to [1], we use the ***SampleLeft*** algorithm to extract signatures in real scheme and use the ***SampleRight*** algorithm to response the adversary's signature queries in the simulation game. The construction is straightforward and the security of our scheme is based on the property of collision resistance of HCHF. In fact, our scheme is homomorphic for any function, and not like those ones in [7, 16, 26] just for linear function. Unlike several recent homomorphic signature schemes [7-8, 16], our scheme is secure in the standard model. These results show that our homomorphic scheme is attractive.

The remainder of this paper is organized in the following manner. We mainly introduce some basic knowledge about lattice and homomorphic signature scheme in section 2. In section 3 we focus on the definition of HCHF and the specific construction from the standard SIS problem. We describe our homomorphic signature scheme, and provide the parameters setting and security analysis in section 4. Section 5 presents the comparison between our scheme and some classical homomorphic signature schemes. Finally, we draw our conclusions in section 6.

Preliminaries

Notation

For any positive integer q, we denote the set $\{1,2,\cdots,q\}$ by $[q]$ and let $\mathbf{Z}_q$ denote the integer ring which represents as integers in $(-q/2, q/2)$. Vectors are assumed to be in column form and are written using bold lower-case letters (e.g. $\boldsymbol{x}$). Similarly, we use bold capital-case letters (e.g. $\boldsymbol{A}$) to represent matrices.

Given two matrices $\boldsymbol{A}_1 \in \mathbf{Z}_q^{n\times m_1}$ and $\boldsymbol{A}_2 \in \mathbf{Z}_q^{n\times m_2}$, we use $[\boldsymbol{A}_1 \| \boldsymbol{A}_2]$ to denote the $n\times(m_1+m_2)$ matrix formed by concatenating $\boldsymbol{A}_1$ and $\boldsymbol{A}_2$. For a matrix $\boldsymbol{A} \in \mathbf{Z}_q^{n\times m}$, let $s_{\boldsymbol{A}}$ denote the maximal singular values of $\boldsymbol{A}$ and use $\|\boldsymbol{A}\|$ to denote the maximum norm of column vector of the matrix $\boldsymbol{A}$, i.e., $\|\boldsymbol{A}\| = \max_{i\in m}\{\|\boldsymbol{a}_i\|\}$, where $\boldsymbol{a}_i$ is the column vector of $\boldsymbol{A}$.

We denote a negligible function $f(n)$ by $negl(n)$ if it is $o(n^{-c})$ for any fixed constant c. We say $f(n)$ is polynomial if it is $O(n^c)$ for any fixed constant c, and we use $poly(n)$ to denote it. Given two distributions X and Y over a countable domain Z, the statistical distance between them is defined as

$$\Delta = \frac{1}{2}\sum_{z\in Z}|X(z) - Y(z)| \tag{1}$$

The min-entropy of a random variable X is denoted by $H_\infty(X) = -\log(\max_{x\in X}\Pr[X = x])$. Given two random variables X and Y, the average min-entropy of X conditioned on (correlated) variable Y is defined as

$$\widetilde{H}_\infty(X|Y) = -\log(\boldsymbol{E}_{y\leftarrow Y}(\max_{x\in X}\Pr[X = x|Y = y])) \tag{2}$$

Lemma 1[14] *Given two random variables X and Y, let $\mathcal{Y}$ be the support of Y. Then*

$$\widetilde{H}_\infty(X|Y) \geq H_\infty(X) - \log(|\mathcal{Y}|) \tag{3}$$

Lattices and SIS problem

Generally speaking, a lattice is a discrete additive subgroup of $\mathbb{R}^n$. A (full rank) lattice Λ can be viewed as the set of all integer linear combinations of n linearly independent basis vectors $\boldsymbol{B} = \{\boldsymbol{b}_1, \boldsymbol{b}_2, \cdots, \boldsymbol{b}_n\}$. Using the matrix notation,

$$\Lambda = \mathcal{L}(\mathbf{B}) = \{\mathbf{Bc} = \sum_{i\in[n]} c_i\mathbf{b}_i : \mathbf{c}\in\mathbb{Z}^n\}. \tag{4}$$

A family of lattices, called as q-ary lattices, is of particular interest to many cryptographic applications.

Definition 1 (q-ary lattices). *For any positive integers $n, m, q (m \geq n)$, let $\boldsymbol{A} \in \mathbf{Z}_q^{n\times m}$ be a matrix. Define the following m-dimensional q-ary lattices:*

$$\Lambda(\boldsymbol{A}^t) = \{\boldsymbol{z}\in\mathbf{Z}^m | \exists \boldsymbol{c}\ s.t.\ \boldsymbol{z} = \boldsymbol{A}^t\boldsymbol{c} \bmod q\}; \tag{5}$$

$$\Lambda^\perp(\boldsymbol{A}) = \{\boldsymbol{z}\in \mathrm{Z}^m | \boldsymbol{Az} = \boldsymbol{0} \bmod q\}. \tag{6}$$

For any $\boldsymbol{v}\in\mathbf{Z}_q^n$ admitting an integral solution $\boldsymbol{x}\in\mathbf{Z}^m$ to $\boldsymbol{Ax} = \boldsymbol{v} \bmod q$, define the shifted lattice as

$$\Lambda_{\boldsymbol{v}}^\perp(\boldsymbol{A}) = \{\boldsymbol{z}\in \mathrm{Z}^m | \boldsymbol{Az} = \boldsymbol{v} \bmod q\}. \tag{7}$$

Definition 2 (Gaussian function). *For any real $s > 0$ and any $\mathbf{c}\in\mathbb{R}^n$, the n-dimensional Gaussian function $\rho_{s,\boldsymbol{c}}(\boldsymbol{x})$ is defined as*

$$\rho_{s,\boldsymbol{c}}(\boldsymbol{x}) = \exp(-\pi\|\boldsymbol{x}-\boldsymbol{c}\|^2/s^2), \tag{8}$$

where $\boldsymbol{x}$ is a n-dimensional vector in $\mathbb{R}^n$.

Definition 3 (Discrete Gaussian distribution). *For any real $s > 0$, any $\mathbf{c} \in \mathbb{R}^n$, and an n-dimensional lattice Λ, the discrete Gaussian distribution $D_{\Lambda,s,\boldsymbol{c}}$ over Λ is defined as*

$$D_{\Lambda,s,\boldsymbol{c}}(\boldsymbol{x}) = \rho_{s,\boldsymbol{c}}(\boldsymbol{x}) / \rho_{s,\boldsymbol{c}}(\Lambda), \tag{9}$$

where $\boldsymbol{x}$ is *a* vector in Λ. We omit s and $\boldsymbol{c}$ when they are taken to be 1 and $\boldsymbol{0}$, respectively.

Definition 4 (Small integer solution (SIS)). *Given positive integers n, m, q, a real constant β and a matrix $\boldsymbol{A} \in \boldsymbol{Z}_q^{n\times m}$ $(m \geq n)$, find a nonzero vector $\boldsymbol{u} \in \boldsymbol{Z}^m$, so that $\boldsymbol{Au} = \boldsymbol{0} \bmod q$ and $\|\boldsymbol{u}\| \leq \beta$.*

In fact, the $\text{SIS}(n,m,q,\beta)$ problem is equivalent to find a short nonzero vector $\|\boldsymbol{u}\| \leq \beta$ in the lattice $\Lambda^{\perp}(\boldsymbol{A})$. Micciancio and Regev showed that the worst case of various promise problems (e.g. GapSVP, GapCVP) can be reduced to the average case of the SIS problem [22].

Trapdoors for lattices and sampling algorithms

Lemma 2 ([4, 19]). *Given any integers $n \geq 1$, $q \geq 2$, and sufficiently large $m = O(n \log q)$, there are three efficient algorithms **TrapGen, SampleDom** and **SamplePre** having the following description:*

1 *The **TrapGen** randomly outputs a parity check matrix $\boldsymbol{A} \in \boldsymbol{Z}_q^{n\times m}$ and a trapdoor short basis $\boldsymbol{T_A}$ for $\Lambda^{\perp}(\boldsymbol{A})$ so that the output distribution of $\boldsymbol{A}$ is statistically close to uniform over $\boldsymbol{Z}_q^{n\times m}$.*

2 *The **SampleDom** produces a matrix $\boldsymbol{U}$ with $\|\boldsymbol{U}\| \leq s\sqrt{m}$ whose column vector is sampled from $D_{\boldsymbol{Z}^m,s}$, where $s \geq w(\sqrt{\log m})$. The output distribution $\boldsymbol{V} = \boldsymbol{AU}$ is statistically close to uniform over $\boldsymbol{Z}_q^{n\times m}$.*

3 *Given a matrix $\boldsymbol{A} \in \boldsymbol{Z}_q^{n\times m}$ together with its trapdoor $\boldsymbol{T_A} \in \boldsymbol{Z}^{m\times m}$, and a matrix $\boldsymbol{V} \in \boldsymbol{Z}_q^{n\times m}$, the **SamplePre** outputs a matrix $\boldsymbol{U} \in \boldsymbol{Z}_q^{m\times m}$ with the conditional distribution of $\boldsymbol{U} \leftarrow$ **SampleDom** so that $\boldsymbol{AU}=\boldsymbol{V}$ and $\|\boldsymbol{U}\| \leq s\sqrt{m}$, where $s \geq \|\widetilde{\boldsymbol{T}}_{\boldsymbol{A}}\| w(\sqrt{\log m})$.*

We also need two classic sampling algorithms [1, 13] (see Algorithm 1 and Algorithm 2). Essentially, the algorithm ***SampleLeft*** will be used in real signature system, and the algorithm ***SampleRight*** will be used to exact signatures for adversary's queried messages in the simulation game.

Algorithm 1 ***SampleLeft***$(\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{T_A}, \boldsymbol{v}, s)$

Require:

1 A random matrix $\boldsymbol{A} \in \boldsymbol{Z}_q^{n\times m_1}$ with rank n and a matrix $\boldsymbol{B} \in \boldsymbol{Z}_q^{n\times m_2}$;

2 A relatively "short" trapdoor basis $\boldsymbol{T_A}$ of $\Lambda_q^{\perp}(\boldsymbol{A})$ and a vector $\boldsymbol{u} \in \boldsymbol{Z}_q^n$;

3 A Gaussian parameter $s \geq \|\widetilde{\boldsymbol{T}}_{\boldsymbol{A}}\| w(\sqrt{\log(m_1 + m_2)})$;

Ensure: A vector $\boldsymbol{u} \in \boldsymbol{Z}^{m_1+m_2}$ sampled from a distribution statistically close to $D_{\Lambda_q^{v}(\boldsymbol{A}\|\boldsymbol{B}),s}$.

Algorithm 2 ***SampleRight***$(\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}, \boldsymbol{T_B}, \boldsymbol{v}, s)$

Require:

1 A random matrix $\boldsymbol{A} \in \boldsymbol{Z}_q^{n\times l}$ and a matrix $\boldsymbol{C} \in \boldsymbol{Z}^{l\times m}$;

2 A matrix $\boldsymbol{B} \in \boldsymbol{Z}_q^{n\times m}$ and the "short" basis $\boldsymbol{T_B}$ of $\Lambda_q^{\perp}(\boldsymbol{B})$;

3 A gaussian parameter $s \geq \|\widetilde{\boldsymbol{T}}_{\boldsymbol{B}}\| s_{\boldsymbol{C}} w(\sqrt{\log m})$, where $s_{\boldsymbol{C}}$ is the maximal singular value of $\boldsymbol{C}$.

Ensure: A vector $\boldsymbol{u} \in \boldsymbol{Z}^{m+l}$ sampled from a distribution statistically close to $D_{\Lambda_q^{v}(\boldsymbol{A}\|\boldsymbol{AC}+\boldsymbol{B}),s}$.

Homomorphic signature scheme: definition and security

Throughout this paper, let λ be the security parameter. We denote the message space by $\mathcal{M}$ and let $\mathcal{C}$ be a collection of circuits which take k inputs over the message space and generate an output in $\mathcal{M}$. Boneh and Freeman [8] first introduced the formal definition of a homomorphic signature scheme for a type of circuit $\mathcal{C}$. A $\mathcal{C}$-homomorphic signature scheme is a tuple of polynomial time algorithms $\Pi =$ (***KeyGen, Sign, Eval, Verify***) with the following syntax.

_ ***KeyGen***$(1^{\lambda}, 1^{k})$. The key generation algorithm takes as input the security parameter λ and the maximum size of the dataset k. It outputs a signing secret key *sk* and a public verification key *pk*.

_ ***Sign***(sk, τ, i, μ). The signing algorithm takes as input the secret key *sk*, a tag $\tau \in \{0,1\}^{\lambda}$, an index $i \in [k]$ and a message $\mu \in \mathcal{M}$. It outputs a signature δ.

_ ***Eval***$(pk, \tau, \{(\mu_i, \delta_i)\}_{i\in[k]}, C)$. The evaluation algorithm takes as input the public key *pk*, a tag τ, a collection of message-signature pairs $\{(\mu_i, \delta_i)\}_{i\in[k]}$, and a circuit $C \in \mathcal{C}$. It outputs a signature δ' for a message μ'.

_ ***Verify***$(pk, \tau, \mu, \delta, C)$. The verification algorithm takes as input the public pk, a tag τ, a message-signature pair (μ, δ), and a circuit $C \in \mathcal{C}$. It outputs either 1 (accept) or 0 (reject).

For correctness, we require that both the original signatures (generated by ***Sign***) and the evaluated signatures (generated by ***Eval***) are accepted. Specifically, we require that the following conditions hold.

1 For all tags $\tau \in \{0,1\}^{\lambda}$, all $\mu \in \mathcal{M}$, and all $i \in [k]$, if $\delta \leftarrow \boldsymbol{Sign}(sk, \tau, i, \mu)$, then we get $\boldsymbol{Verify}(pk, \tau, \mu, \delta, I_i) = 1$. In order to maintain the consistency of the verification algorithm, we use the circuit I_i to denote the identity mapping, namely, $I_i(\mu_1, \mu_2, \cdots, \mu_k) = \mu_i$.

2 For all tags $\tau \in \{0,1\}^{\lambda}$, all messages $(\mu_1, \mu_2, \cdots, \mu_k) \in \mathcal{M}^k$ and all circuits $C \in \mathcal{C}$, if $\delta_i \leftarrow \boldsymbol{Sign}(sk, \tau, i, \mu_i)$ and $\delta' \leftarrow \boldsymbol{Eval}(pk, \tau, \{(\mu_i, \delta_i)\}_{i\in[k]}, C)$, we have $\boldsymbol{Verify}(pk, \tau, C(\mu_1, \mu_2, \cdots, \mu_k)\, \delta', C) = 1$.

A signature scheme is fully homomorphic if it is homomorphic for all polynomial-size circuits. In this work, we construct leveled fully homomorphic signature schemes, i.e., they are homomorphic for all polynomial-depth circuits. Next, we define the selectively unforgeable security for homomorphic signature schemes via the following game between a probabilistic polynomial time adversary $\mathcal{A}$ and a challenger $\mathcal{S}$.

_ The adversary chooses $(\tau^*, \boldsymbol{\mu}^*, C^*)$ as the challenged information and gives all information to the challenger.

_ The challenger generates (pk, sk) and gives pk to the adversary.

_ The adversary can make arbitrary polynomial number of signing queries. In the i-th query, the adversary chooses a fresh tag $\tau_i \in \{0,1\}^{\lambda}$ and a k-length message set $(\mu_{i1}, \mu_{i2}, \cdots, \mu_k) \in \mathcal{M}^k$. The challenger generates the collection of signatures $(\delta_{i1}, \delta_{i2}, \cdots, \delta_k)$ for the i-th query and sends it to the adversary.

_ The adversary outputs a signature δ^* for the chosen tag τ^*, a message μ^* and the circuit C^*.

If $\boldsymbol{Verify}(pk, \tau^*, \mu^*, \delta^*, C^*) = 1$, then the adversary $\mathcal{A}$ wins the game. Due to the definition of selective unforgeability, the adversary can query the signatures of the challenged message vector $\boldsymbol{\mu}^*$. In order to make the challenger response for the challenger message vector, we set the adversary's challenged plaintext as a set of messages, rather than a single message. In fact, there are two types of forgers: one is $\tau^* \neq \tau_i$ for all queried i, and the other is $\tau^* = \tau_i$ for some index i but $\mu^* \neq C^*(\boldsymbol{\mu}^*)$.

Definition 5 (Selective Unforgeability). *A leveled homomorphic signature scheme* $\Pi = (\boldsymbol{KeyGen}, \boldsymbol{Sign}, \boldsymbol{Eval}, \boldsymbol{Verify})$ *is selectively unforgeable if for any probability polynomial time adversary, the probability of wining the above game is negligible.*

Homomorphic Chameleon Hash Functions: Definition and Construction

In [15], Freeman embed a homomorphic chameleon hash function to show the unforgeability of his homomorphic signature scheme. Based on this and the definition of chameleon hash function [12], a generic definition of HCHF are given in this section. Note that compared to chameleon hash function, HCHF has an additional property, i.e., homomorphism. Then we construct a class of HCHFs using the distinguished trapdoor function with preimage sampling technique [19, 21].

Definition 6 (Homomorphic Chameleon Hash Function). *For a message space $\mathcal{M}$ and a randomness space $\mathcal{U}$, a family of homomorphic chameleon hash functions is a collection $\mathcal{H} = \{h_i : \mathcal{M} \times \mathcal{U} \to \mathcal{V}\}$, where i is the index and $\mathcal{V}$ is the range. There is an algorithm which can generate a public index i and the corresponding trapdoor secret key $\boldsymbol{T}_i$. Homomorphic chameleon hash functions consist of the following four properties:*

_ ***Uniformity property.*** *For a randomized index i, $\mu \in \mathcal{M}$, and $u \in \mathcal{U}$, the statistical distance $\triangle((h_i, h_i(\mu, u)), (U_{\mathcal{H}}, U_{\mathcal{V}}))$ is negligible, where $U_{\mathcal{H}}$ and $U_{\mathcal{V}}$ denote the uniform distributions on $\mathcal{H}$ and $\mathcal{V}$.*

_ ***Chameleon property.*** *For any $\mu \in \mathcal{M}$ and $v \in \mathcal{V}$, given the trapdoor $\boldsymbol{T}_i$, anyone can efficiently compute $u \in \mathcal{U}$ so that $h_i(\mu, u) = v$.*

_ ***Collision resistance.*** *Given a public index i, there are no polynomial time adversary which can find a pair $(\mu, u) \neq (\mu^*, u^*)$ so that $h_i(\mu, u) = h_i(\mu^*, u^*)$.*

_ ***Homomorphic property.*** *Given a dataset $(\mu_j, u_j, v_j)_{j\in[k]}$ so that $h_i(\mu_j, u_j) = v_j$ and a circuit*

$C:\mathcal{M}^k \to \mathcal{M}$, anyone can homomorphically compute a u' from μ_j, u_j and a v' from v_j so that $h_i(C(\mu_1,\mu_2,\cdots,\mu_k)\ u') = v'$.

Next, we construct a class of specific HCHFs using the trapdoor technique from standard lattices [19, 21] and prove that it satisfies the above four properties. Let $\mathcal{M} = \mathbf{Z}_q$, $\mathcal{U} = \{\boldsymbol{U} \in \boldsymbol{Z}_q^{m\times m} : \|\boldsymbol{U}\|_2 \le B\}$ and $\mathcal{V} = \boldsymbol{Z}_q^{n\times m}$. We remark that B is the upper bound of the size of evaluated signatures in our homomorphic schemes. Every column of the matrix $\boldsymbol{U}$ is sampled from the distribution $D_{\mathbf{Z}^m,s}$, where s is the Gaussian parameter. All related parameters are defined in section 4.2. We use the ***TrapGen*** algorithm to generate the index and the corresponding trapdoor for our HCHF.

The primitive matrix $\boldsymbol{G}' \in \boldsymbol{Z}_q^{n\times n\lceil \log q\rceil}$, introduced in [21], has public trapdoor short basis $\boldsymbol{T}_{\boldsymbol{G}'}$ for $\Lambda^{\perp}(\boldsymbol{G}')$. Here we construct a new matrix $\boldsymbol{G} = [\boldsymbol{G}' \| \boldsymbol{R}'] \in \boldsymbol{Z}_q^{n\times m}$, where $\boldsymbol{R}' \in \boldsymbol{Z}_q^{n\times m - n\lceil \log q\rceil}$ is a random matrix. Using ***ExtBasis*** algorithm in [13], we can obtain a short basis $\boldsymbol{T}_{\boldsymbol{G}}$ for $\Lambda^{\perp}(\boldsymbol{G})$ so that $\|\boldsymbol{T}_{\boldsymbol{G}}\| = \|\boldsymbol{T}_{\boldsymbol{G}}\|$ [10]. Hence, anyone can efficiently perform ***SamlePre*** algorithm using the trapdoor $\boldsymbol{T}_{\boldsymbol{G}}$. We define the homomorphic chameleon hash function $h_{\boldsymbol{A}}$ with index $\boldsymbol{A}$ as follows:

$$h_{\boldsymbol{A}}(\boldsymbol{U},\mu) = \boldsymbol{A}\boldsymbol{U} + \mu\boldsymbol{G} \bmod q. \tag{10}$$

It is not difficult to verify the uniformity and chameleon properties of $h_{\boldsymbol{A}}$. Specifically, if μ is randomly sampled from $\boldsymbol{Z}_g$, we naturally get the result that the statistical distance $\Delta = ((\boldsymbol{A}, h_{\boldsymbol{A}}), (U_{\boldsymbol{A}}, U_{\mathcal{V}}))$ is negligible in n [19]. Given the trapdoor matrix $\boldsymbol{T}_{\boldsymbol{A}}$, we can use the algorithm ***SamplePre*** to compute $\boldsymbol{U}$ which has the same distribution as $D_{\mathbf{Z}^m,s}$ [19]. Next, we prove that the functions constructed by us satisfy the other two properties, i.e., collision resistance and homomorphism.

Theorem 1. *Given an integer $n = poly(\lambda)$, let $q = poly(\lambda)$ be a prime, $m = n\log q + w(\log n)$ and B be the upper bound of the size of signatures defined in section 4.2. If the $\mathrm{SIS}(n,m,q,\sqrt{m}(2mB+1)$ problem is hard, then the function $h_{\boldsymbol{A}}$ constructed above is collision resistance with probability $1-negl(n)$.*

Proof. Suppose that there is an adversary $\mathcal{A}$ that finds a collision $(\boldsymbol{U}_1,\mu_1)$ and $(\boldsymbol{U}_2,\mu_2)$ for a random function $f_{\boldsymbol{A}}$. Obviously, we have

$$\begin{aligned} \boldsymbol{A}\boldsymbol{U}_1 + \mu_1\boldsymbol{G} &= h_{\boldsymbol{A}}(\boldsymbol{U}_1,\mu_1) \\ &= h_{\boldsymbol{A}}(\boldsymbol{U}_2,\mu_2) \\ &= \boldsymbol{A}\boldsymbol{U}_2 + \mu_2\boldsymbol{G} \bmod q. \end{aligned} \tag{11}$$

That is, $\boldsymbol{A}(\boldsymbol{U}_1 - \boldsymbol{U}_2) = (\mu_2-\mu_1)\boldsymbol{G} \bmod q$.

If $\mu_1 = \mu_2$, then we have a nonzero matrix $\boldsymbol{U} = \boldsymbol{U}_1 - \boldsymbol{U}_2$ so that $\boldsymbol{A}\boldsymbol{U} = \boldsymbol{0} \bmod q$. Note that $\|\boldsymbol{U}_i\| \le B$, so we have $\|\boldsymbol{U}\| \le 2B$.

If $\mu_1 \ne \mu_2$, we first choose a vector $r \in \{0,1\}^m$ at random, and let $\boldsymbol{z} = \boldsymbol{A}\boldsymbol{r}$. Since $\boldsymbol{G}$ is a public primitive matrix and naturally has a trapdoor $\boldsymbol{T}_{\boldsymbol{G}}$, we can invoke the ***SamplePre*** to compute a vector $r' \in \{0,1\}^m$ so that $\boldsymbol{G}\boldsymbol{r}' = \boldsymbol{z}(\mu_2-\mu_1)^{-1} \bmod q$. We have

$$\begin{aligned} \boldsymbol{A}((\boldsymbol{U}_1 - \boldsymbol{U}_2)\boldsymbol{r}' - \boldsymbol{r}) &= ((\mu_2-\mu_1)\boldsymbol{G})\boldsymbol{r}' - \boldsymbol{A}\boldsymbol{r} \\ &= \boldsymbol{z} - \boldsymbol{z} = \boldsymbol{0} \bmod q. \end{aligned} \tag{12}$$

Hence, we get a vector $\boldsymbol{u} = (\boldsymbol{U}_1 - \boldsymbol{U}_2)\boldsymbol{r}' - \boldsymbol{r}$ so that $\boldsymbol{A}\boldsymbol{u} = \boldsymbol{0} \bmod q$. Using the Cauchy-Schwarz inequality, we easily have $\|\boldsymbol{u}\| \le \sqrt{m}(2mB+1)$. Next, we only need to prove that the probability of $\boldsymbol{u} = \boldsymbol{0}$ is negligible in n. Although r is randomly chosen from $\{0,1\}^m$, $\boldsymbol{r}'$ is mainly dependent on $\boldsymbol{z}$. Hence,

$$\begin{aligned} \widetilde{H}_\infty(\boldsymbol{r}|\boldsymbol{r}') &\ge \widetilde{H}_\infty(\boldsymbol{r}|\boldsymbol{z}) \\ &\ge H_\infty(\boldsymbol{r}) - \log(q^n) \\ &= m - n\log q \\ &= w(\log n). \end{aligned} \tag{13}$$

The second inequality follows from Lemma 1. Therefore, from the definition of average min-entropy,

$$\begin{aligned} \Pr[\boldsymbol{u} = \boldsymbol{0}] &= \Pr[\boldsymbol{r} = (\boldsymbol{U}_1 - \boldsymbol{U}_2)\boldsymbol{r}'] \\ &\le 2^{-w(\log n)} = negl(n). \end{aligned} \tag{14}$$

In summary, if there is an adversary $\mathcal{A}$ that finds a collision for a random function $f_{\boldsymbol{A}}$, then we can construct an algorithm to solve the $\mathrm{SIS}(n,m,q,\sqrt{m}$ $(2mB+1)$ problem with probability $1-negl(n)$. This concludes the proof.

For the homomorphic property, we consider general arithmetic circuit C. Specifically, we consider four types of gates: addition, multiplication, addition with constant, and multiplication with constant. These four special gates are completely used to compute an arbitrary arithmetic circuit [24].

Theorem 2. *Given an integer $n = poly(\lambda)$, let $q = poly(\lambda)$ be a prime and $m = O(n\log q)$. The function $h_{\boldsymbol{A}}$ constructed above is homomorphic for any arithmetic circuit.*

Proof. In order to prove this theorem, we consider the four types of gates in turn.

1 For an addition gate f, $f(\mu_1,\mu_2)=\mu_1+\mu_2$. Suppose that there are two datasets $(\boldsymbol{U}_i,\mu_i,\boldsymbol{V}_i)_{i=1,2}$ so that $h_{\boldsymbol{A}}(\boldsymbol{U}_i,\mu_i)=\boldsymbol{V}_i$. Then we have

$$\boldsymbol{V}_1=\boldsymbol{A}\boldsymbol{U}_1+\mu_1\boldsymbol{G} \bmod q, \tag{15}$$

$$\boldsymbol{V}_2=\boldsymbol{A}\boldsymbol{U}_2+\mu_2\boldsymbol{G} \bmod q. \tag{16}$$

Define $\boldsymbol{U}^*=\boldsymbol{U}_1+\boldsymbol{U}_2$ and $\boldsymbol{V}^*=\boldsymbol{V}_1+\boldsymbol{V}_2$. We can easily verify that $h_{\boldsymbol{A}}(\boldsymbol{U}^*,\mu_1+\mu_2)=\boldsymbol{V}^* \bmod q$.

2 Similarly, for a multiplication gate, let $f(\mu_1,\mu_2)=\mu_1\mu_2$. This time we firstly compute the matrix $\boldsymbol{R}\in\{0,1\}^{m\times m}$ so that $\boldsymbol{G}\boldsymbol{R}=\boldsymbol{V}_1 \bmod q$ [12]. Then we define $\boldsymbol{U}^*=\mu_2\boldsymbol{U}_1+\boldsymbol{U}_2\boldsymbol{R} \bmod q$ and $\boldsymbol{V}^*=\boldsymbol{V}_2\boldsymbol{R} \bmod q$.

Hen'ce,

$$\begin{aligned}\boldsymbol{A}\boldsymbol{U}^*+(\mu_1\mu_2)\boldsymbol{G}&=\boldsymbol{A}(\mu_2U_1+\boldsymbol{U}_2\boldsymbol{R})+(\mu_1\mu_2)\boldsymbol{G},\\&=\mu_2\boldsymbol{V}_1+\boldsymbol{A}\boldsymbol{U}_2\boldsymbol{R},\\&=\mu_2\boldsymbol{V}_1+\boldsymbol{V}^*-\mu_2\boldsymbol{V},\\&=\boldsymbol{V}^* \bmod q.\end{aligned} \tag{17}$$

3 For an addition with constant gate, $f(\mu,a)=\mu+a$. For the message μ, suppose that there are two matrices $\boldsymbol{U}$ and $\boldsymbol{V}$ so that $h_{\boldsymbol{A}}(\boldsymbol{U},\mu)=\boldsymbol{V}$. We define $\boldsymbol{U}^*=\boldsymbol{U}$ and $\boldsymbol{V}^*=\boldsymbol{V}-a\boldsymbol{G} \bmod q$. Obviously, $h_{\boldsymbol{A}}(\boldsymbol{U}^*,\mu+a)=\boldsymbol{V}^* \bmod q$ holds.

4 For a multiplication by constant gate, $f(\mu,a)=a\mu$. We define $\boldsymbol{U}^*=\boldsymbol{U}\boldsymbol{R} \bmod q$ and $\boldsymbol{V}^*=\boldsymbol{V}\boldsymbol{R} \bmod q$, where $\boldsymbol{G}\boldsymbol{R}=a\boldsymbol{G} \bmod q$. It is also easy to check that the equation $h_{\boldsymbol{A}}(\boldsymbol{U}^*,a\mu)=\boldsymbol{V}^* \bmod q$ holds.

Note that an arbitrary arithmetic circuit C can be expressed as the above four gate operations. For a circuit C, we compute $\boldsymbol{U}^*$ and $\boldsymbol{V}^*$ recursively gate by gate according to the above rules. Therefore, the function $h_{\boldsymbol{A}}$ constructed by us is homomorphic for any arithmetic circuit.

Our leveled homomorphic signature scheme

In this section, we firstly describe our proposed homomorphic signature scheme and then set related parameters for some types of circuits. After that, we give the correctness analysis and security proof for our scheme.

Our construction

In our construction, we employ the public primitive matrix $\boldsymbol{G}$ introduced by Micciancio and Peikert [21], which naturally has a short basis $\boldsymbol{T}_{\boldsymbol{G}}$ for $\Lambda^{\perp}(\boldsymbol{G})$. Our homomorphic signature scheme $\Pi=$ (***KeyGen***, ***Sign***, ***Eval***, ***Verify***) specifically works as follows.

- ***KeyGen***$(1^\lambda,1^k)$. The algorithm takes the security parameter λ and the maximum size of the dataset k as input.
  1 Choose the parameters n,q,m,s and B as in section 4.2.
  2 Sample a matrix $\boldsymbol{A}\in\mathbb{Z}_q^{n\times m}$ and its corresponding trapdoor matrix $\boldsymbol{T}_{\boldsymbol{A}}\in\mathbb{Z}^{m\times m}$.
  3 Choose $k+1$ random matrices $\boldsymbol{B}$ and $\{\boldsymbol{V}_i\}_{i\in[k]}\in\mathbb{Z}_q^{n\times m}$.
  4 Output the secret key $sk=\boldsymbol{T}_{\boldsymbol{A}}$ and the public key $pk=(\boldsymbol{A},\boldsymbol{B},\boldsymbol{G},\{\boldsymbol{V}_i\}_{i\in[k]})$.
- ***Sign***(sk,τ,i,μ). The algorithm takes the secret key $\boldsymbol{T}_{\boldsymbol{A}}$, a tag $\tau\in\{0,1\}^\lambda$, an index $i\in[k]$ and a message $\mu\in\mathcal{M}$ as input.
  1 Choose a specific homomorphic chameleon hash function $h_{\boldsymbol{A}_\tau}$ for the tag τ, where $\boldsymbol{A}_\tau=[\boldsymbol{A}\|\boldsymbol{B}+\tau\boldsymbol{G}]\in\mathbb{Z}_q^{n\times 2m}$.
  2 Use the secret key $\boldsymbol{T}_{\boldsymbol{A}}$ to compute $\boldsymbol{U}$ so that $h_{\boldsymbol{A}_\tau}(\boldsymbol{U},\mu)=\boldsymbol{V}_i$. Namely, $\boldsymbol{U}\leftarrow$ ***SampleLeft***$(\boldsymbol{A},\boldsymbol{B}+\tau\boldsymbol{G},\boldsymbol{T}_{\boldsymbol{A}},\boldsymbol{V}_i-\mu\boldsymbol{G},s)$.
  3 Output the signature $\delta=\boldsymbol{U}$.
- ***Eval***$(pk,\tau,\{(\mu_i,\delta_i)\}_{i\in[k]},C)$. The evaluation algorithm takes the public key pk, the tag τ, a collection of message-signature pairs $\{(\mu_i,\delta_i)\}_{i\in[k]}$, and a circuit $C\in\mathcal{C}$ as input. It recursively computes a homomorphic signature gate by gate.
  1 Compute the homomorphic chameleon hash function $h_{\boldsymbol{A}_\tau}$ for the tag τ.
  2 Let $f(\mu_1,\mu_2)$ be a gate in C, where μ_1 and μ_2 are the input messages. By induction, we have two signatures $\boldsymbol{U}_1$ and $\boldsymbol{U}_2$ so that $\boldsymbol{A}\boldsymbol{U}_1+\mu_1\boldsymbol{G}=\boldsymbol{V}_1$ and $\boldsymbol{A}\boldsymbol{U}_2+\mu_2\boldsymbol{G}=\boldsymbol{V}_2$. According to Theorem 2, we can homomorphically output the signature $\boldsymbol{U}^*$. Taking the multiplication gate as an example, $\boldsymbol{U}^*=\mu_2\boldsymbol{U}_1+\boldsymbol{U}_2\boldsymbol{R}$, where $\boldsymbol{R}\in\{0,1\}^{m\times m}$ so that $\boldsymbol{G}\boldsymbol{R}=\boldsymbol{V}_1$.
  3 Output the evaluated signature $\delta'=\boldsymbol{U}^C$.
- ***Verify***(pk,τ,μ,δ,C). The verification algorithm takes the public pk, the tag τ, a message-signature

pair (μ, δ), and a circuit $C \in \mathcal{C}$ as input. It outputs 1 if the following conditions hold, otherwise it outputs 0:

1. Let $\delta = \boldsymbol{U}^C$ and verify $\|\boldsymbol{U}^C\| \leq B$;
2. Let $\boldsymbol{A}_\tau = [\boldsymbol{A} \| \boldsymbol{B} + \tau \boldsymbol{G}]$ and check whether $h_{A_\tau}(\boldsymbol{U}^C, \mu) = C(\boldsymbol{V}_i)$ holds or not.

Parameters

Let λ be the security parameter in our scheme. Suppose that the maximum depth of the circuits in our scheme is $d = d(\lambda)$. We use B to denote the upper bound of the size of evaluated signatures, and use B_{int} to denote the size of the original signatures generated by ***Sign*** algorithm.

We assume that $n = poly(\lambda)$, $q = n^{O(d)}$ is a large prime, and $B = 2^{dw(\log \lambda)}$. Due to the ***TrapGen*** and Theorem 1, set the parameter $m = \max\{O(n \log q),\ n \log q + w(\sqrt{\log n})\} = ploy(\lambda)$. In order to use ***SampleLeft***, we need $s \geq \|\widetilde{\boldsymbol{T}}_{\boldsymbol{A}}\| w(\sqrt{\log m})$ where $\|\widetilde{\boldsymbol{T}}_{\boldsymbol{A}}\| \leq O(\sqrt{n \log q})$. Similarly, ***SampleRight*** requires that $s \geq \|\widetilde{\boldsymbol{T}}_{\boldsymbol{G}}\| s_{\boldsymbol{W}} w(\sqrt{\log m})$, where $\boldsymbol{W} \in \{-1,1\}^{m \times m}$ and $s_{\boldsymbol{W}} = O(\sqrt{m})$ [23]. Hence, we use sufficiently large $s = O(\sqrt{n \log q}) w(\sqrt{\log m})$ so that the outputs of ***SampleLeft*** and ***SampleRight*** are indistinguishable. If C is a boolean circuit of maximum depth d, whatever the gate is, we also have $\|\boldsymbol{U}^*\| \leq B_{\text{int}}(m^{1.5}+1)$. Hence, the size of evaluated signatures $\|\boldsymbol{U}^C\| \leq B_{\text{int}} (m^{1.5}+1)^d \leq mw(\sqrt{\log m})(m^{1.5}+1)^d \leq 2^{dw(\log \lambda)} = B$. Next we consider that C is an arithmetic circuit of maximum depth d consisting of fan-in-t addition gates and fan-in-2 multiplication gates, where $t = poly(\lambda)$. Moreover, it is guaranteed that at least one input $\ mu$ about this fan-in-2 multiplication gate is of size polynomial in λ. From Theorem 2, $\|\boldsymbol{U}^*\| \leq B_{\text{int}} \max \{t, m^{1.5} + |\mu|\}$.

Hence, $\|\boldsymbol{U}^*\| \leq B_{\text{int}} \max\{t, m^{1.5} + |\mu|\}^d \leq m\, w(\sqrt{\log m}) \max\{t, m^{1.5} + |\mu|\}^d \leq 2^{dw(\log \lambda)} = B$.

Correctness and security proof

From the parameters setting defined in section 4.2, it is easy to see that the signatures produced by ***Sign*** are correct. The correctness of signatures generated by ***Eval*** follows from the homomorphic property of HCHF. In this subsection, we mainly discuss the security of our scheme.

Theorem 3. *For any adversary $\mathcal{A}$ mounting a selective unforgeability attack with at most Q queries on our homomorphic signature scheme Π, there is a probabilistic polynomial time algorithm $\mathcal{S}$ that can find a collision for the randomized HCHF with the following advantage,*

$$\boldsymbol{Adv}_{HCHF}(\mathcal{S}^{\mathcal{A}}) \geqslant \boldsymbol{Adv}_{\Pi}^{selective}(\mathcal{A})/Q - negl(n). \tag{18}$$

Proof. Let A be an adversary that wins the selective unforgeability security game defined in section 2.4 with advantage $\boldsymbol{Adv}_{\Pi}^{selective}(\mathcal{A})$. Our aim is to construct an algorithm $\mathcal{S}$ which can find a collision for fully homomorphic chameleon function $h_{\boldsymbol{A}}$ over the random $\boldsymbol{A} \in \mathbf{Z}_q^{n \times m}$, where n, q, m are defined in section 4.2. The algorithm $\mathcal{S}$ takes a matrix $\boldsymbol{A}$ whose columns are independent and uniformly random samples from $\mathbf{Z}_q^n$ as input. Let $\tau^*, \boldsymbol{\mu}^*, C^*$ be the challenge information about tag, messages, and circuit. Suppose that the adversary makes Q queries and everytime the tag is τ_i, where $i \in [Q]$. We distinguish between two types of forgers. One is that the adversary will never query all signatures of messages for the tag τ^*, i.e., $\tau^* \neq \tau_i$ for all $i \in [Q]$. The other one is $\tau^* = \tau_i$ for some tag i, but $C^*(\boldsymbol{\mu}^*) \neq \mu^*$, where μ^* is the adversary's forged message.

1. We first consider the situation, where $\tau^* = \tau_i$ for all $i \in [Q]$. The simulation step is as follows:
   - The challenger $\mathcal{S}$ generates a public key for the adversary $\mathcal{A}$. Choose the public parameters n, q, m. Let s be the related Gaussian parameter and denote the upper bound on the size of evaluated signature by B. See section 4.2 for more details. Sample $\boldsymbol{W} \in \{-1,1\}^{m \times m}$ randomly and let $\boldsymbol{B} = \boldsymbol{AW} - \tau^* \boldsymbol{G} \bmod q$. For all $i \in [k]$, choose matrix $\boldsymbol{W}_i \in \{-1,1\}^{m \times m}$ at random and compute $\boldsymbol{V}_i = \boldsymbol{AW}_i$. Output the public key $(\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{G}, \{\boldsymbol{V}_i\}_{i \in [k]})$.
   - The challenger $\mathcal{S}$ generates signatures for the queried messages and the tag τ_i. Since $[\boldsymbol{A} \| \boldsymbol{AW} + (\tau_i - \tau^*)\boldsymbol{G}] = [\boldsymbol{A} \| \boldsymbol{B} + \tau_i \boldsymbol{G}] = \boldsymbol{A}_{\tau_i}$, we can use the trapdoor $\boldsymbol{T}_{\boldsymbol{G}}$ to compute the signature $\boldsymbol{U}_{ij}$ so that $\boldsymbol{A}_{\tau_i}(\boldsymbol{U}_{ij}, \mu_j^*) = \boldsymbol{V}_j$. Namely, $\boldsymbol{U}_j \leftarrow$ ***SampleRight*** $(\boldsymbol{A}, (\tau_i - \tau^*)\boldsymbol{G}, \boldsymbol{W}, \boldsymbol{T}_{\boldsymbol{G}}, \boldsymbol{V}_j - \mu_j^* \boldsymbol{G}, s)$.
   - The challenger $\mathcal{S}$ outputs the signed data $\{\boldsymbol{U}_{ij}\}_{j \in [k]}$ and sends them to the adversary $\mathcal{A}$.

We show that the public keys and signatures in the real scheme and in the simulation game are statistically indistinguishable. For the matrix $\boldsymbol{A}$, it is produced by the ***TrapGen*** algorithm in the real system and is chosen uniformly at random in the simulation game. For the matrix $\boldsymbol{B}$, it is chosen uniformly at random in the real scheme and $\boldsymbol{B}=\boldsymbol{AW}-\tau^*\boldsymbol{G}$ in the simulation game, where $\boldsymbol{W}$ is chosen uniformly at random. For each i, $\boldsymbol{V}_i$ is chosen uniformly at random in the real system and $\boldsymbol{V}_i=\boldsymbol{AW}_i$ is computed using uniformly random $\boldsymbol{W}_i$ in the simulation game. From Lemma 2, the public keys in the real scheme and in the simulation game are statistically indistinguishable. For the sufficient large Gaussian parameter s, the outputs of ***SampleLeft*** used in the real system and ***SampleRight*** used in the simulation are statistically indistinguishable.

If the adversary outputs a forgery $(\boldsymbol{U}^*,\mu^*)$ for the tag τ^* and the circuit C^*, we naturally have $h_{\boldsymbol{A}_{\tau^*}}(\boldsymbol{U}^*,\mu^*)=C^*(\boldsymbol{V}_1,\cdots,\boldsymbol{V}_k)$, i.e., $[\boldsymbol{A}\|\boldsymbol{AW}]\boldsymbol{U}^*+\mu^*\boldsymbol{G}=C^*(\boldsymbol{V}_1,\cdots,\boldsymbol{V}_k)$. Let $\boldsymbol{U}^*=[\boldsymbol{U}_1^*\|\boldsymbol{U}_2^*]^t$, we have

$$\boldsymbol{A}(\boldsymbol{U}_1^*+\boldsymbol{WU}_2^*)+\mu^*\boldsymbol{G}=C^*(\boldsymbol{V}_1,\cdots,\boldsymbol{V}_k)\,. \tag{19}$$

Equivalently,

$$h_{\boldsymbol{A}}(\boldsymbol{U}_1^*+\boldsymbol{WU}_2^*,\mu^*)=C^*(\boldsymbol{V}_1,\cdots,\boldsymbol{V}_k)\,. \tag{20}$$

From Theorem 2, we can see that the challenger $\mathcal{S}$ can compute a matrix $\boldsymbol{U}^{C^*}\in\mathbf{Z}_q^{m\times m}$ and an integer $x\in\mathbf{Z}_q$ so that $C^*(\boldsymbol{V}_1,\cdots,\boldsymbol{V}_k)=\boldsymbol{AU}^{C^*}+x\boldsymbol{G}$. In other words, $h_{\boldsymbol{A}}(\boldsymbol{U}^{C^*},x)=C^*(\boldsymbol{V}_1,\cdots,\boldsymbol{V}_k)$. Therefore, we have $h_{\boldsymbol{A}}(\boldsymbol{U}^{C^*},x)=h_{\boldsymbol{A}}(\boldsymbol{U}_1^*+\boldsymbol{WU}_2^*,\mu^*)$. In the simulation game, all queried signatures are produced independently through ***SampleRight*** algorithm. The adversary $\mathcal{A}$ does not query signatures of all the messages with the tag τ^*. Thus, $\mathcal{A}$ gets no information about $\boldsymbol{U}^{C^*}$. The probability that $\boldsymbol{U}_1^*+\boldsymbol{WU}_2^*-\boldsymbol{U}^{C^*}=\boldsymbol{0}$ can be negligible. From the above analysis, the challenger finds a collision for the fully homomorphic chameleon function $h_{\boldsymbol{A}}$ with the advantage

$$\boldsymbol{Adv}_{HCHF}(\Sigma^{\mathrm{A}})\geq\boldsymbol{Adv}_{\Pi}^{selective}(\mathrm{A})-negl(n). \tag{21}$$

2 Next, we consider the other type of forgers:

- The challenger $\mathcal{S}$ generates a public key for the adversary $\mathcal{A}$. $\mathcal{S}$ first chooses the public parameters n,q,m,s,B which are the same as above. Then $\mathcal{S}$ randomly samples $\boldsymbol{U}_i^t\leftarrow(D_{\mathbf{Z}^m,w(\sqrt{\log m})})^{2m}$ and chooses $\boldsymbol{W}\in\{-1,1\}^{m\times m}$. Last, $\mathcal{S}$ computes $\boldsymbol{V}_i=[\boldsymbol{A}\|\boldsymbol{AW}]\boldsymbol{U}_i+\mu_i^*\boldsymbol{G}\bmod q$. Moreover, let $\boldsymbol{B}=\boldsymbol{AW}-\tau^*\boldsymbol{G}\bmod q$. After that, the challenger $\mathcal{S}$ outputs the public key $(\boldsymbol{A},\boldsymbol{B},\boldsymbol{G},\{\boldsymbol{V}_i\}_{i\in[k]})$.
- The challenger $\mathcal{S}$ generates signatures for the queried messages and the tag τ_i. If $\tau_i\neq\tau^*$, the challenger aborts the game. Otherwise, $\mathcal{S}$ straightforwardly outputs the signatures $\{\boldsymbol{U}_i\}_{i\in[k]}$ for the challenged tag.

Obviously, the challenger does not abort the game with probability $1/Q$. Similarly to the above analysis, we can also find that the public keys and signatures in the real scheme and in the simulation game are statistically indistinguishable.

If the adversary outputs a forgery $(\boldsymbol{U}^*,\mu^*)$ for the tag τ^* and the circuit C^*, we naturally have

$$[\boldsymbol{A}\|\boldsymbol{AW}]\boldsymbol{U}^*+\mu^*\boldsymbol{G}=C^*(\boldsymbol{V}_1,\cdots,\boldsymbol{V}_k)\,. \tag{22}$$

Letting $\boldsymbol{U}^*=[\boldsymbol{U}_1^*\|\boldsymbol{U}_2^*]^t$, we can obtain

$$h_{\boldsymbol{A}}(\boldsymbol{U}_1^*+\boldsymbol{WU}_2^*,\mu^*)=C^*(\boldsymbol{V}_1,\cdots,\boldsymbol{V}_k)\,. \tag{23}$$

On the other hand, the adversary has the collection of signatures $\{\boldsymbol{U}_i\}_{i\in[k]}$ for the challenged message vector $\boldsymbol{\mu}^*$. Therefore, the challenger $\mathcal{S}$ can compute the evaluated signature $\boldsymbol{U}^{C^*}$ using the ***Eval*** algorithm. Namely,

$$[\boldsymbol{A}\|\boldsymbol{AW}]\boldsymbol{U}^{C^*}+C^*(\boldsymbol{\mu}^*)\boldsymbol{G}=C^*(\boldsymbol{V}_1,\cdots,\boldsymbol{V}_k)\,. \tag{24}$$

Letting $\boldsymbol{U}^{C^*}=[\boldsymbol{U}_1^{C^*}\|\boldsymbol{U}_2^{C^*}]^t$, we can also obtain

$$h_{\boldsymbol{A}}(\boldsymbol{U}_1^{C^*}+\boldsymbol{WU}_2^{C^*},C^*(\boldsymbol{\mu}^*))=C^*(\boldsymbol{V}_1,\cdots,\boldsymbol{V}_k)\,. \tag{25}$$

Hence, $h_{\boldsymbol{A}}(\boldsymbol{U}_1^*+\boldsymbol{WU}_2^*,\mu^*)=h_{\boldsymbol{A}}(\boldsymbol{U}_1^{C^*}+\boldsymbol{WU}_2^{C^*},C^*(\boldsymbol{\mu}^*))$. Since $C^*(\boldsymbol{\mu}^*)\neq\mu^*$, the adversary finds a collision for the randomized fully homomorphic chameleon function $h_{\boldsymbol{A}}$ with advantage

$$\boldsymbol{Adv}_{HCHF}(\Sigma^{\mathrm{A}})\geq\boldsymbol{Adv}_{\Pi}^{selective}(\mathrm{A})/Q-negl(n). \tag{26}$$

In table 1, the original and evaluated signatures represent the signatures generated by the ***Sign*** and ***Eval*** algorithm, respectively. “RO” is an abbreviation for “Random Oracle”, and similarly “ST” is an abbrevi-

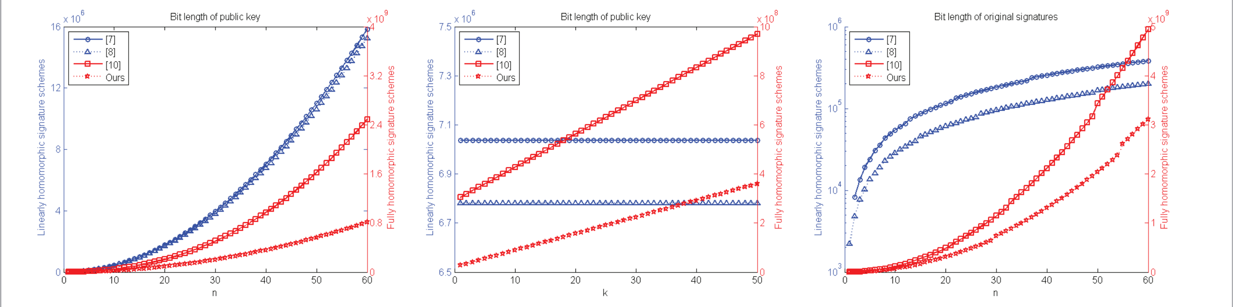
Table 1

Comparison between our scheme and some classical homomorphic signature schemes

Scheme	Bit length of the public key	Bit length of the private key	Bit length of original signatures	Bit length of evaluated signatures	Model	Permissible functions
[7]	$mn\log 2q$	$m^2\log(cn\log 2q)$	$2m\log(\sigma_1\sqrt{2m})$	$2m\log(k\sigma_1\sqrt{2m})$	RO	Linear
[8]	$\log(p+k+\sigma_2)+mn\log q$	$m^2\log(cn\log q)$	$m\log(\sigma_2\sqrt{m})$	$m\log(0.5kp\sigma_2\sqrt{m})$	RO	Linear
[10]	$(2k+3+\lambda)mn\log q$	$m^2\log(cn\log q)$	$2m^2\log(\sigma_3\sqrt{2m})$	$2m^2\log B_1$	ST	Any
Ours	$(k+3)mn\log q$	$m^2\log(cn\log q)$	$2m^2\log(\sigma_4\sqrt{2m})$	$2m^2\log B_2$	ST	Any

Figure 1

Comparison of the bit lengths of public/private key and original signatures



(a) $q=100000007, k=50, c=30$ (b) $q=100000007, n=40, c=30$ (c) $q=100000007, c=30$

ation for "Standard". The last column "permissible functions" means that the signature scheme can support the corresponding type of functions for homomorphic computation over signed data. Note that if some entries in Table 1 are non-integer, we should transform them into integers using the ceil function.

Efficiency

In this section, we consider the efficiency of our scheme by comparing it with some existing classical homomorphic signature schemes in terms of the bit length of the public/private key size, the bit length of signatures, the security model and permissible functions for homomorphic computation. Table 1 shows the specific comparison results. In [7], Boneh and Freeman presented a linearly homomorphic signature scheme that can authenticate vectors defined over binary fields. In order to generate the private key, they adopted the method introduced in [5], which can generate short bases of hard random lattices. Suppose that the generated trapdoor short basis (private key) is $\boldsymbol{T_A}$. It has been shown that $\boldsymbol{T_A} \le O(n\log q)$ [5,7]. Thus in our table, c is a constant so that $\boldsymbol{T_A} \le cn\log q$. According to their construction, the parameter m and the Gaussian parameter σ_1 are set equal to $\lceil 6n\log q \rceil$ and $c\sqrt{n\log 2q}\,w(\sqrt{\log n})$, respectively. In the same year, they proposed another linearly homomorphic signature scheme in section 4 of [8], which can authenticate any linear function of signed vectors defined over small fields $\mathbb{F}_p$. In their scheme, p and q are two primes so that $q \ge (nkp)^2$. For convenience, we denote $\sigma_2 = p\log m\sqrt{m\log q}$ in Table 1. In 2014, Boyen et al. proposed an adaptively secure homomorphic signature scheme that can evaluate any circuit over signed data [10]. In their scheme, the Gaussian parameter $\sigma_3 = w(m\log q\sqrt{\log m})$ and the upper bound of the size of evaluated signatures $B_1 = w(2^d)$, where d is the maximum depth of the circuits. According to section 4.2, the Gaussian parameter σ_4 in our scheme is equal to $O(\sqrt{n\log q})w(\sqrt{\log m})$, and the upper bound $B_2 = 2^{dw(\log\lambda)}$. In order to achieve the same security level, all the above-mentioned homomorphic signature schemes adopt the same parameters when performing the ***TrapGen*** algorithm [5]. That is to say, the comparison is fair.

Note that in Table 1, the first two signature schemes [7-8] are linearly homomorphic in the random oracle model and the latter two ones ([10] and ours) are fully homomorphic in the standard model. Nevertheless, the comparison result shows that the bit lengths of the private keys are almost exactly the same. Unfortunately, the bit lengths of evaluated signatures in fully homomorphic schemes are larger than those in linearly homomorphic schemes. However, the bit length of evaluated signatures in [10] is almost the same as that in our scheme. Next, we compare the public key size and the size of the original signatures from an experimental point of view. In [8], the scheme requires two primes p and q. Thus in our experiments, we choose two specific primes $p = 2$ and $q = 100000007$ which can meet their requirements. The dimension of random lattices m and the specific constant c are set equal to $\lceil 6n\log q\rceil$ and 30, respectively [7-8]. We set $\sigma_1 = c\sqrt{n\log 2q}\log n$, $\sigma_3 = m\log q\log m$, and $\sigma_4 = \sqrt{n\log q}\log m$. In Fig. 1(a) and 1(b), we investigate the bit length of the public key in terms of the parameter n and the maximum size of the dataset k, respectively. Note that we set the security parameter λ in [10] to n. In Fig. 1(c), we investigate the bit length of original signatures in terms of n. Evidently, the experimental results imply that the public key size and the size of original signatures in our scheme are smaller than those in [10]. Simultaneously, the public key size and the size of original signatures in our fully homomorphic signature scheme are larger than those in these two linearly homomorphic signature scheme [7-8]. It is acceptable because fully homomorphic signatures can support any homomorphic computation over signed data, rather than linear homomorphic computation. This may be a compromise between the functionality and efficiency.

Conclusions

In this paper, we first construct a type of HCHFs based on the SIS problem in hard random lattices. Then we use this type of HCHFs to construct fully homomorphic signature schemes for poly-depth circuits. Our construction has many advantages compared to previous works on this study. It is secure in the standard model and the public parameters grow linearly in the size of input circuit. The public key size and the bit length of original signatures of our scheme are smaller than those of the classical fully homomorphic signature scheme [10]. Our future work mainly focuses on designing fully homomorphic signature schemes with constant-size public keys. From a security perspective, the security parameter of the SIS problem in our scheme is $\beta = \sqrt{m}(2mB+1) = O(m^{1.5}2^{dw(\log\lambda)})$. In fact, the size of the evaluated signatures B affects the security of our scheme. Another open problem is to construct fully homomorphic signature schemes in which the size of evaluated signatures is smaller than that in ours.

Acknowledgments

The authors gratefully thank the reviewers for their valuable comments. This paper is supported by the National Key Research and Development Program of China (Grant no. 2016YFB0800602), the National Natural Science Foundation of China (Grant nos. 61573067, 61472045), and the Beijing City Board of Education Science and technology key project (Grant no. KZ201510015015), the Beijing City Board of Education Science and technology project (Grant no. KM 201510015009).

References

1. Agrawal, S., Boneh, D., Boyen, X. Efficient lattice (H)IBE in the standard model. Advances in Cryptology-EUROCRYPT, 2010, 553-572. https://doi.org/10.1007/978-3-642-13190-5_28
2. Agrawal, S., Boyen, X., Vaikuntanathan, V., Voulgaris, P., Wee, H. Functional encryption for threshold functions (or fuzzy ibe) from lattices. Public Key Cryptography. Springer, Berlin-Heidelberg, 2012, 280-297.

3. Agrawal, S., Freeman, D. M., Vaikuntanathan, V. Functional encryption for inner product predicates from learning with errors. Advances in Cryptology-ASIACRYPT. Springer, Berlin-Heidelberg, 2011, 21-40.

4. Ajtai, M. Generating hard instances of lattice problems. Extended abstracts of the Proceedings of the 28 annual ACM symposium on Theory of computing, 1988.

5. Alwen, J., Peikert, C. Generating shorter bases for hard random lattices. Theory of Computing Systems, 2011, 48, 535-553. https://doi.org/10.1007/s00224-010-9278-3

6. Barbosa, M., Farshim, P. On the semantic security of functional encryption schemes. Public Key Cryptography. Springer, Berlin-Heidelberg, 2013, 143-161.

7. Boneh, D., Freeman, D. M. Linearly homomorphic signatures over binary fields and new tools for latticebased signatures. Public Key Cryptography, volume 6571 of Lecture Notes in Computer Science. Springer, 2011, 1-16.

8. Boneh, D., Freeman, D. M. Homomorphic signatures for polynomial functions. Advances in Cryptology- EUROCRYPT, 2011, 149-168. https://doi.org/10.1007/978-3-642-20465-4_10

9. Boyen, X. Attribute-based functional encryption on lattices. Theory of Cryptography. Springer, Berlin-Heidelberg, 2013, 122-142.

10. Boyen, X., Fan, X., Shi, E. Adaptively Secure Fully Homomorphic Signatures Based on Lattices. IACR Cryptology ePrint Archive, 2014, 916.

11. Brakerski, Z., Gentry, C., Vaikuntanathan, V. (Leveled) fully homomorphic encryption without bootstrapping. Proceedings of the 3rd Innovations in Theoretical Computer Science Conference, ACM, 2012, 309-325. https://doi.org/10.1145/2090236.2090262

12. Brassard, G., Chaum, D., Crépeau, C. Minimum disclosure proofs of knowledge. Journal of Computer and System Sciences, 1988, 37, 156-189. https://doi.org/10.1016/0022-0000(88)90005-0

13. Cash, D., Hofheinz, D., Kiltz, E., Peikert, C. Bonsai trees, or how to delegate a lattice basis. Journal of Cryptology, 2012, 25, 601-639. https://doi.org/10.1007/s00145-011-9105-2

14. Dodis, Y., Reyzin, L., Smith, A. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. Advances in Cryptology-Eurocrypt. Springer, Berlin-Heidelberg, 2004, 523-540.

15. Freeman, D. M. Improved security for linearly homomorphic signatures: A generic framework. In: International Workshop on Public Key Cryptography. Springer, Berlin-Heidelberg, 2012, 697-714. https://doi.org/10.1007/978-3-642-30057-8_41

16. Gennaro, R., Katz, J., Krawczyk, H., Rabin T. Secure network coding over the integers. Public Key Cryptography. Springer, 2010, 142-160.

17. Gentry, C. Fully homomorphic encryption using ideal lattices. Proceedings of the 41 Annual ACM Symposium on Theory of Computing, ACM, 2009, 169-178. https://doi.org/10.1145/1536414.1536440

18. Gentry, C., Halevi, S., Vaikuntanathan, V. i-hop homomorphic encryption and rerandomizable Yao circuits. Advances in Cryptology-CRYPTO. Springer, Berlin-Heidelberg, 2010, 155-172.

19. Gentry, C., Peikert, C., Vaikuntanathan, V. Trapdoors for hard lattices and new cryptographic constructions. Proceedings of the fortieth Annual ACM Symposium on Theory of Computing, ACM, 2008, 197-206. https://doi.org/10.1145/1374376.1374407

20. Krawczyk, H., Rabin, T. Chameleon hashing and signatures. IACR Cryptology ePrint Archive, 1998, 10.

21. Micciancio, D., Peikert, C. Trapdoors for lattices: Simpler, tighter, faster, smaller. Advances in Cryptology-EUROCRYPT. Springer, Berlin-Heidelberg, 2012, 700-718.

22. Micciancio, D., Regev, O. Worst-case to averagecase reductions based on Gaussian measures. SIAM Journal on Computing, 2007, 37(1), 267-302. http://epubs.siam.org/doi/abs/10.1137/S0097539705447360.

23. Mohassel, P. One-time signatures and chameleon hash functions. Selected Areas in Cryptography. Springer, Berlin-Heidelberg, 2011, 302-319. https://doi.org/10.1007/978-3-642-19574-7_21

24. Shpilka, A., Yehudayoff, A. Arithmetic circuits: A survey of recent results and open questions. Foundations and Trends in Theoretical Computer Science, 2010, 5, 207-388. https://doi.org/10.1561/0400000039

25. Van, D. M., Gentry, C., Halevi, S., Vaikuntanathan, V. Fully homomorphic encryption over the integers. Advances in Cryptology-EUROCRYPT. Springer, Berlin-Heidelberg, 2010, 24-43.

26. Xie, D., Peng, H., Li, L., Yang, Y. Efficient PostQuantum Secure Network Coding Signatures in the Standard Model. KSII Transactions on Internet and Information Systems, 2016, 10, 2427-2445.

Summary / Santrauka

Homomorphic signature schemes provide a feasible solution to the authenticity of computations on an untrusted server (e.g. cloud). In a homomorphic signature scheme, given a k -length message set $\mu = \{\mu_1, \mu_2, \cdots, \mu_k\}$ and its corresponding signed dataset $\delta = \{\delta_1, \delta_2, \cdots, \delta_k\}$, anyone can publicly perform homomorphic computations and produce a new signature $\delta^{'}$ for the messages $\mu^{'} = f(\mu_1, \mu_2, \cdots, \mu_k)$, where f is a function or a circuit. If the generated homomorphic signature $\delta^{'}$ is valid, then the owner of the dataset (e.g. cloud users) convinces that μ' is indeed the correct output of the function f over the original messages even if he/she forgets them. In this work, the main contribution is to build a bridge between the leveled Fully Homomorphic Signature Scheme (FHSS) and Homomorphic Chameleon Hash Function (HCHF), which is a new cryptographic primitive introduced by us based on prior works. We first present the definition and specific construction of HCHF and then use this forceful technique to construct leveled fully homomorphic signature schemes for any polynomial-depth circuit. In our standard model scheme, the size of evaluated homomorphic signature grows polynomially in the depth of the circuit. The security of our scheme is based on the property of collision resistance of HCHF, which can be reduced to the Small Integer Solution (SIS) in hard random lattices.

Homomorfinio parašo schemose pateikiamas galimas sprendimas nepatikimo serverio (pvz., debesies) apskaičiavimų autentiškumui nustatyti. Homomorfinio parašo schemoje, turint k-ilgio žinučių rinkinį $\mu = \{\mu_1, ..., \mu_k\}$ ir atitinkamą pasirašytą duomenų rinkinį $\delta = \{ \delta_1, ... , \delta_k\}$, bet kas gali viešai atlikti homomorfinius skaičiavimus ir sukurti naują parašą δ' žinutėms $\mu' = f\{ \mu_1, \mu_2, \mu_3, ... , \mu_k\}$; čia f – grandinės funkcija. Jei gautas homomorfinis parašas δ' yra validus, duomenų rinkinio savininkas (pvz., debesų vartotojas) įtikina, kad, palyginti su originaliomis žinutėmis (net jei apie jas pamirštama), μ' išties yra teisinga funkcijos f išeiga. Pagrindinis šio straipsnio indėlis – sukurti sąsają tarp išlygintos visiškai homomorfinės parašo sistemos (angl. *Fully Homomorphic Signature Scheme (FHSS)*) ir homomorfinės chameleoninės maišos funkcijos (angl. *Homomorphic Chameleon Hash Function (HCHF)*), kuri yra nauja kriptografinė bazė, autorių pristatyta remiantis jų ankstesniais darbais. Straipsnyje pirmiausia apibūdinama *HCHF* ir pateikiamas jos specifinio sudarymo mechanizmas, tada ši veržli technologija taikoma išlygintoms visiškai homomorfinėms parašo schemoms bet kokiai daugianarei gylio grandinei konstruoti. Standartinėje autorių modelio schemoje įvertintų homomorfinių parašų dydis daugianariškai auga grandinės gylyje. Schemos saugumas paremtas *HCHF* susidūrimo pasipriešinimo savybe, kuri gali būti sumažinta iki mažojo sveikojo skaičiaus sprendinio (angl. *Small Integer Solution (SIS)*) kietosiose atsitiktinėse gardelėse.