Adaptively Secure Fully Homomorphic Message Authentication Code with Pre-processable Verification

Jeongsu Kim and Aaram Yun

Ewha Womans University, Korea
jsk2357@gmail.com
aaramyun@ewha.ac.kr

Abstract. There has been remarkable progress in fully homomorphic encryption, ever since Gentry's first scheme. In contrast, fully homomorphic authentication primitives received relatively less attention, despite existence of some previous constructions. While there exist various schemes with different functionalities for fully homomorphic encryption, there are only a few options for fully homomorphic authentication. Moreover, there are even fewer options when considering two of the most important properties: adaptive security, and pre-processable verification. To our knowledge, except for some concurrent works, achieving both properties requires the use of nested construction, which involves homomorphically authenticating a homomorphic authentication tag of a message, making the scheme costly and complicated.
In this work, we propose a dedicated scheme for (leveled) fully homomorphic message authentication code that is adaptively secure and has pre-processable verification. Leveraging the secrecy of the primitive, we demonstrate that a slight modification of a selectively secure (leveled) fully homomorphic signature scheme yields an adaptively secure (leveled) fully homomorphic message authentication code with pre-processable verification. Additionally, we introduce a novel notion and generic transform to enhance the security of a homomorphic message authentication code, which also exploits the secrecy of the primitive.

Keywords: homomorphic message authentication code, homomorphic authenticator, adaptive security, pre-processable verification

1 Introduction

As cloud computing has made extensive growth over time, the security of data delegated to cloud servers has become an important issue as well. Followed by Gentry's first fully homomorphic encryption scheme, numerous schemes with different functionalities have been suggested, all of which provides the privacy of delegated data [20,10,9,19,13]. However, there has been relatively less works about the authenticity of the evaluation of delegated data. While there already exist schemes for fully homomorphic message authentication codes (HomMAC),

fully homomorphic authenticated encryption, and fully homomorphic signature, achieving some important properties has been challenging [17,25,21]. Among such properties, two of the most important are pre-processable verification and adaptive security.

Most fully homomorphic authentication schemes have verification algorithms with complexity proportional to the complexity of homomorphically evaluating a circuit. Therefore, in order to make use of fully homomorphic authentication, it is desirable to have pre-processable verification, which allows pre-processing procedure that only needs a circuit as an input, and the rest of the verification process is independent of the complexity of the circuit. Unfortunately, apart from some concurrent works, the only viable options for pre-processable verification are the variants of the first (leveled) fully homomorphic signature scheme of Gorbunov, Vaikuntanathan, and Wichs [21].

While it is easier to construct a fully homomorphic authentication scheme that is adaptively secure, the difficulty comes from achieving pre-processable verification as well. To our knowledge, aside from certain concurrent works, the only known way to achieve both properties is to use nested construction. More specifically, an authentication of a message m for such functionalities (informally) consists of a selectively secure homomorphic authentication tag, $\mathsf{HomAuth}(m)$, followed by an authentication of the previous tag, $\mathsf{HomAuth}(\mathsf{HomAuth}(m))$ [21]. Homomorphic evaluation of such scheme is much expensive as one should homomorphically evaluate homomorphic evaluation of a desired circuit.

1.1 Contribution and Application

Contribution. In this paper, we propose a new way to construct an adaptively secure (leveled) fully homomorphic authentication with pre-processable verification, without relying on the nested construction. Exploiting the secrecy of HomMAC, we construct a dedicated scheme for (leveled) fully homomorphic MAC that is adaptively secure and has pre-processable verification, based on the first (leveled) fully homomorphic signature scheme of Gorbunov, Vaikuntanathan, and Wichs [21]. Our scheme is secure against verification queries, and is composable, allowing one to homomorphically evaluate on already homomorphically evaluated authentication tags.

In addition, we allow an adversary to make a forgery associated with a circuit, some of whose inputs have not yet been determined (note: we use 'circuit' and 'inputs' for simplicity, but we actually mean 'labeled program' and 'input labels'). To address such a forgery, we introduce a novel notion called (pseudo-)ambiguity At a high level, a HomMAC is considered *ambiguous* when there exist exponentially many possible candidates for an authentication tag associated with a circuit with undetermined inputs. We call a HomMAC H is *pseudo-ambiguous* when there exists an ambiguous HomMAC H' that is indistinguishable from H. Using ambiguity of HomMAC, we can statistically get rid of the possibility of an adversary to make a forgery associated with a circuit with undetermined inputs. Pseudo-ambiguity also provides protection against such forgeries as well.

We also provide a novel generic construction that turns any HomMAC into a pseudo-ambiguous one, using a pseudo-random function.

Application. As an application of our scheme, we may consider secure outsourcing of both storage and computation. A client, Alice, can authenticate her data, send various pieces of data to the server together with authentication tags, and erase local copies to save local storage. Alice can then ask the server to compute a function on her data for her, receive the computed value together with the authentication tag, and verify the correctness of the result.

As our scheme supports pre-processable verification, Alice can pre-process computationally expensive part of the verification algorithm with respect to a circuit of her choice. It can be done offline, since there is no communication needed between Alice and the server. The result of the process is a short secret key associated with the circuit, where the size of the resulting secret key is independent of the number of the inputs to the circuit. By storing the resulting key, Alice can use it in the future to efficiently verify the computation of the circuit she has chosen. After pre-processing, Alice only needs to be able to compute the rest of verification algorithm, which is independent of the complexity of evaluating the circuit, and the authentication algorithm.

Due to the adaptive security of our scheme, Alice can send data and its authentication tag to the server as soon as she gains access to the data. Combined with pre-processable verification, our scheme is especially useful for delegation of computation of time-sensitive data, such as statistics of votes or stock indices. Alice can pre-process the verification during idle periods, and verify delegated computation very efficiently later.

In addition, as we consider a stronger version of the security of HomMAC, as SUF-CMA is to UF-CMA, we can use our scheme to construct a (leveled) fully homomorphic authenticated encryption in encrypt-then-authenticate manner [25]. Moreover, using our scheme provides tighter security than the first (leveled) fully homomorphic authenticated encryption scheme of Kim and Yun because they used selectively secure (leveled) fully homomorphic MAC to deal with an adaptive adversary [25].

1.2 Related Works

Security Notion of Fully Homomorphic Authentications. Catalano, Fiore, and Nizzardo suggested a security notion for fully homomorphic signatures where an adversary can create a forgery associated to a circuit, some of whose inputs have not yet been determined [11]. Kim and Yun adopted this notion to homomorphic authenticated encryption and established the redundancy of a verification query in symmetric homomorphic primitives [25].

The First Schemes for Fully Homomorphic Authentication Primitives and Their Variants. Gennaro and Wichs proposed the first fully homomorphic MAC scheme using fully homomorphic encryption [17]. However, their scheme

was only secure against adversaries that cannot make verification queries, and did not support pre-processable verification. Later, Gorbunov, Vaikuntanathan, and Wichs introduced the first (leveled) fully homomorphic signature scheme with pre-processable verification. They have started with a selectively secure scheme, and then generically converted such a scheme into an adaptively secure one. The generic transform resembles chameleon hashing technique, and the resultant signature (informally) consists of a selectively secure homomorphic signature $\mathsf{HomSign}(m)$ followed by a homomorphic signature of the previous signature $\mathsf{HomSign}(\mathsf{HomSign}(m))$.

Since the introduction of the first (leveled) fully homomorphic signature scheme, a few variants have been proposed. Wang, Wang, Li, and Gao proposed an identity-based (leveled) fully homomorphic signature scheme using trapdoor delegation technique of Micciancio and Peikert [33,28]. Their base scheme was selectively secure, and provided strong unforgeability. Boyen, Fan, and Shi suggested an adaptively secure (leveled) fully homomorphic signature scheme, but their signature was not compact because its size depended on the number of the inputs of the evaluated circuit [7]. There have been other adaptively secure variants, but most of them could not preserve pre-processable verification [27,32,34]

Using the encrypt-then-authenticate paradigm, a well-known nested construction, Kim and Yun proposed the first (leveled) fully homomorphic authenticated encryption scheme [25]. They have presented how to generically construct a (leveled) fully homomorphic authenticated encryption using an adaptively secure, fully homomorphic encryption and a selectively secure (leveled) fully homomorphic MAC. Their scheme provided pre-processable verification as well.

Homomorphic Signatures from Functional Commitments. In recent years, there have been significant improvements in designing fully homomorphic signatures from functional commitments. Catalano, Fiore, and Tucker proposed a multi-input homomorphic signature for constant-depth circuits based on an additive-homomorphic functional commitment [12]. Balbás, Catalano, Fiore, and Lai introduced the chainable functional commitment and proposed a homomorphic signature for circuits with polynomial width and depth based on it [6]. Wee and Wu further advanced this approach, constructing a homomorphic signature for circuits with polynomial width (and potentially unbounded depth) [35]. While these remarkable works advance the field, our focus lies in (leveled) fully homomorphic authentications, which allow homomorphic evaluation of circuits with polynomial depth (and potentially exponential width).

Homomorphic Signatures from Indistinguishability Obfuscation. Gay and Ursu proposed the first unleveled fully homomorphic signature from falsifiable assumptions [16]. Their construction is based on indistinguishability obfuscation (iO), unleveled fully homomorphic encryption, and non-interactive zero-knowledge proof system (NIZK). While their work enables a wide range of applications, we aim to develop a *dedicated* scheme for fully homomorphic MAC with adaptive security and pre-processable verification.

Attribute-Based Signatures. Tsabary proved the equivalence between attribute-based signatures and homomorphic signatures, and proposed new schemes based on lattice trapdoors [31]. Tsabary also used nested construction to achieve adaptive security as the first (leveled) fully homomorphic signature scheme [21]. Through Tsabarys work, attribute-based signatures could be translated into homomorphic signatures. Such signatures support circuits with polynomial width [29,15,30,14,26] or based on succinct non-interactive arguments of knowledge (SNARKs) [8], which is not yet known to be based on standard assumptions.

Concurrent Works. Until recently, options for fully homomorphic signatures were limited, but significant advances have emerged. Goyal introduced mutable batch arguments (mutable BARGs) to construct (leveled) fully homomorphic signatures from standard assumptions [22]. While this signature could initially be composed only a constant number of times, Afshar, Cheng, and Goyal later improved the result so that it could be composed a polynomial number of times [1]. Anthoine, Balbás, and Fiore proposed a fully-succinct multi-key (leveled) fully homomorphic signatures from standard assumptions by combining BARGs and functional commitments, while providing chained-composability (i.e., you cannot compose two or more signatures) [5]. Hayashi, Sakai, and Yamada proposed the first attribute-based signature for unbounded circuits with optimal parameter size from standard assumptions, which could be translated into a fully homomorphic signature [23]. While these signatures are excellent choices for fully homomorphic MACs, our scheme is the only dedicated algorithm for adaptive security and pre-processability, which brings simplicity. Also, our scheme does not rely on common reference string (CRS), and is composable without any limitation.

1.3 Technical Overview

In this section, we provide a high-level overview of our base construction. First, we briefly review the construction and security of the first (leveled) fully homomorphic signature [21]. We then identify a key reason why this construction achieves only selective security. Finally, we describe how it is modified to obtain an adaptively secure fully homomorphic MAC.

The First (Leveled) Fully Homomorphic Signatures. The base scheme of the first (leveled) fully homomorphic signature is a lattice-based signature scheme that follows hash-and-sign paradigm. KeyGen generates random matrices $\mathbf{V}_1, \ldots, \mathbf{V}_N \in \mathbb{Z}^{k \times l}$, and a trapdoor matrix $\mathbf{A} \in \mathbb{Z}^{n \times k}$ with its trapdoor $\mathbf{R}$ that satisfies following properties:

1. $\mathbf{A} \overset{\text{stat}}{\approx} \mathbf{A}'$ for a randomly chosen matrix $\mathbf{A}' \overset{\$}{\leftarrow} \mathbb{Z}^{n \times k}$.
2. For a random $\mathbf{V} \overset{\$}{\leftarrow} \mathbb{Z}^{n \times l}$, there exists an algorithm that can generate a matrix $\mathbf{U} \in \mathbb{Z}^{k \times l}$ with small entries such that $\mathbf{AU} = \mathbf{V}$ using the trapdoor $\mathbf{R}$. There also exists a distribution $\mathcal{U}$ over $\mathbb{Z}^{k \times l}$ such that if $\mathbf{U}' \leftarrow \mathcal{U}$ and $\mathbf{V}' := \mathbf{AU}'$, then $(\mathbf{A}, \mathbf{R}, \mathbf{U}, \mathbf{V}) \overset{\text{stat}}{\approx} (\mathbf{A}, \mathbf{R}, \mathbf{U}', \mathbf{V}')$.

KeyGen outputs $\mathbf{V}_1, \ldots, \mathbf{V}_N$, and $\mathbf{A}$ as public parameters, and keeps the trapdoor $\mathbf{R}$ secret. $\mathsf{Sign}(m_1, \ldots, m_N)$ generates the preimages $\mathbf{U}_1, \ldots, \mathbf{U}_N$ of $\mathbf{V}_1 - m_1\mathbf{G}, \ldots, \mathbf{V}_N - m_N\mathbf{G}$ using the trapdoor $\mathbf{R}$ for some gadget matrix $\mathbf{G} \in \mathbb{Z}^{n \times l}$ so that $\mathbf{A}\mathbf{U}_i + m_i\mathbf{G} = \mathbf{V}_i$ for $i = 1, \ldots, N$. $\mathsf{Sign}(m_1, \ldots, m_N)$ then outputs $\mathbf{U}_1, \ldots, \mathbf{U}_N$ as the signatures of $m_1, \ldots, m_N \in \{0, 1\}$. $\mathsf{Verify}(f, m, \mathbf{U}_m)$ deterministically generates $\mathbf{V}_f$ from $\mathbf{V}_1, \ldots, \mathbf{V}_N$, and outputs 1 if and only if $\mathbf{A}\mathbf{U}_m + m\mathbf{G} = \mathbf{V}_f$. Since the main focus of this overview is to describe how we achieved adaptive security, we omit the description of homomorphic evaluation.

The security of the first (leveled) fully homomorphic signature scheme comes from the hard lattice problem called short integer solution (SIS) problem. For any adversary $\mathcal{A}$ that makes a forgery against the above scheme, there exists an algorithm $\mathcal{B}$ that solves SIS problem by running $\mathcal{A}$ internally. $\mathcal{B}$ operates as follows:

1. Receive $\mathbf{A} \xleftarrow{\$} \mathbb{Z}^{n \times k}$
2. Run $\mathcal{A}$, and $\mathcal{A}$ makes a selective query $(m_1, \ldots, m_N)$
3. Sample $\mathbf{U}_1, \ldots, \mathbf{U}_N \leftarrow \mathcal{U}$, and set $\mathbf{V}_i := \mathbf{A}\mathbf{U}_i + m_i\mathbf{G}$ for $i = 1, \ldots, N$
4. Send $\mathbf{U}_1, \ldots, \mathbf{U}_N$ to $\mathcal{A}$ as the response of the selective query.
5. $\mathcal{A}$ makes a forgery, and obtain two tuples $(m, \mathbf{U})$ and $(m', \mathbf{U}')$ such that $\mathsf{Verify}(f, m, \mathbf{U}) = \mathsf{Verify}(f, m', \mathbf{U}') = 1$
6. Make an SIS solution from $(m, \mathbf{U})$ and $(m', \mathbf{U}')$

Given two tuples $(m, \mathbf{U})$ and $(m', \mathbf{U}')$ such that $\mathbf{A}\mathbf{U} + m\mathbf{G} = \mathbf{A}\mathbf{U}' + m'\mathbf{G} = \mathbf{V}_f$ for some $\mathbf{V}_f \in \mathbb{Z}^{n \times l}$, one can generate an SIS solution for $\mathbf{A}$ [21,33]. If $\mathcal{B}$'s simulation for $\mathcal{A}$ is statistically close to a real response of the selective query, then $\mathcal{B}$'s success probability in solving SIS is nearly the same as $\mathcal{A}$'s probability of forging a signature. Fortunately, due to the second property of the trapdoor matrix, $\mathcal{B}$'s simulation is statistically indistinguishable from a real query response. Therefore, as long as the SIS problem is hard to solve, there does not exist an adversary $\mathcal{A}$ that can forge the signature with non-negligible probability.

Selective Security and Public Verification. A reason the signature scheme above is selectively secure is public verification. Public verification requires $\mathbf{V}_1, \ldots, \mathbf{V}_N$ to be public, preventing $\mathcal{B}$ from setting $\mathbf{V}_i$ as $\mathbf{A}\mathbf{U}_i + m_i\mathbf{G}$ for an adaptive signing query m_i. More specifically, to simulate the response of the ith signing query for an adaptive adversary $\mathcal{A}$, $\mathcal{B}$ must generate tuples $(0, \mathbf{U}_i)$ and $(1, \mathbf{U}'_i)$ such that $\mathbf{A}\mathbf{U}_i + 0 \cdot \mathbf{G} = \mathbf{A}\mathbf{U}'_i + 1 \cdot \mathbf{G} = \mathbf{V}_i$. However, generating such tuples implies that $\mathcal{B}$ can solve the SIS problem for $\mathbf{A}$ without $\mathcal{A}$. There have been some modifications to the scheme to satisfy adaptive security, but they cost pre-processable verification [27,32,34].

Adaptive Security from the Secrecy of HomMAC. In contrast, a fully homomorphic MAC does not require public verification. Keeping $\mathbf{V}_i$ secret prior to the ith adaptive query m_i of $\mathcal{A}$ allows $\mathcal{B}$ to set $\mathbf{V}_i := \mathbf{A}\mathbf{U}_i + m_i\mathbf{G}$, resulting adaptive security. Building on this idea, we modified the above signature

scheme into an adaptively secure fully homomorphic MAC while preserving preprocessable verification. At a high level, our scheme works as follows: KeyGen generates a random function $F : \mathbb{N} \to \mathbb{Z}^{k \times l}$ and a trapdoor matrix $\mathbf{A}$ with its trapdoor $\mathbf{R}$, outputs $\mathbf{A}$ as an evaluation key, and keeps the trapdoor $\mathbf{R}$ and F as secret keys. For the ith authentication query m, $\mathsf{Auth}(i, m)$ sets $\mathbf{V}_i := F(i)$, generates the preimage $\mathbf{U}_i$ of $\mathbf{V}_i - m_i\mathbf{G}$ so that $\mathbf{A}\mathbf{U}_i + m_i\mathbf{G} = \mathbf{V}_i$, and outputs $\mathbf{U}_i$. $\mathsf{Verify}(f, m, \mathbf{U}_m)$ deterministically generates $\mathbf{V}_f$ from F, and outputs 1 if and only if $\mathbf{A}\mathbf{U}_m + m\mathbf{G} = \mathbf{V}_f$. Homomorphic evaluation process is the same as the first (leveled) fully homomorphic scheme, which will be described in Section 5.

2 Preliminaries

Notations and Conventions. We write $(\tau, \cdot) \in S$ when there exists an x such that $(\tau, x) \in S$. Conversely, we write $(\tau, \cdot) \notin S$ when there is no x such that $(\tau, x) \in S$. This can be generalized to two or more coordinates, such as $(\cdot, x, \cdot) \in S$ and $(\tau, \cdot, \cdot) \notin S$. When a multivariate function $f : \mathcal{X} \times \mathcal{Y} \to \mathcal{Z}$ is given, we let $f(x, \cdot) : \mathcal{Y} \to \mathcal{Z}$ be a function such that $f(x, \cdot)(y) = f(x, y)$ for any $y \in \mathcal{Y}$. Similarly, when a multivariate algorithm $\mathsf{Alg}(x, y)$ is given, we let $\mathsf{Alg}(x, \cdot)$ be an algorithm with x hard-wired, that takes an input $y \in \mathcal{Y}$ and outputs $\mathsf{Alg}(x, y)$. We will use generalized versions of this notation for two or more inputs such as $f(\cdot, x, \cdot)$ and $\mathsf{Alg}(x, y, \cdot, \cdot)$. If an algorithm takes variable number of inputs, we will use the notation $\mathsf{Alg}(x, y, \dots)$.

We let $\mathbb{Z}_q$ be the ring of integers modulo q. We use integers in $(-q/2, q/2]$ to represent the elements in $\mathbb{Z}_q$. For a matrix $\mathbf{U} \in \mathbb{Z}_q^{n \times m}$, we write $\|\mathbf{U}\|_\infty \leq \beta$ if every entry in $\mathbf{U}$ lies in $[-\beta, \beta]$.

Let f be a function. We write $f(\lambda) = \mathsf{poly}(\lambda)$ if there exists a constant $C > 0$ such that $f(\lambda) = O(\lambda^C)$. In contrast, we write $f(\lambda) = \mathsf{negl}(\lambda)$ if $f(\lambda) = o(\lambda^{-C})$ for any constant C. We often say a function is negligible (or polynomial) when the function is negligible (or polynomial) with respect to the security parameter λ.

Circuit. We define a *circuit* to be a directed acyclic graph, where each vertex is a *gate*, with an associated operation. We assume that we have a dedicated output wire in the circuit. For simplicity, we assume that every gate has 1 or 2 indegree.

Labeled program. Since HomMAC needs to verify whether a computed function value is correct, without having the original inputs to the function, we need to be able to describe a function using labels of the input arguments. We will adopt the notion of the *labeled program* [17] to do that.

Let $\mathcal{M}$ and $\mathcal{T}$ be a message space and a label space, respectively. A *labeled program* is a tuple $P = (f, \tau_1, \dots, \tau_l)$ where f is a function from $\mathcal{M}^l$ to $\mathcal{M}$, and $\tau_i \in \mathcal{T}$ is the associated *label* of the ith input. If messages $m_1, \dots, m_l \in \mathcal{M}$ are associated with labels $\tau_1, \dots, \tau_l \in \mathcal{T}$, then calculating $P = (f, \tau_1, \dots, \tau_l)$ means

evaluating $f(m_1, \ldots, m_l)$. Also, we write $P(m_1, \ldots, m_l)$ to indicate the value $f(m_1, \ldots, m_l)$. A label could be considered as an address, an index, or even a 'metadata' of a message (generalizing the 'positional index' 1, 2, 3 of the function expression $f(m_1, m_2, m_3)$, for example), and one could assume that it is known to public.

We say a labeled program $P = (f, \tau_1, \ldots, \tau_l)$ is fully bound when there is a path from any internal wire (including input wires) to the output wire. For any labeled program P, we can divide P into two unique sub-programs, $\dot{P}$ and $\bar{P}$, that are not connected to each other such that $\dot{P}$ is fully bound and contains the output wire of P. We call $\dot{P}$ the *fully bound sub-program* of P. Note that the input wires of $\dot{P}$ are the only input wires of P that are needed to compute the output of P. For example, suppose $f(x, y, z, w)$ is a circuit that computes two circuits $f_1(x, y) = x + y$ and $f_2(z, w) = z + w$, and outputs $f_1(x, y)$. If we let $P := (f, \tau_1, \tau_2, \tau_3, \tau_4)$, then the fully bound sub-program of P is $\dot{P} := (f_1, \tau_1, \tau_2)$

Admissible program and function. Let H be a HomMAC. As we use labeled program to verify an authentication tag is correctly generated, we write a program P is an *admissible program* of H when the homomorphic evaluation of P is supported by H. We also define the space of admissible programs, denoted by $\mathcal{P}$, to be the set of labeled programs so that each element $P \in \mathcal{P}$ is supported by H.

3 Homomorphic Message Authentication Code

3.1 Syntax

We consider homomorphic message authentication code (HomMAC) defined as follows.

Definition 1 (HomMAC). *A homomorphic message authentication code is a tuple of probabilistic polynomial time algorithms* $(\mathsf{KeyGen}, \mathsf{Auth}, \mathsf{Eval}, \mathsf{Verify})$ *as follows:*

- $(sk, ek) \leftarrow \mathsf{KeyGen}(1^\lambda)$*: outputs a secret key* sk *and an evaluation key* ek *for a given security parameter* λ*.*
- $\sigma \leftarrow \mathsf{Auth}(sk, \tau, m)$*: outputs an authentication tag* $\sigma \in \Sigma$ *of a message* m *with respect to the label* τ*. This associates the message* m *to the label* τ*.*
- $\sigma_f \leftarrow \mathsf{Eval}(ek, f, (m_1, \sigma_1), \ldots, (m_l, \sigma_l))$*: deterministically outputs homomorphically evaluated authentication tag* σ_f*, given a function* f *of arity* l*, and message-tag tuples* $(m_1, \sigma_1), \ldots, (m_l, \sigma_l) \in \mathcal{M} \times \Sigma$*.*
- $b \leftarrow \mathsf{Verify}(sk, P, m, \sigma)$*: deterministically outputs acceptance bit* b *of an authentication tag* σ *so that* b *represents the verification that a message* m *is the output of the labeled program* $P \in \mathcal{P}$ *with respect to* sk*.*

We let $\mathcal{M}$, Σ, $\mathcal{T}$, $\mathcal{P}$ to be the message space, the authentication tag space, the label space, and the space of admissible programs of the HomMAC. For a

HomMAC H, we write $H.\mathsf{KeyGen}()$, $H.\mathsf{Auth}()$, $H.\mathsf{Eval}()$ and $H.\mathsf{Verify}()$ to denote its algorithms. We say a HomMAC is (leveled) fully homomorphic when it can homomorphically evaluate any depth d program, or equivalent version of such program for some $d = d(\lambda) = \mathsf{poly}(\lambda)$.

Correctness. A HomMAC has to satisfy the following correctness properties except for negligible probability:

- *Correctness of the evaluation*:

  $$\mathsf{Verify}(sk, (f, \tau_1, \ldots, \tau_l), f(m_1, \ldots, m_l), \mathsf{Eval}(ek, f, (m_1, \sigma_1), \ldots, (m_l, \sigma_l))) = 1$$

  where $(sk, ek) \leftarrow \mathsf{KeyGen}(1^\lambda)$, $\sigma_i \leftarrow \mathsf{Auth}(sk, \tau_i, m_i)$ for $i = 1, \ldots, l$, and $(f, \tau_1, \ldots, \tau_l) \in \mathcal{P}$.
- *Projection preservation*:

  $$\sigma_i = \mathsf{Eval}(ek, \pi_i, (m_1, \sigma_1), \ldots, (m_l, \sigma_l))$$

  where $(sk, ek) \leftarrow \mathsf{KeyGen}(1^\lambda)$, $\sigma_i \leftarrow \mathsf{Auth}(sk, \tau_i, m_i)$ for $i = 1, \ldots, l$, and π_i is the ith projection function among l tuples.

Note that the correctness properties above imply the correctness of authentication tag:

$$\mathsf{Verify}(sk, (\mathrm{id}, \tau), m, \mathsf{Auth}(sk, \tau, m)) = 1$$

when id is the identity function.

Compactness. We say a HomMAC is *compact* if $(sk, ek) \leftarrow \mathsf{KeyGen}(1^\lambda)$ for some security parameter λ and the sizes of the output of $\mathsf{Auth}(sk, \cdot, \cdot)$ and $\mathsf{Eval}(ek, \ldots)$ are bounded by a polynomial in λ which is determined independently from the inputs of the algorithms. We require a HomMAC to be compact, which implies the size of any authentication tag is independent of its homomorphically evaluated function f.

Pre-processable verification. We say a HomMAC supports *pre-processable verification* when there is two additional algorithms with the following properties:

- $sk_P \leftarrow \mathsf{Prep}(sk, P)$: deterministically outputs a secret key sk_P associated with given sk and a labeled program P.
- $b \leftarrow \mathsf{EffVerify}(sk_P, m, \sigma)$: deterministically outputs acceptance bit b of an authentication tag σ so that b represents the verification that a message m is the output of labeled program $P \in \mathcal{P}$ using sk_P.

These algorithms must satisfy the following properties:

- *Correctness*: $\mathsf{EffVerify}(sk_P, m, \sigma) = \mathsf{Verify}(sk, P, m, \sigma)$ for any $m \in \mathcal{M}$, $\sigma \in \Sigma$ and labeled program P when $(sk, ek) \leftarrow \mathsf{KeyGen}(1^\lambda)$ and $sk_P \leftarrow \mathsf{Prep}(sk, P)$
- *Pre-processability*: if we let $t(l)$ to be the time required to compute an arity l labeled program P, then the time required to compute $\mathsf{EffVerify}(sk_P, \cdot, \cdot)$ is $o(t(l))$ when $(sk, ek) \leftarrow \mathsf{KeyGen}(1^\lambda)$ and $sk_P \leftarrow \mathsf{Prep}(sk, P)$

3.2 Security notion

We consider a strong variant of a security notion of a HomMAC. Our security notion is similar to the one that Gennaro and Wichs suggested, except that ours is stronger, as SUF-CMA is compared to EUF-CMA [17]. Moreover, our security game allows *non-redundant* verification queries as a definition of a security game of a homomorphic authenticated encryption of Kim and Yun [25]. The definitions of a non-redundant verification query and a *forgery* are given after the description of our security notion.

$\mathrm{Game}^{\mathsf{HomMAC}}_{H,\mathcal{A}}(\lambda)$:

Initialization. A keypair $(sk, ek) \leftarrow \mathsf{KeyGen}(1^\lambda)$ is generated, and the evaluation key ek is sent to $\mathcal{A}$. The challenger initializes the query history as a set $S = \emptyset$

Queries. $\mathcal{A}$ makes authentication and verification queries adaptively. The challenger responds to the queries as follows:

- On an authentication query (τ, m), if $(\tau, \cdot, \cdot) \notin S$, then the challenger sends an authentication tag $\sigma \leftarrow H.\mathsf{Auth}(sk, \tau, m)$, and updates $S \leftarrow S \cup \{(\tau, m, \sigma)\}$. If $(\tau, \cdot, \cdot) \in S$, then the challenger rejects the query.
- On a verification query (P, m, σ), if the query is non-redundant (which we will define soon below) with respect to the query history S, then the challenger sends the result of $H.\mathsf{Verify}(sk, P, m, \sigma)$ to $\mathcal{A}$. If the query is redundant, then the challenger rejects the query.

Finalization. If $\mathcal{A}$ made at least one forgery (which we define soon below), then the game outputs 1. Otherwise, the game outputs 0.

We say a HomMAC is *secure* if the advantage

$$\mathbf{Adv}^{\mathsf{HomMAC}}_{H,\mathcal{A}}(\lambda) := \mathbf{Pr}[\mathrm{Game}^{\mathsf{HomMAC}}_{H,\mathcal{A}}(1^\lambda) = 1]$$

is negligible for any probabilistic polynomial time algorithm $\mathcal{A}$.

Redundant queries. Let (P, m, σ) be a verification query, and $\dot{P} = (f, \tau_1, \ldots, \tau_l)$ be the fully bound sub-program of P. The query (P, m, σ) is *redundant* with respect to a query history S when, for all $i \in \{1, \ldots, l\}$, there exist (unique) tuples $(m_i, \sigma_i) \in \mathcal{M} \times \Sigma$ such that $(\tau_i, m_i, \sigma_i) \in S$, $f(m_1, \ldots, m_l) = m$, and $H.\mathsf{Eval}(ek, f, (m_1, \sigma_1), \ldots, (m_l, \sigma_l)) = \sigma$. We say a verification query is *non-redundant* if it is not redundant. We often omit the query history and say (P, m, σ) is redundant (respectively, non-redundant) when the query history is clear.

Types of verification queries and forgeries. Let (P, m, σ) be a verification query, $\dot{P} := (f, \tau_1, \ldots, \tau_l)$ be the fully bound sub-program of P, and S be the query history. We can categorize types of non-redundant verification queries with respect to S as follows:

– Type I: There exists an $i \in \{1, \ldots, l\}$ such that $(\tau_i, \cdot, \cdot) \notin S$.
– Type II: For all $i \in \{1, \ldots, l\}$, $(\tau_i, m_i, \sigma_i) \in S$ for some (unique) $(m_i, \sigma_i) \in \mathcal{M} \times \Sigma$ and

$$(f(m_1, \ldots, m_l), H.\mathsf{Eval}(ek, f, (m_1, \sigma_1), \ldots, (m_l, \sigma_l))) \neq (m, \sigma)$$

We call a verification query (P, m, σ) a *forgery* if the query is non-redundant with respect to S, and $H.\mathsf{Verify}(P, m, \sigma) = 1$. We also apply the same criteria to categorize a forgery as Type I or Type II.

Note also that the notion of forgery defined here is essentially a homomorphic version of a weak forgery: it has to be valid, and either m is not correct, or σ is different from the canonical authentication tag $H.\mathsf{Eval}(ek, f, (m_1, \sigma_1), \ldots, (m_l, \sigma_l))$. Therefore, the security notion we have defined for HomMAC is a homomorphic version of strong unforgeability.

Type II secure HomMAC. Let $\mathrm{Game}^{\mathsf{HomMAC,II}}_{H,\mathcal{A}}(1^\lambda)$ be a security game modified from $\mathrm{Game}^{\mathsf{HomMAC}}_{H,\mathcal{A}}(1^\lambda)$ so that an adversary $\mathcal{A}$ is only allowed to make Type II verification queries. We say a HomMAC is Type II secure if the advantage

$$\mathbf{Adv}^{\mathsf{HomMAC,II}}_{H,\mathcal{A}}(\lambda) := \mathbf{Pr}[\mathrm{Game}^{\mathsf{HomMAC,II}}_{H,\mathcal{A}}(1^\lambda) = 1]$$

is negligible for any probabilistic polynomial time algorithm $\mathcal{A}$.

3.3 Ambiguous HomMAC

Ambiguity. We say a HomMAC H is *ambiguous* when the following holds. For any admissible program P and its fully bound sub-program $\dot{P} = (f, \tau_1, \ldots, \tau_l)$, let $\mathcal{I} \subsetneq \{1, \ldots, l\}$ be an arbitrary subset. Without loss of generality, we let $\mathcal{I} = \{1, \ldots, i_0\}$ where $i_0 < l$. For arbitrary messages $m_1, \ldots, m_l \in \mathcal{M}$ and signatures $\sigma_1, \ldots, \sigma_{i_0}$ such that $\sigma_i \leftarrow H.\mathsf{Auth}(sk, \tau_i, m_i)$ for all $i = 1, \ldots, i_0$ where $(sk, ek) \leftarrow \mathsf{KeyGen}(1^\lambda)$, and an arbitrary element $\sigma \in \Sigma$, the probability

$$\mathbf{Pr}\left[H.\mathsf{Eval}(ek, f, (m_1, \sigma_1), \ldots, (m_l, \sigma_l)) = \sigma \;\middle|\; \begin{array}{c} \forall j = i_0 + 1, \ldots, l, \\ \sigma_j \leftarrow H.\mathsf{Auth}(sk, \tau_j, m_j) \end{array}\right]$$

is less than or equal to $2^{-\lambda}$ with respect to λ.

Implication of ambiguous HomMAC is that if one of $\tau_1, \ldots, \tau_l$ has not been queried yet, then there are exponentially many candidates for the authentication tag of P.

Using ambiguity of a HomMAC, we can statistically get rid of Type I forgery.

Theorem 1. *Let H be a HomMAC. If H is ambiguous and Type II secure, then it also is a secure HomMAC.*

Proof. Let $\mathcal{A}$ be an adversary of H that makes at most Q_v verification queries. Let L be the maximum number of inputs of the fully bound sub-program $\dot{P}$ of P, for all verification queries (P, m, σ) that $\mathcal{A}$ makes. Our purpose of the proof

is to show that there is a probabilistic polynomial time algorithm $\mathcal{B}$ against $\mathrm{Game}^{\mathsf{HomMAC,II}}$ that makes at most $Q_v(L+2)$ verification queries such that

$$\mathbf{Adv}^{\mathsf{HomMAC}}_{H,\mathcal{A}}(\lambda) \leq \mathbf{Adv}^{\mathsf{HomMAC,II}}_{H,\mathcal{B}}(\lambda) + \mathsf{negl}(\lambda)$$

We first note that the only possible way for $\mathcal{A}$ to win the game is to make a forgery. Moreover, the event that $\mathcal{A}$ makes a forgery can be categorized into two events:

$$E_{\mathrm{I}} := \{\text{the first forgery that } \mathcal{A} \text{ makes is Type I forgery in } Game^{\mathsf{HomMAC}}_{H,\mathcal{A}}(1^\lambda)\}$$
$$E_{\mathrm{II}} := \{\text{the first forgery that } \mathcal{A} \text{ makes is Type II forgery in } Game^{\mathsf{HomMAC}}_{H,\mathcal{A}}(1^\lambda)\}$$

Therefore, we can write

$$\begin{aligned}\mathbf{Adv}^{\mathsf{HomMAC}}_{H,\mathcal{A}}(\lambda) &= \mathbf{Pr}\left[\mathcal{A} \text{ makes a forgery in } \mathrm{Game}^{\mathsf{HomMAC}}_{H,\mathcal{A}}(1^\lambda)\right] \\ &= \mathbf{Pr}[E_{\mathrm{I}}] + \mathbf{Pr}[E_{\mathrm{II}}]\end{aligned}$$

We now construct $\mathcal{B}$ against the challenger of $\mathrm{Game}^{\mathsf{HomMAC,II}}_{H,\mathcal{B}}(1^\lambda)$ that runs $\mathcal{A}$. At a high level, $\mathcal{B}$ converts the first forgery that $\mathcal{A}$ makes into $\mathcal{B}$'s own forgery. When $\mathcal{A}$ makes a Type II forgery, $\mathcal{B}$ just passes the forgery and wins the game. When $\mathcal{A}$ makes a Type I forgery, $\mathcal{B}$ queries for undefined input labels of the forgery. In this way, $\mathcal{A}$'s forgery becomes either a redundant query or a Type II forgery. The probability that the forgery becomes a redundant query is negligible from the ambiguity of H. Therefore, $\mathcal{B}$ can always convert $\mathcal{A}$'s first forgery into a Type II forgery except for negligible probability.

Initialization. $\mathcal{B}$ gets ek from the challenger, and send ek to $\mathcal{A}$. $\mathcal{B}$ initializes sets S_{Auth} and $S^{\mathrm{I}}_{\mathsf{Verify}}$ as $\emptyset$.

Queries. Except when $\mathcal{A}$ makes verification queries, $\mathcal{B}$ just passes the queries and its responses between $\mathcal{A}$ and the challenger while updating S_{Auth} the same way as the challenger updates its authentication history S in $\mathrm{Game}^{\mathsf{HomMAC,II}}_{H,\mathcal{B}}(1^\lambda)$. When $\mathcal{A}$ makes a verification query (P,m,σ), $\mathcal{B}$ checks if the query is redundant, Type I, or Type II using S_{Auth}. If (P,m,σ) is redundant, then $\mathcal{B}$ just rejects the query. If (P,m,σ) is Type I verification query, $\mathcal{B}$ first updates $S^{\mathrm{I}}_{\mathsf{Verify}} \leftarrow S^{\mathrm{I}}_{\mathsf{Verify}} \cup \{(P,m,\sigma)\}$, then responds 0 to $\mathcal{A}$. If (P,m,σ) is Type II verification query, $\mathcal{B}$ forwards the query to the challenger and passes the response of the challenger back to $\mathcal{A}$.

Additional Queries. When $\mathcal{A}$ finishes its queries, $\mathcal{B}$ makes at most Q_vL additional authentication queries (while updating S_{Auth} the same as the challenger's history S) such that every verification queries in $S^{\mathrm{I}}_{\mathsf{Verify}}$ becomes Type II or redundant with respect to S_{Auth}. After such additional queries, if there exists at least one redundant verification query in $S^{\mathrm{I}}_{\mathsf{Verify}}$ with respect to S_{Auth}, then $\mathcal{B}$ returns "Bad" and halts. Otherwise, $\mathcal{B}$ makes Type II verification queries in $S^{\mathrm{I}}_{\mathsf{Verify}}$ in same order as $\mathcal{A}$.

We can categorize the event that $\mathcal{A}$ makes a forgery against $\mathcal{B}$ as follows as well.

$$E_{\mathrm{I}}^{\mathcal{B}} := \{\text{the first forgery that } \mathcal{A} \text{ makes against } \mathcal{B} \text{ is Type I}\}$$
$$E_{\mathrm{II}}^{\mathcal{B}} := \{\text{the first forgery that } \mathcal{A} \text{ makes against } \mathcal{B} \text{ is Type II}\}.$$

Note that $\mathbf{Pr}[E_{\mathrm{I}}^{\mathcal{B}}] = \mathbf{Pr}[E_{\mathrm{I}}]$ and $\mathbf{Pr}[E_{\mathrm{II}}^{\mathcal{B}}] = \mathbf{Pr}[E_{\mathrm{II}}]$ because, in $\mathcal{A}$'s perspective, $\mathcal{B}$ is indistinguishable from the challenger of $\mathrm{Game}_{H,\mathcal{A}}^{\mathsf{HomMAC}}(1^\lambda)$ before $\mathcal{A}$'s first forgery.

Now, let Bad := {$\mathcal{B}$ outputs "Bad"}. From the fact that H is ambiguous, we know that $\mathbf{Pr}[\mathrm{Bad}] \leq \frac{Q_v}{2^\lambda}$.

On event $E_{\mathrm{II}}^{\mathcal{B}}$, $\mathcal{B}$ passes $\mathcal{A}$'s Type II forgery to the challenger, and thus makes Type II forgery. In other words,

$$\mathbf{Pr}\left[E_{\mathrm{II}}^{\mathcal{B}}\right] \leq \mathbf{Pr}\left[\mathcal{B} \text{ makes Type II forgery in } \textbf{Queries} \text{ phase}\right]$$

On event $E_{\mathrm{I}}^{\mathcal{B}} \cap \mathrm{Bad}^{\complement}$, $\mathcal{B}$ makes $\mathcal{A}$'s Type I forgery into a Type II forgery. In other words,

$$\begin{aligned}
\mathbf{Pr}\left[E_{\mathrm{I}}^{\mathcal{B}}\right] &= \mathbf{Pr}\left[E_{\mathrm{I}}^{\mathcal{B}} \cap \mathrm{Bad}^{\complement}\right] + \mathbf{Pr}\left[E_{\mathrm{I}}^{\mathcal{B}} \cap \mathrm{Bad}\right] \\
&\leq \mathbf{Pr}\left[E_{\mathrm{I}}^{\mathcal{B}} \cap \mathrm{Bad}^{\complement}\right] + \mathbf{Pr}\left[\mathrm{Bad}\right] \\
&\leq \mathbf{Pr}\left[E_{\mathrm{I}}^{\mathcal{B}} \cap \mathrm{Bad}^{\complement}\right] + \frac{Q_v}{2^\lambda} \\
&\leq \mathbf{Pr}\left[\mathcal{B} \text{ makes Type II forgery in } \textbf{Additional Queries} \text{ phase}\right] + \frac{Q_v}{2^\lambda}
\end{aligned}$$

Conclusively, we can write

$$\begin{aligned}
\mathbf{Adv}_{H,\mathcal{A}}^{\mathsf{HomMAC}}(\lambda) &\leq \mathbf{Pr}[E_{\mathrm{I}}] + \mathbf{Pr}[E_{\mathrm{II}}] \\
&= \mathbf{Pr}[E_{\mathrm{I}}^{\mathcal{B}}] + \mathbf{Pr}[E_{\mathrm{II}}^{\mathcal{B}}] \\
&\leq \mathbf{Pr}\left[\mathcal{B} \text{ makes Type II forgery in } \textbf{Queries} \text{ phase}\right] + \frac{Q_v}{2^\lambda} \\
&\quad + \mathbf{Pr}\left[\mathcal{B} \text{ makes Type II forgery in } \textbf{Additional Queries} \text{ phase}\right] \\
&\leq \mathbf{Adv}_{H,\mathcal{B}}^{\mathsf{HomMAC},\mathrm{II}}(\lambda) + \frac{Q_v}{2^\lambda}
\end{aligned}$$

□

Remark 1. We can use the same analogy to define an ambiguous homomorphic authenticated encryption. Moreover, we can also prove that if a homomorphic authenticated encryption is Type II secure and ambiguous, then it is also secure.

While we can construct an ambiguous HomMAC using a random oracle, it is more plausible to define and construct a *pseudo*-ambiguous scheme using a pseudo random function.

Definition 2. *We say a HomMAC H is* pseudo-ambiguous *if there exists an ambiguous HomMAC $\tilde{H}$ such that*

$$\left|\mathbf{Pr}\left[\mathrm{Game}^{\mathsf{HomMAC}}_{H,\mathcal{A}}(1^\lambda)=1\right]-\mathbf{Pr}\left[\mathrm{Game}^{\mathsf{HomMAC}}_{\tilde{H},\mathcal{A}}(1^\lambda)=1\right]\right|\leq \mathsf{negl}(\lambda)$$

for any probabilistic polynomial time algorithm $\mathcal{A}$.

Theorem 2. *Let H be a HomMAC. If H is pseudo-ambiguous and Type II secure, then it is also a secure HomMAC.*

Proof. Let $\tilde{H}$ be an ambiguous HomMAC such that

$$\left|\mathbf{Pr}\left[\mathrm{Game}^{\mathsf{HomMAC}}_{H,\mathcal{A}}(1^\lambda)=1\right]-\mathbf{Pr}\left[\mathrm{Game}^{\mathsf{HomMAC}}_{\tilde{H},\mathcal{A}}(1^\lambda)=1\right]\right|\leq \mathsf{negl}(\lambda)$$

for any probabilistic polynomial time algorithm $\mathcal{A}$. Using pseudo-ambiguity, we first prove that $\tilde{H}$ is also Type II secure. For an arbitrary probabilistic polynomial time algorithm $\mathcal{A}^*$ and an arbitrary HomMAC H^*, we later construct a probabilistic polynomial time algorithm $\mathcal{A}^*$ such that

$$\mathbf{Pr}\left[\mathrm{Game}^{\mathsf{HomMAC},\mathrm{II}}_{H^*,\mathcal{A}^*}(1^\lambda)=1\right]=\mathbf{Pr}\left[\mathrm{Game}^{\mathsf{HomMAC}}_{H^*,\mathcal{B}^*}(1^\lambda)=1\right].$$

Then, we can write that for any probabilistic algorithm $\tilde{\mathcal{A}}$, we can find a probabilistic polynomial time algorithm $\tilde{\mathcal{B}}$ such that

$$\begin{aligned}\mathbf{Pr}\left[\mathrm{Game}^{\mathsf{HomMAC},\mathrm{II}}_{H,\tilde{\mathcal{A}}}(1^\lambda)=1\right]&=\mathbf{Pr}\left[\mathrm{Game}^{\mathsf{HomMAC}}_{H,\tilde{\mathcal{B}}}(1^\lambda)=1\right]\\ \mathbf{Pr}\left[\mathrm{Game}^{\mathsf{HomMAC},\mathrm{II}}_{\tilde{H},\tilde{\mathcal{A}}}(1^\lambda)=1\right]&=\mathbf{Pr}\left[\mathrm{Game}^{\mathsf{HomMAC}}_{\tilde{H},\tilde{\mathcal{B}}}(1^\lambda)=1\right],\end{aligned}$$

and thus

$$\begin{aligned}&\mathbf{Pr}\left[\mathrm{Game}^{\mathsf{HomMAC},\mathrm{II}}_{\tilde{H},\tilde{\mathcal{A}}}(1^\lambda)=1\right]\\ \leq&\mathbf{Pr}\left[\mathrm{Game}^{\mathsf{HomMAC},\mathrm{II}}_{H,\tilde{\mathcal{A}}}(1^\lambda)=1\right]+\left|\mathbf{Pr}\left[\mathrm{Game}^{\mathsf{HomMAC},\mathrm{II}}_{H,\tilde{\mathcal{A}}}(1^\lambda)=1\right]-\mathbf{Pr}\left[\mathrm{Game}^{\mathsf{HomMAC},\mathrm{II}}_{\tilde{H},\tilde{\mathcal{A}}}(1^\lambda)=1\right]\right|\\ =&\mathbf{Pr}\left[\mathrm{Game}^{\mathsf{HomMAC},\mathrm{II}}_{H,\tilde{\mathcal{A}}}(1^\lambda)=1\right]+\left|\mathbf{Pr}\left[\mathrm{Game}^{\mathsf{HomMAC}}_{H,\tilde{\mathcal{B}}}(1^\lambda)=1\right]-\mathbf{Pr}\left[\mathrm{Game}^{\mathsf{HomMAC}}_{\tilde{H},\tilde{\mathcal{B}}}(1^\lambda)=1\right]\right|\\ \leq&\mathbf{Pr}\left[\mathrm{Game}^{\mathsf{HomMAC},\mathrm{II}}_{H,\tilde{\mathcal{A}}}(1^\lambda)=1\right]+\mathsf{negl}(\lambda),\end{aligned}$$

which implies that $\tilde{H}$ is Type II secure since H is Type II secure. From Theorem 1, $\tilde{H}$ becomes a secure HomMAC. Therefore, H is also a secure HomMAC.

All we have left to do is to construct $\mathcal{A}^*$ above. We construct $\mathcal{A}^*$ that runs $\mathcal{A}^*$ internally, and acts as the challenger of $\mathrm{Game}^{\mathsf{HomMAC},\mathrm{II}}_{H^*,\mathcal{A}^*}(1^\lambda)$ as follows:

Authentication Queries. $\mathcal{A}^*$ passes authentication queries and their responses between $\mathcal{A}^*$ and the challenger of $\mathrm{Game}^{\mathsf{HomMAC}}_{H^*,\mathcal{B}^*}(1^\lambda)$, while maintaining the query history S_{Auth} in the same way as the query history S of the challenger.

Verification Queries. When $\mathcal{A}^*$ makes a verification query, $\mathcal{A}^*$ uses S_{Auth} to check if the query is redundant, Type I, or Type II. If the query is redundant or Type I, then $\mathcal{A}^*$ rejects $\mathcal{A}^*$'s query. If the query is Type II, then $\mathcal{A}^*$ makes the same verification query to the challenger of $\mathrm{Game}^{\mathsf{HomMAC}}_{H^*,\mathcal{B}^*}(1^\lambda)$ and sends its response to $\mathcal{A}^*$.

From the construction of $\mathcal{A}^*$, in $\mathcal{A}^*$'s perspective, $\mathcal{A}^*$ acts exactly the same as the challenger of $Game^{\mathsf{HomMAC,II}}_{H^*,\mathcal{A}^*}(1^\lambda)$. Therefore, the following probabilities are the same:

$$\mathbf{Pr}\left[\mathrm{Game}^{\mathsf{HomMAC,II}}_{H^*,\mathcal{A}^*}(1^\lambda)=1\right]=\mathbf{Pr}\left[\mathrm{Game}^{\mathsf{HomMAC}}_{H^*,\mathcal{B}^*}(1^\lambda)=1\right].$$

□

Remark 2. Note that the notions and theorems of this section can also be applied to any other homomorphic authentication primitives. In other words, one can define ambiguity and pseudo-ambiguity for homomorphic authenticated encryption or homomorphic signature, and apply variants of Theorem 1 and Theorem 2 to enhance their security.

4 Generic Construction for Security Enhancement

We now suggest a generic construction that converts an ordinary HomMAC into a pseudo-ambiguous one using a pseudorandom function (PRF). Moreover, our generic construction preserves Type II security and pre-processable verification. Therefore, if we have a Type II secure (leveled) fully homomorphic MAC scheme with pre-processable verification, then we can make the scheme into a (fully) secure one. While there are existing generic constructions in [17,24] that utilizes hash tree method, our generic construction is novel and simpler.

Construction 1 *Let H' be a HomMAC, and $\mathcal{M}'$, Σ', $\mathcal{T}'$, $\mathcal{P}'$ be the message space, the authentication tag space, the label space, and the admissible program space of H', respectively. Let d be the maximum depth of the programs in H' and p be the smallest prime number that exceeds $\max(2^\lambda, 2^d)$. Let $\mathcal{M} := \mathcal{M}'$, $\Sigma := \Sigma' \times \mathbb{Z}_p$, $\mathcal{T} := \mathcal{T}'$, and $\mathcal{P} := \mathcal{P}'$ be the message space, the authentication tag space, the label space, and the admissible program space of a HomMAC H that will be defined below. Let $F : \{0,1\}^\lambda \times \mathcal{T} \to \mathbb{Z}_p$ be a secure PRF. Using H' and F, we define H as follows.*

- $H.\mathsf{KeyGen}(1^\lambda)$*: generate* $(sk', ek') \leftarrow H'.\mathsf{KeyGen}(1^\lambda)$ *and* $K \xleftarrow{\$} \{0,1\}^\lambda$*. Let* $sk := sk'\|K$*,* $ek := ek'$*, and output* (sk, ek)*.*
- $H.\mathsf{Auth}(sk, \tau, m)$*: parse* $sk = sk'\|K$ *and let* $\sigma' \leftarrow H'.\mathsf{Auth}(sk', \tau, m)$ *and* $r := F(K, \tau)$*. Output* $\sigma := (\sigma', r)$*.*
- $H.\mathsf{Eval}(ek, f, (m_1, \sigma_1), \ldots, (m_l, \sigma_l))$*: let* $ek' = ek$*, parse* $\sigma_i = (\sigma'_i, r_i)$ *for* $i = 1, \ldots, l$*. Compute* $\sigma'_f \leftarrow H'.\mathsf{Eval}(ek', f, (m_1, \sigma'_1), \ldots, (m_l, \sigma'_l))$ *and* $r := f^+(r_1, \ldots, r_l)$ *where* f^+ *is the circuit* f *that every binary gate is replaced to an addition gate over* $\mathbb{Z}_p$*, and every unary gate is replaced to an identity gate. Output* $\sigma_f := (\sigma'_f, r)$*.*

– $H.\mathsf{Verify}(sk, P, m, \sigma)$: *parse* $sk = sk' \| K$, $P = (f, \tau_1, \ldots, \tau_l)$, *and* $\sigma = (\sigma', r)$. *Let* $r_i := F(K, \tau_i)$. *If* $r = f^+(r_1, \ldots, r_l)$ *and* $1 = H'.\mathsf{Verify}(sk', P, m, \sigma')$, *then output 1. Otherwise, output 0.*

Remark 3. Construction 1 naturally satisfies correctness properties from the correctness of H' and the fact that f^+ becomes a projection if f is a projection.

Remark 4. If H' supports pre-processable verification, then H also supports pre-processable verification. We can define $H.\mathsf{Prep}$ and $H.\mathsf{EffVerify}$ as follows:

– $H.\mathsf{Prep}(sk, P)$: parse $sk = sk' \| K$ and $P = (f, \tau_1, \ldots, \tau_l)$. Let $r'_i := F(K, \tau_i)$, $r' := f^+(r'_1, \ldots, r'_l)$, and $sk'_p := H'.\mathsf{Prep}(sk, P)$. Output $sk_p := sk'_p \| r'$.
– $H.\mathsf{EffVerify}(sk_P, m, \sigma)$: parse $sk_P = sk'_P \| r'$ and $\sigma = (\sigma', r)$. If $r = r'$ and $1 = H'.\mathsf{EffVerify}(sk'_P, m, \sigma)$, then output 1. Otherwise, output 0.

Remark 5. We can know that an admissible program of H' is also admissible to H. Therefore, if H' is (leveled) fully homomorphic, then so is H.

Remark 6. Suppose $(P, m, \sigma = (\sigma', r))$ is a Type II forgery with respect to a query history S. Let $\dot{P} := (f, \tau_1, \ldots, \tau_l)$ be the fully bound sub-program of P. Then from the fact that (P, m, σ) is a Type II forgery, we can find (unique) $(m_i, \sigma_i) \in \mathcal{M} \times \Sigma$ such that $(\tau_i, m_i, \sigma_i) \in S$ for all $i = 1, \ldots, l$. If we let $(\sigma'', r') \leftarrow H.\mathsf{Eval}(ek, f, (m_1, \sigma_1), \ldots, (m_l, \sigma_l))$, then we know that $(\sigma', r) \neq (\sigma'', r')$ and $r = r' = f^+(F(K, \tau_1), \ldots, F(K, \tau_l))$, which implies $\sigma' \neq \sigma''$. Therefore, one can make a Type II forgery of H into a Type II forgery of H'. In other words, Type II security of H' implies Type II security of H.

Theorem 3. *If F is a secure pseudorandom function, then H is pseudo-ambiguous.*

Proof. We write an advantage of differentiating PRF F from a random function $F^* : \mathcal{T} \to \mathbb{Z}_p$ as $\mathbf{Adv}_F^{\mathsf{PRF}}(\lambda)$. Let H^* be the same as H except it uses a random function instead of $F(K, \cdot)$. $H^*.\mathsf{KeyGen}(1^\lambda)$ samples a random function $F^* : \mathcal{T} \to \mathbb{Z}_p$ and lets $sk := sk' \| F^*$, and $H^*.\mathsf{Auth}$ and $H^*.\mathsf{Verify}$ use $F^*(\cdot)$ instead of $F(K, \cdot)$. If there is an adversary$\mathcal{A}$ that behaves differently in $\mathrm{Game}_{H,\mathcal{A}}^{\mathsf{HomMAC}}(1^\lambda)$ compared to $\mathrm{Game}_{H^*,\mathcal{A}}^{\mathsf{HomMAC}}(1^\lambda)$, then one can use $\mathcal{A}$ to distinguish F from F^*. Therefore, we can write

$$\left|\mathbf{Pr}\left[\mathrm{Game}_{H,\mathcal{A}}^{\mathsf{HomMAC}}(1^\lambda) = 1\right] - \mathbf{Pr}\left[\mathrm{Game}_{H^*,\mathcal{A}}^{\mathsf{HomMAC}}(1^\lambda) = 1\right]\right| \leq \mathbf{Adv}_F^{\mathsf{PRF}}(\lambda) \leq \mathsf{negl}(\lambda)$$

from the security of the PRF F.

All we have left to do is showing that H^* is ambiguous. For any admissible program P and its fully bound sub-program $\dot{P} = (f, \tau_1, \ldots, \tau_l)$, let $I \subsetneq \{1, \ldots, l\}$ be an arbitrary subset. Without loss of generality, we let $I = \{1, \ldots, i_0\}$ where $i_0 < l$. For arbitrary messages $m_1, \ldots, m_l$ and authentication tags $\sigma_1, \ldots, \sigma_{i_0}$ such that

$$\forall i = 1, \ldots, i_0,\ \sigma_i = (\sigma'_i, r_i) \leftarrow H^*.\mathsf{Auth}(sk, \tau_i, m_i)$$

for some $(ek, sk) \leftarrow H^*.\mathsf{KeyGen}(1^\lambda)$, and arbitrary element $\sigma = (\sigma', r) \in \Sigma$, we can write

$$\mathbf{Pr}\left[H^*.\mathsf{Eval}(ek, f, (m_1, \sigma_1), \ldots, (m_l, \sigma_l)) = \sigma \;\middle|\; \begin{array}{c} \forall j = i_0 + 1, \ldots, l, \\ \sigma_j = (\sigma'_j, r_j) \leftarrow H^*.\mathsf{Auth}(sk, \tau_j, m_j) \end{array}\right]$$

$$\leq \mathbf{Pr}\left[f^+(r_1, \ldots, r_{i_0}, r_{i_0+1}, \ldots, r_l) = r \mid \forall j = i_0 + 1, \ldots, l,\ r_j = F^*(\tau_j)\right]$$

where f^+ is the circuit f that every binary gate is replaced to an addition gate over $\mathbb{Z}_p$, and every unary gate is replaced to an identity gate. Note that we can write

$$f^+(r_1^*, \ldots, r_l^*) = \sum_{i=1}^{l} a_i r_i^* \pmod{p}$$

for any $r_1^*, \ldots, r_l^*$ where $1 \leq a_i \leq 2^d < p$ for $i = 1, \ldots, l$ ($1 \leq a_i$ because $\dot{P} = (f, \tau_1, \ldots, \tau_l)$ is the fully bound sub-program of P). Now, define a polynomial h with respect to variables $r_{i_0+1}^*, \ldots, r_l^*$ as

$$\begin{aligned} h(r_{i_0+1}^*, \ldots, r_l^*) =& f^+(r_1, \ldots, r_{i_0}, r_{i_0+1}^*, \ldots, r_l^*) - r \\ =& \sum_{j=i_0+1}^{l} a_j r_j^* + \sum_{k=1}^{i_0} a_k r_k - r \end{aligned}$$

then since $1 \leq a_i < p$ for all $i = 1, \ldots, l$, we can know that h is a degree 1 polynomial with respect to $r_{i_0+1}^*, \ldots, r_l^*$. Therefore, from Schwartz-Zippel lemma,

$$\begin{aligned} &\leq \mathbf{Pr}\left[f^+(r_1, \ldots, r_{i_0}, r_{i_0+1}, \ldots, r_l) = r \mid \forall j = i_0 + 1, \ldots, l,\ r_j = F^*(\tau_j)\right] \\ &= \mathbf{Pr}\left[h(r_{i_0+1}, \ldots, r_l) = 0 \mid \forall j = i_0 + 1, \ldots, l,\ r_j = F^*(\tau_j)\right] \\ &= \mathbf{Pr}\left[h(r_{i_0+1}, \ldots, r_l) = 0 \;\middle|\; \forall j = i_0 + 1, \ldots, l,\ r_j \xleftarrow{\$} \mathbb{Z}_p\right] \\ &\leq \frac{1}{p} \leq \frac{1}{2^\lambda} \end{aligned}$$

□

Remark 7. The very technique employed in Construction 1 is also applicable to homomorphic authenticated encryption as well, to achieve stronger security while preserving pre-processable verification. However, it is not directly applicable to homomorphic signatures due to the public verification.

5 Type II Secure (Leveled) Fully Homomorphic Message Authentication Code

We construct a secure HomMAC using a primitive called homomorphic trapdoor function (HTDF) as Gorbunov, Vaikuntanathan, and Wichs [21]. We first introduce the definition of HTDF and its security notion, and provide a construction

of leveled fully homomorphic trapdoor function from lattice trapdoors. We then propose an adaptively secure (leveled) fully homomorphic MAC scheme with pre-processable verification using a leveled fully homomorphic trapdoor function.

5.1 Homomorphic trapdoor function

Definition 3 (HTDF). *A homomorphic trapdoor function is a tuple of probabilistic polynomial time algorithms* $(\mathsf{KeyGen}, \rho, \mathsf{Inv}, \mathsf{Eval}^{\mathrm{in}}, \mathsf{Eval}^{\mathrm{out}})$ *as follows:*

- $(pk, sk) \leftarrow \mathsf{KeyGen}(1^\lambda)$: *outputs a public key pk and a secret key sk for a given security parameter λ*
- $\rho_{pk,m} : \mathcal{U} \to \mathcal{V}$: *a deterministic function indexed by a public key pk and a message $m \in \mathcal{M}$.*
- $\mathsf{Inv}_{sk,,m} : \mathcal{V} \to \mathcal{U}$: *a probabilistic inverting algorithm of ρ indexed by a secret key sk, and a message m.*
- $u_f \leftarrow \mathsf{Eval}^{\mathrm{in}}(f, (m_1, u_1, v_1), \ldots, (m_l, u_l, v_l))$: *a deterministic evaluation algorithm for inputs of ρ, given an admissible function $f \in \mathcal{F}$ of arity l and (m_1, u_1, v_1), …, $(m_l, u_l, v_l) \in \mathcal{M} \times \mathcal{U} \times \mathcal{V}$.*
- $v_f \leftarrow \mathsf{Eval}^{\mathrm{out}}(f, v_1, \ldots, v_l)$: *a deterministic evaluation algorithm for outputs of ρ, given an admissible function $f \in \mathcal{F}$ of arity l and $v_1, \ldots, v_l \in \mathcal{V}$.*

We assume that a public key pk implicitly contains the information about the message space $\mathcal{M}$, the input space $\mathcal{U}$, the output space $\mathcal{V}$, and the admissible function space $\mathcal{F}$. As HomMAC, we say an HTDF is leveled fully homomorphic if it can homomorphically evaluate any depth d circuit f, or equivalent version of such circuit, for some $d = d(\lambda) = \mathsf{poly}(\lambda)$.

Correctness. An HTDF is required to satisfy the following correctness properties except for negligible probability.

- *Correctness of the evaluation*: For any $m_1, \ldots, m_l \in \mathcal{M}$, arity-$l$ admissible function $f : \mathcal{M}^l \to \mathcal{M}$, security parameter λ, and $(pk, sk) \leftarrow \mathsf{KeyGen}(1^\lambda)$, let
$$\begin{aligned} u_f =& \mathsf{Eval}^{\mathrm{in}}(f, (m_1, u_1, v_1), \ldots, (m_l, u_l, v_l)) \\ v_f =& \mathsf{Eval}^{\mathrm{out}}(f, v_1, \ldots, v_l), \end{aligned}$$
where $v_i = \rho_{pk,m_i}(u_i)$ for $i = 1, \ldots, l$. Then
$$v_f = \rho_{pk,f(m_1,\ldots,m_l)}(u_f)$$
holds.
- *Projection preservation*: For any $m_1, \ldots, m_l \in \mathcal{M}$, arity-$l$ admissible function $f : \mathcal{M}^l \to \mathcal{M}$, security parameter λ and $(pk, sk) \leftarrow \mathsf{KeyGen}(1^\lambda)$,
$$\begin{aligned} u_i =& \mathsf{Eval}^{\mathrm{in}}(\pi_i, (m_1, u_1, v_1), \ldots, (m_l, u_l, v_l)), \\ v_i =& \mathsf{Eval}^{\mathrm{out}}(\pi_i, v_1, \ldots, v_l) \end{aligned}$$
holds where π_i is the ith projection function and $v_i = \rho_{pk,m_i}(u_i)$ for $i = 1, \ldots, l$

Relaxation of correctness In a leveled fully homomorphic scheme, each $u_i \in \mathcal{U}$ will have noise $\beta_i \in \mathbb{Z}$. There is an initial input distribution $D_{\mathcal{U}}$ with each of the element has small noise β_{init}. The noise β_f of an evaluated input $u_f \leftarrow \mathsf{Eval}^{\mathrm{in}}(f, (m_1, u_1, v_1), \ldots, (m_l, u_l, v_l))$ depends on the inputs of $\mathsf{Eval}^{\mathrm{in}}$ and β_i for $i = 1, \ldots, l$. If the noise β_f exceeds a certain threshold $\beta_f > \beta_{max}$, then the correctness of evaluation above does not hold. Therefore, we define the admissible functions accordingly: the function f is admissible if the noise β_f of $u_f \leftarrow \mathsf{Eval}^{\mathrm{in}}(f, (m_1, u_1, v_1), \ldots, (m_l, u_l, v_l))$ does not exceed the noise threshold β_{max} for any choices of $m_1, \ldots, m_l \in \mathcal{M}$, where each noise β_i of u_i does not exceed β_{init} for all $i = 1, \ldots, l$.

Distributional equivalence of inversion Let $(pk, sk) \leftarrow \mathsf{KeyGen}(1^\lambda)$ then the following two distributions are statistically indistinguishable

$$(pk, sk, m, u, v) \overset{\mathrm{stat}}{\approx} (pk, sk, m, u', v')$$

where $m \in \mathcal{M}$, $u \leftarrow D_{\mathcal{U}}$, $v = \rho_{pk,m}(u)$, $v' \xleftarrow{\$} \mathcal{V}$ and $u' \leftarrow \mathsf{Inv}_{sk,m}(v')$.

HTDF security We define the security of an HTDF. Originally, Gorbunov, Vaikuntanathan, and Wichs [21] required an HTDF to satisfy *claw-freeness*. Similarly, we require an HTDF to satisfy *strong claw-freeness* that can be defined by the game below. Let $T = (T.\mathsf{KeyGen}, \rho, \mathsf{Inv}, T.\mathsf{Eval}^{\mathrm{in}}, T.\mathsf{Eval}^{\mathrm{out}})$ be an HTDF.

$\mathrm{Game}^{\mathsf{HTDF}}_{T,\mathcal{A}}(\lambda)$:

Key generation. The challenger generates $(pk, sk) \leftarrow T.\mathsf{KeyGen}(1^\lambda)$ and send pk to $\mathcal{A}$.

Finalization. $\mathcal{A}$ outputs (u, u', m, m') where $u, u' \in \mathcal{U}$, $m, m' \in \mathcal{M}$. The game outputs 1 if $(m, u) \neq (m', u')$ and

$$\rho_{pk,m}(u) = \rho_{pk,m'}(u')$$

Otherwise, the game outputs 0

The *advantage* of the adversary $\mathcal{A}$ in the game for the scheme T is defined as

$$\mathbf{Adv}^{\mathsf{HTDF}}_{T,\mathcal{A}}(\lambda) := \mathbf{Pr}[\mathrm{Game}^{\mathsf{HTDF}}_{T,\mathcal{A}}(\lambda) = 1]$$

We say that an HTDF T satisfies *strong claw-freeness*, if the advantage $\mathbf{Adv}^{\mathsf{HTDF}}_{T,\mathcal{A}}(\lambda)$ is negligible for any probabilistic polynomial time adversary $\mathcal{A}$.

Lattice-based homomorphic trapdoor function We construct a lattice-based HTDF using lattice trapdoors as Gorbunov, Vaikuntanathan, and Wichs [21] based on SIS problem. For given a security parameter λ, let $n = \mathsf{poly}(\lambda)$, $k = \mathsf{poly}(\lambda)$, $q = 2^{\mathsf{poly}(\lambda)}$, and $\beta = \beta(\lambda)$ where $\beta < q$. We define $\mathsf{SIS}_{n,k,q,\beta}$ hardness assumption to imply the following: For any random matrix $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{n \times k}$, there is no probabilistic polynomial time algorithm $\mathcal{A}(1^\lambda, \mathbf{A})$ that outputs a nonzero vector $\mathbf{u} \in \mathbb{Z}_q^k$ such that $\mathbf{Au} = \mathbf{0}$ and $\|\mathbf{u}\|_\infty \leq \beta$.

Lemma 1. *([3,18,4,28,2]) For given integers n and q, we let $k_1 = n\lceil \log q \rceil$, $k_0 = O(n \log q)$, $k = k_0 + k_1$ and $\beta_{sam} = \beta_{sam}(n, q) = O(n\sqrt{\log q})$. We also let $\mathbf{I}_n$ be the n-dimensional identity matrix, $\mathbf{g}^T = \left[1\ 2\ 2^2 \ldots\ 2^{\lceil \log q \rceil - 1}\right]$, and $\mathbf{G}_0 = \mathbf{I}_n \otimes \mathbf{g}^T \in \mathbb{Z}_q^{n \times nk_1}$. For such numbers and an arbitrary $\bar{k} = \bar{k}(n) = \mathsf{poly}(n)$, there exist efficient algorithms* Sam, $\mathsf{TrapGen}$, *and* SamPre *that satisfy the following:*

1. $\mathbf{U} \leftarrow \mathsf{Sam}(1^k, 1^{\bar{k}}, q)$ *outputs a matrix* $\mathbf{U} \in \mathbb{Z}_q^{k \times \bar{k}}$ *such that* $\|\mathbf{U}\|_\infty \leq \beta_{sam}$.
2. $(\mathbf{A}, \mathsf{td}) \leftarrow \mathsf{TrapGen}(\mathbf{A}_0, \mathbf{H})$ *takes a random matrix* $\mathbf{A}_0 \xleftarrow{\$} \mathbb{Z}_q^{n \times k_0}$ *and an arbitrary invertible matrix* $\mathbf{H} \in \mathbb{Z}_q^{n \times n}$ *as its inputs, and outputs a random matrix* $\mathbf{R} \leftarrow D_R$ *from certain distribution* D_R *over* $\mathbb{Z}_q^{k_0 \times k_1}$, *and a matrix* $\mathbf{A} = [\mathbf{A}_0 | \mathbf{H}\mathbf{G}_0 - \mathbf{A}_0\mathbf{R}]$. *Especially,* $(\mathbf{A}, \mathbf{R})$ *and* SamPre *satisfies the following:*
   - $\mathbf{A} \overset{\text{stat}}{\approx} \mathbf{A}'$ *for a randomly chosen matrix* $\mathbf{A}' \xleftarrow{\$} \mathbb{Z}_q^{n \times k}$
   - $(\mathbf{A}, \mathbf{R}, \mathbf{U}, \mathbf{V}) \overset{\text{stat}}{\approx} (\mathbf{A}, \mathbf{R}, \mathbf{U}', \mathbf{V}')$ *when* $\mathbf{U} \leftarrow \mathsf{Sam}(1^k, 1^{\bar{k}}, q)$ *and* $\mathbf{V} := \mathbf{A}\mathbf{U}$, *whereas* $\mathbf{V}' \xleftarrow{\$} \mathbb{Z}_q^{n \times \bar{k}}$ *and* $\mathbf{U}' \leftarrow \mathsf{SamPre}(\mathbf{A}_0, \mathbf{R}, \mathbf{H}, \mathbf{V})$. *Moreover, the output* $\mathbf{U}'$ *of* SamPre *satisfies* $\mathbf{A}\mathbf{U}' = \mathbf{V}'$ *and* $\|\mathbf{U}'\|_\infty \leq \beta_{sam}$.
3. *If we let* $\mathbf{G} := [\mathbf{G}_0 | \mathbf{0}] \in \mathbb{Z}_q^{n \times k}$, *then there exist a deterministic algorithm* $\mathbf{G}^{-1}$ *such that for an arbitrary input* $\mathbf{V} \in \mathbb{Z}_q^{n \times \bar{k}}$, $\mathbf{G}^{-1}$ *outputs* $\mathbf{B} \leftarrow \mathbf{G}^{-1}(\mathbf{V})$ *that satisfies* $\mathbf{G}\mathbf{B} = \mathbf{V}$ *and* $\mathbf{B} \in \{0,1\}^{k \times \bar{k}}$.

Using the result of the above lemma, we construct a lattice-based fully homomorphic trapdoor function as follows.

Construction 2 *For a security parameter λ, we first choose a parameter $d = d(\lambda) = \mathsf{poly}(\lambda)$ which is related to the depth of the admissible functions and let $\beta_{\mathsf{max}} = 2^{\omega(\log \lambda) d}$,$\beta_{\mathsf{SIS}} = 2^{\omega(\log \lambda)} \beta_{\mathsf{max}}$. Then, we choose $n = \mathsf{poly}(\lambda) = \omega(\log \lambda)$, a prime $q = 2^{\mathsf{poly}(\lambda)}$ so that $\mathsf{SIS}_{n,k,q,\beta_{\mathsf{SIS}}}$ hardness assumption holds, where $k_0 = \Theta(n \log q)$, $k_1 = n\lceil \log q \rceil$ and $k = k_0 + k_1$. Let D_R be the distribution in Lemma 1, $D_{\mathcal{U}}$ be the distribution of the outputs of $\mathsf{Sam}(1^k, 1^k, q)$, and $\beta_{\mathsf{init}} = \beta_{sam}$. Let $\mathcal{M} = \{0,1\}$, $\mathcal{U} = \{\mathbf{U} \in \mathbb{Z}_q^{k \times k} \mid \|\mathbf{U}\|_\infty \leq \beta_{\mathsf{max}}\}$, $\mathcal{V} = \mathbb{Z}_q^{n \times k}$, and*

$$\begin{aligned}\mathcal{F} = \{&f : \mathcal{M}^l \to \mathcal{M} \mid \mathbf{U} \leftarrow T.\mathsf{Eval}^{\text{in}}(f, (m_1, \mathbf{U}_1, \mathbf{V}_1), \ldots, (m_l, \mathbf{U}_l, \mathbf{V}_l)) \\ &\text{satisfies } \|\mathbf{U}\|_\infty \leq \beta_{\mathsf{max}} \text{ for any choices of } \mathbf{U}_i \leftarrow D_{\mathcal{U}}, \text{ and } m_i \in \mathcal{M}, \text{ where} \\ &\mathbf{V}_i = \rho_{pk,m}(\mathbf{U}_i), \text{ for all } i = 1, \ldots, l\}\end{aligned}$$

where $T.\mathsf{Eval}^{\text{in}}$ is defined below. We define a leveled fully homomorphic HTDF T as follows:

- $T.\mathsf{KeyGen}(1^\lambda)$: *sample* $\mathbf{A}_0 \xleftarrow{\$} \mathbb{Z}_q^{n \times k_0}$, $\mathbf{R} \leftarrow D_R$ *and let* $\mathbf{A} = [\mathbf{A}_0 | \mathbf{A}_1] = [\mathbf{A}_0 | \mathbf{H}\mathbf{G} - \mathbf{A}_0\mathbf{R}]$ *where* $\mathbf{H} \in \mathbb{Z}_q^{n \times n}$ *is an invertible matrix. Output* $(pk, sk) = (\mathbf{A}, (\mathbf{A}_0, \mathbf{R}, \mathbf{H}))$.
- $\rho_{pk,m} : \mathcal{U} \to \mathcal{V}$: *let* $\mathbf{A} = pk$. *For an input* $\mathbf{U} \in \mathcal{U}$, *deterministically output* $\rho_{pk,m}(\mathbf{U}) = \mathbf{A}\mathbf{U} + m\mathbf{G}$ *where* $\mathbf{G}$ *is defined the same as Lemma 1.*
- $\mathsf{Inv}_{sk,m} : \mathcal{V} \to \mathcal{U}$: *parse* $sk = (\mathbf{A}_0, \mathbf{R}, \mathbf{H})$. *For an input* $\mathbf{V} \in \mathcal{V}$, *run the probabilistic algorithm* $\mathbf{U} \leftarrow \mathsf{SamPre}(\mathbf{A}_0, \mathbf{R}, \mathbf{H}, \mathbf{V} - m\mathbf{G})$ *and output* $\mathbf{U}$.

– $\mathbf{U} \leftarrow T.\mathsf{Eval}^{\mathrm{in}}(f, (m_1, \mathbf{U}_1, \mathbf{V}_1), \ldots, (m_l, \mathbf{U}_l, \mathbf{V}_l))$ *and* $\mathbf{V} \leftarrow T.\mathsf{Eval}^{\mathrm{out}}(f, \mathbf{V}_1, \ldots, \mathbf{V}_l)$ *are defined by evaluation of each gates as follows:*

1. *When* $f(m_1, m_2) = m_1 + m_2$ *is an addition gate,*

$$\mathbf{U} = \mathbf{U}_1 + \mathbf{U}_2, \quad \mathbf{V} = \mathbf{V}_1 + \mathbf{V}_2$$

2. *When* $f(m_1, m_2) = m_1 \cdot m_2$ *is a multiplication gate,*

$$\mathbf{U} = m_2\mathbf{U}_1 + \mathbf{U}_2\mathbf{G}^{-1}(\mathbf{V}_1), \quad \mathbf{V} = \mathbf{V}_2\mathbf{G}^{-1}(\mathbf{V}_1)$$

3. *When* $f(m_1) = m_1 + a$ *is an addition with constant gate for some constant* $a \in \mathbb{Z}_q$*,*

$$\mathbf{U} = \mathbf{U}_1, \quad \mathbf{V} = \mathbf{V}_1 + a\mathbf{G}$$

4. *When* $f(m_1) = a \cdot m_1$ *is a multiplication with constant gate for some constant* $a \in \mathbb{Z}_q$*,*

$$\mathbf{U} = a\mathbf{U}_1, \quad \mathbf{V} = a\mathbf{V}_1$$

Remark 8. The HTDF T defined on Construction 2 satisfies the correctness properties and the distributional equivalence of inversion from Lemma 1.

Remark 9. The HTDF T is leveled fully homomorphic. Think of the NAND gate $g(m_1, m_2) = 1 - m_1 m_2$ and $\mathbf{U}_1, \mathbf{U}_2 \in \mathcal{U}$ such that $\|\mathbf{U}_1\|_\infty \leq \beta$ and $\|\mathbf{U}_2\|_\infty \leq \beta$ for some $\beta > 0$, and $\mathbf{U} \leftarrow T.\mathsf{Eval}^{\mathrm{in}}(g, (m_1, \mathbf{U}_1, \rho_{pk,m_1}(\mathbf{U}_1)), (m_2, \mathbf{U}_2, \rho_{pk,m_2}(\mathbf{U}_2)))$. From the description of $T.\mathsf{Eval}^{\mathrm{in}}$, we can know that the homomorphically evaluated input $\mathbf{U}$ satisfies $\|\mathbf{U}\|_\infty \leq (k+1)\beta$. Therefore, for any depth d circuit f that consists of NAND gates, if we let $\|\mathbf{U}_i\|_\infty \leq \beta_{\mathsf{init}}$ and $\mathbf{V}_i := \rho_{pk,m_i}(\mathbf{U}_i)$ for some $m_i \in \mathcal{M}$ for all $i = 1, \ldots, l$, then the homomorphically evaluated input $U \leftarrow T.\mathsf{Eval}^{\mathrm{in}}(f, (m_1, \mathbf{U}_1, \mathbf{V}_1), \ldots, (m_l, \mathbf{U}_l, \mathbf{V}_l))$ satisfies $\|\mathbf{U}\|_\infty \leq (k+1)^d \beta_{\mathsf{init}} \leq 2^{O(\log \lambda) \cdot d} \leq \beta_{\mathsf{max}}$. Let c be the constant that represents the maximal depth to represent any gate with NAND gate. Since any depth d circuit consists of NAND gate is admissible to T, any circuit with depth $d' = \lfloor d/c \rfloor$, or equivalent of such circuit, is admissible to T as well. As c is a constant, we can write $d' = d'(\lambda) = \mathsf{poly}(\lambda)$.

Remark 10. Using Barrington's theorem, one can convert a depth d circuit into a length 4^d permutation branching program, and the norm of homomorphically evaluated input of such program becomes $(3 \cdot 4^d k + 1)\beta_{\mathsf{init}}$ [33]. Therefore, we can set β_{max} to satisfy $(3 \cdot 4^d k + 1) \leq 2^{O(\log \lambda)} \cdot 2^{2d} \leq \beta_{\mathsf{max}}$. In this way, one can reduce the size of β_{max} and q.

Theorem 4 ([21,33]). *The leveled fully homomorphic HTDF scheme T defined on Construction 2 satisfies strong claw-freeness under $\mathsf{SIS}_{n,k,q,\beta_{SIS}}$ hardness assumption.*

5.2 (Leveled) Fully Homomorphic MAC from HTDF

Using a leveled fully homomorphic trapdoor function that satisfies strong claw-freeness, we can construct a secure (leveled) fully homomorphic MAC scheme with pre-processable verification as follows.

Construction 3 *Let T be an HTDF. Let $\mathcal{M}_T$, $\mathcal{U}_T$, $\mathcal{V}_T$, and $\mathcal{F}_T$ be the message space, the input space, the output space and the admissible function space of T. Let $\mathcal{M} = \mathcal{M}_T$, $\Sigma = \mathcal{U}_T$, $\mathcal{T}$, $\mathcal{P} = \{(f, \tau_1, \ldots, \tau_l) \mid f \in \mathcal{F}_T\}$ be the message space, the ciphertext space, the data identifier space, and the admissible program space of H.*

Let $F : \{0,1\}^\lambda \times \mathcal{T} \to \mathcal{V}_T$ be a secure PRF.

Using T and F, we can construct an HomMAC H as follows.

- $H.\mathsf{KeyGen}(1^\lambda)$: *sample* $k_F \xleftarrow{\$} \{0,1\}^\lambda$, $(pk_T, sk_T) \leftarrow T.\mathsf{KeyGen}(1^\lambda)$, *and output* $(sk, ek) := (pk_T \| sk_T \| k_F, pk_T)$
- $H.\mathsf{Auth}(sk, \tau, m)$: *parse* $sk = pk_T \| sk_T \| k_F$, *let* $v := F(k_F, \tau)$, *run* $u \leftarrow \mathsf{Inv}_{sk_T, m}(v)$, *and output* u.
- $H.\mathsf{Eval}(ek, f, (m_1, u_1), \ldots, (m_l, u_l))$: *let* $pk_T = ek$ *and* $v_i = \rho_{pk_T, m_i}(u_i)$ *for* $i = 1, \ldots, l$. *Compute* $u \leftarrow T.\mathsf{Eval}^{\mathrm{in}}(f, (m_1, u_1, v_1), \ldots, (m_l, u_l, v_l))$ *and* $m = f(m_1, \ldots, m_l)$. *Output* u.
- $H.\mathsf{Prep}(sk, P)$: *parse* $sk = pk_T \| sk_T \| k_F$ *and* $P = (f, \tau_1, \ldots, \tau_l)$. *Let* $v_i := F(k_F, \tau_i)$ *for* $i = 1, \ldots, l$. *Let* $v \leftarrow T.\mathsf{Eval}^{\mathrm{out}}(f, v_1, \ldots, v_l)$, *and output* $sk_P := pk_T \| v$.
- $H.\mathsf{EffVerify}(sk_P, m, u)$: *parse* $sk_P = pk_T \| v$. *If* $v = \rho_{pk_T, m}(u)$, *output 1. Otherwise, output 0.*
- $H.\mathsf{Verify}(sk, P, m, u) = H.\mathsf{EffVerify}(H.\mathsf{Prep}(sk, P), m, u)$.

Remark 11. From the correctness of the HTDF T, H satisfies the correctness properties of HomMAC. Since $H.\mathsf{EffVerify}$ is independent of the complexity of P, H supports pre-processable verification. If T is leveled fully homomorphic, then so is H.

Note that Construction 3 resembles the first (leveled) fully homomorphic signature scheme except for the fact that k_F remains secret. Using this secrecy, we can prove that Construction 3 is adaptively secure against Type II forgery.

Theorem 5. *If T satisfies strong claw-freeness, then H of Construction 3 is Type II secure.*

Proof. Define $\mathbf{Adv}_F^{\mathsf{PRF}}(\lambda)$ to be distinguishing advantage of $F(k_F, \cdot)$ from a random function $F' : \mathcal{T} \to \mathcal{V}_T$ for randomly chosen k_F with respect to the security parameter λ. Now, we switch pseudorandom function F with a random function using the games defined as follows:

Game $0(\lambda)$: The original security game $\mathrm{Game}_{H,\mathcal{A}}^{\mathsf{HomMAC,II}}(\lambda)$.

Game $1(\lambda)$: The same as Game $0(\lambda)$, but uses a slightly different construction H' which is the same as H except for the parts that use F. In this game, $H'.\mathsf{KeyGen}(1^\lambda)$ samples a random function $F' : \mathcal{T} \to \mathcal{V}_T$ and lets $sk := pk' \| sk' \| F'$ as a secret key. Also, $H'.\mathsf{Auth}$ and $H'.\mathsf{Prep}$ use $F'(\cdot)$ instead of $F(k_F, \cdot)$.

If we define $\mathbf{Adv}_{\mathcal{A}}^{\text{Game } x, \text{Game } y}(\lambda) := |\mathbf{Pr}[\text{Game } x(\lambda) = 1] - \mathbf{Pr}[\text{Game } y(\lambda) = 1]|$ for any indices x and y, then we can write

$$\mathbf{Adv}_F^{\mathsf{PRF}}(\lambda) \geq \mathbf{Adv}_{\mathcal{A}}^{\text{Game } 0, \text{Game } 1}(\lambda)$$

Let $\mathcal{A}$ be an adversary of H against Game 1 with at most q queries. We define a probabilistic polynomial time adversary $\mathcal{B}$ that runs $\mathcal{A}$ internally to win the game $\text{Game}_{T,\mathcal{B}}^{\mathsf{HTDF}}(\lambda)$ as follows.

When $\text{Game}_{T,\mathcal{B}}^{\mathsf{HTDF}}(\lambda)$ starts, the challenger initializes the keys $(pk_T, sk_T) \leftarrow T.\mathsf{KeyGen}(1^\lambda)$ and sends pk_T to $\mathcal{B}$. Now, without knowing sk_T, $\mathcal{B}$ simulates the challenger of Game 1 as follows:

- $\mathcal{B}$ samples a random function $F' : \mathcal{T} \to \mathcal{V}_T$ and sends $ek = pk_T$ to $\mathcal{A}$.
- $\mathcal{B}$ initializes a set S as $\emptyset$.
- For each authentication query (τ, m), $\mathcal{B}$ first checks if $(\tau, \cdot, \cdot) \in S$. If $(\tau, \cdot, \cdot) \notin S$, then $\mathcal{B}$ samples $u \leftarrow D_{\mathcal{U}}$ (which is defined in Construction 2), computes $v := \rho_{pk_T, m}(u)$, programs F' to satisfy $F'(\tau) = v$, and updates $S \leftarrow S \cup \{(\tau, m, u)\}$. If $(\tau, \cdot, \cdot) \in S$, then $\mathcal{B}$ rejects the query.
- For each verification query (P, m, u), $\mathcal{B}$ first checks if the query is redundant, Type I, or Type II with respect to S. If the query is redundant or Type I with respect to S, then $\mathcal{B}$ rejects the verification query. If the query is Type II with respect to S, then we let $\dot{P} := (f, \tau_1, \ldots, \tau_l)$ be the fully bound subprogram of P, and find (unique) $(m_i, u_i) \in \mathcal{M} \times \Sigma$ such that $(\tau_i, m_i, u_i) \in S$. Since (P, m, u) is Type II verification query, we can know that

$$(f(m_1, \ldots, m_l), H'.\mathsf{Eval}(ek, f, (m_1, u_1), \ldots, (m_l, u_l))) \neq (m, u)$$

  As $H'.\mathsf{Prep}$ and $H'.\mathsf{Verify}$ does not use sk_T, $\mathcal{B}$ can compute $b = H'.\mathsf{Verify}(sk, P, m, u)$ using pk_T and F'. If $b = 0$, then $\mathcal{B}$ also send 0 to $\mathcal{A}$. If $b = 1$, then $\mathcal{B}$ lets $(m', u') := (f(m_1, \ldots, m_l), H'.\mathsf{Eval}(ek, f, (m_1, u_1), \ldots, (m_l, u_l)))$ and outputs (u, u', m, m') and halts.

If we let $v_i := \rho_{pk_T, m_i}(u_i)$ and $v := T.\mathsf{Eval}^{\text{out}}(f, v_1, \ldots, v_l)$, then we can write

$$\begin{aligned} u' =& H'.\mathsf{Eval}(ek, f, (m_1, u_1), \ldots, (m_l, u_l)) \\ =& T.\mathsf{Eval}^{\text{in}}(f, (m_1, u_1, v_1), \ldots, (m_l, u_l, v_l)) \end{aligned}$$

and $\rho_{pk_T, m}(u) = \rho_{pk_T, m'}(u') = v$ from the correctness of T and the fact that (P, m, u) is a Type II forgery. Since $(m, u) \neq (m', u')$, $\mathcal{B}$'s output (u, u', m, m') becomes a claw of T and wins $\text{Game}_{T,\mathcal{B}}^{\mathsf{HTDF}}(\lambda)$.

In $\mathcal{A}$'s perspective, thanks to Lemma 1, $\mathcal{B}$'s simulation is statistically indistinguishable to the challenger of Game $1(\lambda)$. Therefore, if we let $\mathbf{Adv}_{\mathcal{A}}^{\text{Game } i}(\lambda) := \mathbf{Pr}[\mathcal{A} \text{ wins Game } i(\lambda)]$ for $i = 1, 2$, then we can write

$$\mathbf{Adv}_{\mathcal{A}}^{\text{Game } 1}(\lambda) \leq \mathbf{Adv}_{T,\mathcal{B}}^{\mathsf{HTDF}}(\lambda) + \mathsf{negl}()$$

and conclude that

$$\begin{aligned}\mathbf{Adv}_{H,\mathcal{A}}^{\mathsf{HomMAC},\Pi}(\lambda) = \mathbf{Adv}_{\mathcal{A}}^{\text{Game } 0}(\lambda) &\leq \mathbf{Adv}_{\mathcal{A}}^{\text{Game } 0,\text{Game } 1} + \mathbf{Adv}_{\mathcal{A}}^{\text{Game } 1}(\lambda) \\ &\leq \mathbf{Adv}_{F}^{\mathsf{PRF}}(\lambda) + \mathbf{Adv}_{T,\mathcal{B}}^{\mathsf{HTDF}}(\lambda) + \mathsf{negl}()\end{aligned}$$

□

6 Conclusion

In this work, we suggest a dedicated scheme for (leveled) fully homomorphic MAC that is adaptively secure, and has pre-processable verification. However, due to the secrecy of the primitive, it is yet unknown how to convert a single-dataset scheme into a multi-dataset scheme while preserving security and pre-processable verification.

References

1. Afshar, A., Cheng, J., Goyal, R.: Leveled fully-homomorphic signatures from batch arguments. Cryptology ePrint Archive (2024)
2. Agrawal, S., Boneh, D., Boyen, X.: Efficient lattice (H) IBE in the standard model. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 553–572. Springer (2010)
3. Ajtai, M.: Generating hard instances of the short basis problem. In: International Colloquium on Automata, Languages, and Programming. pp. 1–9. Springer (1999)
4. Alwen, J., Peikert, C.: Generating shorter bases for hard random lattices (2009)
5. Anthoine, G., Balbás, D., Fiore, D.: Fully-succinct multi-key homomorphic signatures from standard assumptions. In: Annual International Cryptology Conference. pp. 317–351. Springer (2024)
6. Balbás, D., Catalano, D., Fiore, D., Lai, R.W.: Chainable functional commitments for unbounded-depth circuits. In: Theory of Cryptography Conference. pp. 363–393. Springer (2023)
7. Boyen, X., Fan, X., Shi, E.: Adaptively secure fully homomorphic signatures based on lattices. Cryptology ePrint Archive (2014)
8. Boyle, E., Goldwasser, S., Ivan, I.: Functional signatures and pseudorandom functions. In: International workshop on public key cryptography. pp. 501–519. Springer (2014)
9. Brakerski, Z., Gentry, C., Vaikuntanathan, V.: (leveled) fully homomorphic encryption without bootstrapping. ACM Transactions on Computation Theory (TOCT) **6**(3), 13 (2014)
10. Brakerski, Z., Vaikuntanathan, V.: Efficient fully homomorphic encryption from (standard) LWE. SIAM Journal on computing **43**(2), 831–871 (2014)

11. Catalano, D., Fiore, D., Nizzardo, L.: On the security notions for homomorphic signatures. In: Preneel, B., Vercauteren, F. (eds.) Applied Cryptography and Network Security - 16th International Conference, ACNS 2018, Leuven, Belgium, July 2-4, 2018, Proceedings. Lecture Notes in Computer Science, vol. 10892, pp. 183–201. Springer (2018)
12. Catalano, D., Fiore, D., Tucker, I.: Additive-homomorphic functional commitments and applications to homomorphic signatures. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 159–188. Springer (2022)
13. Cheon, J.H., Kim, A., Kim, M., Song, Y.: Homomorphic encryption for arithmetic of approximate numbers. In: Advances in Cryptology–ASIACRYPT 2017: 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part I 23. pp. 409–437. Springer (2017)
14. Datta, P., Dutta, R., Mukhopadhyay, S.: Short attribute-based signatures for arbitrary turing machines from standard assumptions. Designs, Codes and Cryptography **91**(5), 1845–1872 (2023)
15. El Kaafarani, A., Katsumata, S.: Attribute-based signatures for unbounded circuits in the rom and efficient instantiations from lattices. In: Public-Key Cryptography–PKC 2018: 21st IACR International Conference on Practice and Theory of Public-Key Cryptography, Rio de Janeiro, Brazil, March 25-29, 2018, Proceedings, Part II 21. pp. 89–119. Springer (2018)
16. Gay, R., Ursu, B.: On instantiating unleveled fully-homomorphic signatures from falsifiable assumptions. In: IACR International Conference on Public-Key Cryptography. pp. 74–104. Springer (2024)
17. Gennaro, R., Wichs, D.: Fully homomorphic message authenticators. In: Sako, K., Sarkar, P. (eds.) Advances in Cryptology - ASIACRYPT 2013 - 19th International Conference on the Theory and Application of Cryptology and Information Security, Bengaluru, India, December 1-5, 2013, Proceedings, Part II. Lecture Notes in Computer Science, vol. 8270, pp. 301–320. Springer (2013)
18. Gentry, C., Peikert, C., Vaikuntanathan, V.: Trapdoors for hard lattices and new cryptographic constructions. In: Proceedings of the fortieth annual ACM symposium on Theory of computing. pp. 197–206. ACM (2008)
19. Gentry, C., Sahai, A., Waters, B.: Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In: Annual Cryptology Conference. pp. 75–92. Springer (2013)
20. Gentry, C., et al.: Fully homomorphic encryption using ideal lattices. In: Stoc. vol. 9, pp. 169–178 (2009)
21. Gorbunov, S., Vaikuntanathan, V., Wichs, D.: Leveled fully homomorphic signatures from standard lattices. In: Servedio, R.A., Rubinfeld, R. (eds.) Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015. pp. 469–477. ACM (2015)
22. Goyal, R.: Mutable batch arguments and applications. Cryptology ePrint Archive (2024)
23. Hayashi, R., Sakai, Y., Yamada, S.: Attribute-based signatures for circuits with optimal parameter size from standard assumptions. Cryptology ePrint Archive (2024)
24. Joo, C., Yun, A.: Homomorphic authenticated encryption secure against chosen-ciphertext attack. In: Sarkar, P., Iwata, T. (eds.) Advances in Cryptology - ASIACRYPT 2014 - 20th International Conference on the Theory and Application

of Cryptology and Information Security, Kaoshiung, Taiwan, R.O.C., December 7-11, 2014, Proceedings, Part II. Lecture Notes in Computer Science, vol. 8874, pp. 173–192. Springer (2014)
25. Kim, J., Yun, A.: Secure fully homomorphic authenticated encryption. IEEE Access **9**, 107279–107297 (2021)
26. Ling, S., Nguyen, K., Phan, D.H., Tang, K.H., Wang, H., Xu, Y.: Fully dynamic attribute-based signatures for circuits from codes. In: IACR International Conference on Public-Key Cryptography. pp. 37–73. Springer (2024)
27. Luo, F., Wang, F., Wang, K., Chen, K.: A more efficient leveled strongly-unforgeable fully homomorphic signature scheme. Information Sciences **480**, 70–89 (2019)
28. Micciancio, D., Peikert, C.: Trapdoors for lattices: Simpler, tighter, faster, smaller. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 700–718. Springer (2012)
29. Sakai, Y., Attrapadung, N., Hanaoka, G.: Attribute-based signatures for circuits from bilinear map. In: Public-Key Cryptography–PKC 2016: 19th IACR International Conference on Practice and Theory in Public-Key Cryptography, Taipei, Taiwan, March 6-9, 2016, Proceedings, Part I. pp. 283–300. Springer (2016)
30. Sakai, Y., Katsumata, S., Attrapadung, N., Hanaoka, G.: Attribute-based signatures for unbounded languages from standard assumptions. In: Advances in Cryptology–ASIACRYPT 2018: 24th International Conference on the Theory and Application of Cryptology and Information Security, Brisbane, QLD, Australia, December 2–6, 2018, Proceedings, Part II 24. pp. 493–522. Springer (2018)
31. Tsabary, R.: An equivalence between attribute-based signatures and homomorphic signatures, and new constructions for both. In: Theory of Cryptography Conference. pp. 489–518. Springer (2017)
32. Wang, C., Wu, B., Yao, H.: Leveled adaptively strong-unforgeable identity-based fully homomorphic signatures. IEEE Access **8**, 119431–119447 (2020)
33. Wang, F., Wang, K., Li, B., Gao, Y.: Leveled strongly-unforgeable identity-based fully homomorphic signatures. In: López, J., Mitchell, C.J. (eds.) Information Security - 18th International Conference, ISC 2015, Trondheim, Norway, September 9-11, 2015, Proceedings. Lecture Notes in Computer Science, vol. 9290, pp. 42–60. Springer (2015)
34. Wang, Y., Wang, M.: A new fully homomorphic signatures from standard lattices. In: International Conference on Wireless Algorithms, Systems, and Applications. pp. 494–506. Springer (2020)
35. Wee, H., Wu, D.J.: Succinct functional commitments for circuits from k-lin. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 280–310. Springer (2024)