IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 00.0000/ACCESS.0000.DOI

Secure Fully Homomorphic Authenticated Encryption

JEONGSU KIM[1] AND AARAM YUN[2]

[1]Department of Computer Science and Engineering, Ulsan National Institute of Science and Technology (UNIST), Republic of Korea
[2]Department of Cyber Security, Ewha Womans University, Republic of Korea

Corresponding authors: Jeongsu Kim (e-mail: jsk2357@unist.ac.kr) and Aaram Yun (e-mail: aaramyun@ewha.ac.kr).

This work was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No. 2016-6-00598, The mathematical structure of functional encryption and its analysis).

ABSTRACT Homomorphic authenticated encryption allows implicit computation on plaintexts using corresponding ciphertexts without losing privacy, and provides authenticity of the computation and the resultant plaintext of the computation when performing a decryption. However, due to its special functionality, the security notions of the homomorphic authenticated encryption is somewhat complicated and the construction of fully homomorphic authenticated encryption has never been given. In this work, we propose a new security notion and the first construction of fully homomorphic authenticated encryption. Our new security notion is a unified definition for data privacy and authenticity of homomorphic authenticated encryption. Moreover, our security notion is simpler and stronger than the previous ones. To realize our new security notion, we also suggest a construction of fully homomorphic authenticated encryption via generic construction. We combine a fully homomorphic encryption and two homomorphic authenticators, one fully homomorphic and one OR-homomorphic, to construct a fully homomorphic authenticated encryption that satisfies our security notion. Our construction requires its fully homomorphic encryption to be indistinguishable under chosen plaintext attacks and its homomorphic authenticators to be unforgeable under selectively chosen plaintext queries. Our construction also supports multiple datasets and amortized efficiency. For efficiency, we also construct a multi-dataset fully homomorphic authenticator scheme, which is a variant of the first fully homomorphic signature scheme. Our multi-dataset fully homomorphic authenticator scheme satisfies the security requirement of our generic construction above and supports amortized efficiency.

INDEX TERMS authentication, cryptography, data security, encryption, fully homomorphic authenticated encryption, homomorphic authenticator, homomorphic encryption, security notion

I. INTRODUCTION

While the idea of homomorphic cryptography itself is quite old [1], it was Gentry's first fully homomorphic encryption (FHE) scheme [2] which has strongly motivated the whole homomorphic cryptography area. Since then, there has been many works (for example, [3]–[6]) on cryptographic primitives that have homomorphic property such as homomorphic authenticators (HA), homomorphic authenticated encryption (HAE) and homomorphic encryption (HE) itself. These primitives play an important role in cloud computing since they preserve security while allowing homomorphic evaluations on ciphertexts or authentication tags. In this work, we focus on HAE.

HAE provides privacy and authenticity of the plaintext and assures that the ciphertext is honestly generated with respect to a given circuit. Similar to HE, we may use ciphertexts of HAE to homomorphically evaluate a given circuit to produce the ciphertext which decrypts to the value of the circuit, without losing privacy. Moreover, using the decryption algorithm of HAE, we can verify whether the ciphertext of HAE is valid or not.

The first formal definition and construction of HAE was given by Joo and Yun [5]. In their work, the security notion of HAE was given in two parts; privacy and authenticity. As an encryption scheme, a secure HAE scheme is required to be indistinguishable against chosen plaintext attack (IND-CPA), or even chosen ciphertext attack (IND-CCA) formalized in the usual find-then-guess games. Also, as an authentication scheme, a secure HAE scheme must be strongly unforgeable against chosen plaintext attack (SUF-CPA) or chosen cipher-

text attack (SUF-CCA). The authors showed that an HAE scheme which satisfies IND-CPA and SUF-CPA, relatively weaker security notions, necessarily satisfies the strongest security notions, IND-CCA and SUF-CCA. Using this, the authors showed that their construction of a somewhat homomorphic authenticated encryption scheme satisfies IND-CCA and SUF-CCA.

However, there are some shortcomings of Joo and Yun's work [5]. First, the definition of IND-CCA security has complicated restriction on the decryption queries that adversaries are allowed to make. This restriction comes from the homomorphic functionality of HAE and the find-then-guess formalism that was used for defining IND-CCA security. Specifically, after the challenge phase, the adversary can easily produce a ciphertext that allows winning the challenge, using homomorphic evaluation on the challenge ciphertext. Therefore, rather complicated restrictions on the decryption queries after the challenge phase were unavoidable to define IND-CCA security in the find-then-guess formalism. Second, the first construction of HAE is only somewhat homomorphic and not compatible with multiple datasets. For broader usage, multi-dataset fully homomorphic authenticated encryption (MDFHAE) is more desirable.

A. OUR CONTRIBUTIONS

In this work, we mainly propose some improvements of aforementioned shortcomings of the original HAE work [5]. First, we propose a new security notion that is simpler and stronger than the original ones and implies privacy and authenticity simultaneously. Second, we propose the first fully homomorphic authenticated encryption (FHAE) scheme that is compatible with multiple datasets and has amortized efficiency. Lastly, we propose an efficient multi-dataset fully homomorphic authenticator (MDFHA) scheme that can be used as a part of the generic construction of FHAE.

Our security notion follows real-or-random-like formalism instead of find-then-guess. The challenger first flips a coin b and defines oracle E and D according to the value of b. If $b = 0$, then the challenger lets $E = \mathrm{Enc}$ and $D = \mathrm{Dec}$ where Enc and Dec are encryption and decryption algorithms, respectively, of an HAE scheme. On the other hand, if $b = 1$, then the challenger lets $E = \$$ and $D = \perp$ where $\$(\cdot)$ is an algorithm that samples a ciphertext independent of the message input and $\perp(\cdot)$ is the trivial algorithm that always outputs $\perp$. Afterwards, the challenger gives oracle access of E and D to the adversary. To win the security game, the adversary must guess b correctly. To prevent trivial distinguishing attacks, the adversary is not allowed to make decryption queries, the queries to the oracle D, that he or she already knows the answer of; a ciphertext that is generated by homomorphic evaluation on the previous encryption queries, the queries to the oracle E. One may say that the purpose of the security game is to distinguish the real world, $b = 0$, and the ideal world, $b = 1$.

Our new security notion has two advantages over the original definition. First, it is simpler. Unlike the definition of IND-CCA security, our security notion's restriction on adversaries is straightforward. Later on, the restricted decryption queries will be called *redundant* queries since it is a homomorphic version of classical redundant queries. Second, it is stronger. It can be proved that our new security notion implies IND-CCA and SUF-CCA security, the strongest notions prior to ours, at the same time.

For realization, we also propose the first FHAE scheme that satisfies our new security notion. Overall, our FHAE scheme can be considered as following the encrypt-then-authenticate generic composition paradigm, using one FHE scheme and one fully homomorphic authenticator. In order to achieve the security definition we propose, our construction uses one additional OR-homomorphic HA. The ciphertext c of our scheme consists of three parts: a ciphertext $\mathring{c}$ of the FHE scheme that encrypts a plaintext m, an authentication tag $\bar{\sigma}$ of the FHA scheme that authenticates $\mathring{c}$, and another authentication tag $\breve{\sigma}$ of the OR-homomorphic authenticator. For broader usage, our FHAE scheme maintains the multi-dataset compatibility and amortized efficiency if the two HAs used also have the same property.

Our MDFHAE construction can also be seen as an MDFHA with amortized efficiency. To our knowledge, our MDFHAE construction is the second method to achieve an adaptively secure MDFHA with amortized efficiency. The first such scheme is the fully homomorphic signature scheme of Gorbunov, Vaikuntanathan and Wichs [4], which is constructed using their selective-secure basic scheme. Compared with theirs, our construction satisfies stronger authenticity guarantee, have essentially the same efficiency, and provides privacy as well.

For efficiency of our FHAE scheme, we also propose an efficient MDFHA that satisfies the security requirement of our generic construction of FHAE. Namely, our MDFHA scheme is strongly unforgeable against selectively chosen message attacks. Our MDFHA scheme is a variant of the first fully homomorphic signature scheme [4], but our construction supports multiple datasets more efficiently.

B. RELATED WORKS

1) Adaptively secure fully homomorphic authenticators

Since the first construction of fully homomorphic authenticator (FHA) scheme [3], there have been some studies about adaptively secure FHA. The most notable one is the work with the first fully homomorphic signature scheme presented by Gorbunov, Vaikuntanathan and Wichs [4]. The authors proposed a selectively secure fully homomorphic signature scheme using lattice trapdoors, and constructed methods to strengthen security and functionality. Based on their methods, one can construct an adaptively secure MDFHA scheme with amortized efficiency. Other works proposed adaptively secure fully homomorphic authenticators as well, but failed to preserve amortized efficiency [7]–[9].

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3100852, IEEE Access

2) Stronger notion of fully homomorphic authenticators

Catalano, Fiore, and Nizzardo have proposed a simpler and stronger security notion for homomorphic signatures [10]. Their new notion is more intuitive than previous ones. The authors also have proposed a generic transform using OR-homomorphic signatures that can make existing homomorphic signatures to satisfy their new security notion. Our construction adapts their use of OR-homomorphic authentication for our purposes.

3) Unified security notion for authenticated encryptions

Our security definition can be considered as a homomorphic adaptation of the 'all-in-one' security definition for authenticated encryptions first given by Rogaway and Shrimpton in [11]. As ours, their security notion challenges the adversary to distinguish the encryption/decryption oracles from the random/rejection oracles. The authors also proved that their security notion is equivalent to the original definitions of privacy and authenticity.

II. PRELIMINARIES

A. NOTATIONS AND CONVENTIONS

$(\gamma, \cdot) \in S$ and $(\gamma, \cdot) \notin S$ are shorthand notations for $\exists x, (\gamma, x) \in S$ and $\forall x, (\gamma, x) \notin S$, respectively. Similarly, we will use notations like $(\gamma, \cdot, \cdot, \cdot) \in S$.

For a function $f : \mathcal{X} \times \mathcal{Y} \to \mathcal{Z}$, we define a function $f(x, \cdot) : \mathcal{Y} \to \mathcal{Z}$ as $f(x, \cdot)(y) := f(x, y)$. Similarly, for an algorithm Alg defined on input space $\mathcal{X} \times \mathcal{Y}$, we define $Alg(x, \cdot)$ as the algorithm that takes $y \in \mathcal{Y}$ as an input and outputs $Alg(x, y)$. Similarly, we will use notations like $f'(\cdot, x, \cdot, \cdot)$ and $Alg'(x, \cdot, y, \cdot)$.

For any positive integer n, the set $\{1, \ldots, n\}$ is denoted by $[n]$

Let X and Y be random variables on $\mathcal{X}$ and $\mathcal{Y}$, respectively. We say X and Y are *statistically indistinguishable*, or $X \overset{\text{stat}}{\approx} Y$ if the value $\frac{1}{2}\sum_{z \in \mathcal{X} \cup \mathcal{Y}} |\mathbf{Pr}[X = z] - \mathbf{Pr}[Y = z]|$ is negligible.

Let D be a distribution and $\mathcal{X}$ be a set. If x is sampled according to the distribution D, then we write $x \leftarrow D$. If x is sampled from uniform distribution of the set $\mathcal{X}$, then we write $x \xleftarrow{\$} \mathcal{X}$.

We define advantages of indexed games $\mathbf{Adv}_A^{\textbf{Game } i}(\lambda) := \left|\frac{1}{2} - \mathbf{Pr}[\textbf{Game } i(\lambda) = 1]\right|$ and $\mathbf{Adv}_A^{\textbf{Game } i, \textbf{Game } j}(\lambda) := \frac{1}{2}\left|\mathbf{Pr}\left[\textbf{Game } i(\lambda) = 1\right] - \mathbf{Pr}\left[\textbf{Game } j(\lambda) = 1\right]\right|$ for any indices i and j,

For an integer q, we write the ring of integers modulo q as $\mathbb{Z}_q$. We represent elements in $\mathbb{Z}_q$ by the integers in $(-q/2, q/2]$. For a matrix (or a vector) $\mathbf{U} \in \mathbb{Z}_q^{n \times m}$, we write $\|\mathbf{U}\|_\infty \leq \beta$ if the absolute values of every entry in $\mathbf{U}$ does not exceed β.

For a function f, we write $f(\lambda) = \mathsf{poly}(\lambda)$ if there is a constant $C > 0$ such that $f(\lambda) = O(\lambda^C)$.

For a function f, we say that f is negligible with respect to λ, or write $f(\lambda) = \mathsf{negl}(\lambda)$, if $f(\lambda) = o(\lambda^{-C})$ for any constant C. In this work, we often just say f is negligible if f is negligible with respect to the security parameter λ that can be inferred.

We will consider cryptographic schemes which can be formalized as a tuple of algorithms. When $H = (alg_1, alg_2, \ldots, alg_l)$ is such a cryptographic scheme, we will refer to its components using the dot notation: $H.alg_i$ is alg_i in the tuple $H = (alg_1, alg_2, \ldots, alg_l)$.

1) Circuits

As in Joo and Yun [5], here a *circuit* is a directed acyclic graph (DAG) where a gate is assigned to each vertex with a positive indegree and a positive outdegree, and there is a unique dedicated wire w_{out} called the output wire. For convenience, we assume that all gates in a circuit have indegree 1 or 2, but the construction can be generalized to more general cases.

2) Binary encoding

We assume that any set S in this work has a certain integer n such that any element $e \in S$ is given by a binary encoding $\langle e_1, \ldots, e_n \rangle$ of e such that $e_i \in \{0, 1\}$ for $i \in [n]$.

3) Bitwisely described circuits

Suppose every element of $\mathcal{C}$ is given by n-bit binary encoding. A *bitwisely described* circuit $\bar{f} = (f_1, \ldots, f_n)$ is a function from $\mathcal{C}^l$ to $\mathcal{C}$ such that

$$\begin{aligned}&\bar{f}(c_1, \ldots, c_l)\\ &= \left\langle f_1\left((c_{i',j'})_{(i',j') \in [l] \times [n]}\right), \ldots, f_n\left((c_{i',j'})_{(i',j') \in [l] \times [n]}\right)\right\rangle\end{aligned}$$

where $f_j : \{0,1\}^{ln} \to \{0,1\}$ are circuits for $j \in [n]$ and $\langle c_{i,1}, \ldots, c_{i,n} \rangle$ is the binary encoding of $c_i \in \mathcal{C}$ for $i \in [l]$. We say f_i is the *ith encoding circuit* of a bitwisely described circuit $\bar{f} = (f_1, \ldots, f_n)$. The *depth* of a bitwisely described circuit $\bar{f} = (f_1, \ldots, f_n)$ is defined by the maximum value of the depths of the circuits $f_1, \ldots, f_n$. Any deterministic algorithm that takes an element in $\mathcal{C}^l$ and outputs an element in $\mathcal{C}$ can be given as a bitwisely described circuit. We also say $\bar{\pi}_k$ is the *bitwisely described kth projection* from $\mathcal{C}^l$ to $\mathcal{C}$ if $\bar{\pi}_k = (\pi_{k,1}, \ldots, \pi_{k,n})$ for some n projection circuits $\pi_{k,1}, \ldots, \pi_{k,n}$ such that $\pi_{k,j}((b_{i',j'})_{(i',j') \in [l] \times [n]}) = b_{k,j}$ for any bits $b_{i',j'}$ for $(i', j') \in [l] \times [n]$. We can see that

$$\begin{aligned}&\bar{\pi}_k(c_1, \ldots, c_l)\\ &= \left\langle \pi_{k,1}((c_{i',j'})_{(i',j') \in [l] \times [n]}), \ldots, \pi_{k,n}((c_{i',j'})_{(i',j') \in [l] \times [n]})\right\rangle\\ &= \langle c_{k,1}, \ldots, c_{k,n} \rangle = c_k\end{aligned}$$

where $c_{i',j'}$ are bits such that $\langle c_{i,1}, \ldots, c_{i,n} \rangle$ is the binary encoding of $c_i \in \mathcal{C}$ for $i = [l]$.

B. (MULTI-)LABELED PROGRAMS

1) Labeled programs

Since we consider situations where storage is outsourced to a server and all the client has is only some metadata, we need to be able to refer to the client's data when evaluating a function. This notion is formalized as the labeled program [3].

An HAE encrypts a plaintext $m \in \mathcal{M}$ under a 'label' $\tau \in \mathcal{L}$, and a *labeled program* is a function together with information telling which plaintexts should be used as inputs.

More formally, let $\mathcal{M}$ be a set which we consider as the message space and $\mathcal{L}$ the 'label space'. A *label* is simply an arbitrary element $\tau \in \mathcal{L}$. A *labeled program* is a tuple $P = (f, \tau_1, \ldots, \tau_l)$, where $f : \mathcal{M}^l \to \mathcal{M}$ is a function with arity l, and $\tau_1, \ldots, \tau_l \in \mathcal{L}$ are labels. The intuition is that a message $m \in \mathcal{M}$ is associated with the label $\tau \in \mathcal{L}$, and the label τ is used to refer to m. Then, evaluating the labeled program $P = (f, \tau_1, \ldots, \tau_l)$ means computing $f(m_1, \ldots, m_l)$, where m_i is the message associated with the label τ_i. We also write the resulting function value as $P(m_1, \ldots, m_l)$.

Let $P_1, \ldots, P_l$ be labeled programs such that $P_i = (f_i, \tau_{i,1}, \ldots, \tau_{i,l_i})$. For $P_1, \ldots, P_l$ and a circuit $f : \mathcal{M}^l \to \mathcal{M}$, the *composed program* denoted by $f(P_1, \ldots, P_l)$ is a labeled program that evaluates the circuit f on the outputs of $P_1, \ldots, P_l$. More specifically, $f(P_1, \ldots, P_l) = (f^*, \tau, \ldots, \tau_{l^*})$ where f^* is a circuit that evaluates f on the outputs of circuits $f_1, \ldots, f_l$ where the input wires with the same labels are merged, and $\tau, \ldots, \tau_{l^*}$ are all distinct labels of $P_1, \ldots, P_l$.

If there is a set of admissible functions $\mathcal{F}$ for an HAE or homomorphic authenticator defined below, then we say $P = (f, \tau_1, \ldots, \tau_l)$ is an admissible program if $f \in \mathcal{F}$.

Let $P = (f, \tau_1, \ldots, \tau_l)$ be a labeled program. We say P is *fully bound* if there is a path from every internal wire (including input wire) to the output wire in the circuit f. For any labeled program P, using a graph traversal algorithm, we can always find the labeled programs P_0 and P' such that $P' = (f', \tau'_1, \ldots, \tau'_{l'})$ is fully bound, $P = \pi_1(P', P_0)$ and every path that ends with output wire is included in P' where π_1 is the first projection function. We say such P' is the *fully bound sub-program* of P.

2) Bitwisely described programs

Suppose every element of $\mathcal{M}$ is given by n-bit binary encoding. A *bitwisely described program* is a tuple $\bar{P} = \left(\bar{f}, \tau_1, \ldots, \tau_l\right)$, where $\bar{f} = (f_1, \ldots, f_n)$ is a bitwisely described circuit, which also is a function from $\mathcal{M}^l$ to $\mathcal{M}$ for some integer l, and $\tau_1, \ldots, \tau_l \in \mathcal{L}$ are labels. We assume that elements in $\mathcal{M}$ are given by their unique binary encodings. The intuition is that, for $i \in [n]$, the circuit f_i has ln input wires that are associated with elements of $\{\tau_1, \ldots, \tau_l\} \times [n]$, and a bit $m_{j,k}$ of the jth input message $\bar{m}_j = \langle m_{j,1}, \ldots, m_{j,n} \rangle \in \mathcal{M}$ of $\bar{f}$ associated with τ_j is associated with the input wire of f_i corresponding to $(\tau_j, k) \in \{\tau_1, \ldots, \tau_l\} \times [n]$. Evaluating the bitwisely described program $\bar{P}$ for an input $(\bar{m}_1, \ldots, \bar{m}_l) \in \mathcal{M}^l$ means computing

$$\left\langle f_1 \left(m_{j,k}\right)_{(j,k)\in[l]\times[n]}, \ldots, f_n \left(m_{j,k}\right)_{(j,k)\in[l]\times[n]} \right\rangle$$

where $\bar{m}_j = \langle m_{j,1}, \ldots, m_{j,n} \rangle$. We also write the resulting function value as $\bar{P}\left(\bar{m}_1, \ldots, \bar{m}_l\right)$.

Let $\bar{P}_1, \ldots, \bar{P}_l$ be bitwisely described programs such that $\bar{P}_i = \left(\bar{f}_i, \tau_{i,1}, \ldots, \tau_{i,r_i}\right)$ for a bitwisely described circuit $\bar{f}_i = (f_{i,1}, \ldots, f_{i,n})$ and an integer r_i for $i \in [l]$. Also, let $P_{i,j} := \left(f_{i,j}, (\tau_{i,s}, t)_{(s,t)\in[r_i]\times[n]}\right)$ for $(i,j) \in [l] \times [n]$. For $\bar{P}_1, \ldots, \bar{P}_l$ and a bitwisely described circuit $\bar{g} = (g_1, \ldots, g_n)$, from $\mathcal{M}^l$ to $\mathcal{M}$, the *composed bitwisely described program* denoted by $\bar{g}(\bar{P}_1, \ldots, \bar{P}_l)$ is a bitwisely described program that evaluates the function $\bar{g}$ on the outputs of $\bar{P}_1, \ldots, \bar{P}_l$. More specifically, $\bar{g}(\bar{P}_1, \ldots, \bar{P}_l) = (\bar{g}^*, \tau_1, \ldots, \tau_{l^*})$ where $\bar{g}^* = (g_1^*, \ldots, g_n^*)$ and g_i^* is the circuit of the composed program $g_i\left((P_{s,t})_{(s,t)\in[l]\times[n]}\right)$ for $i \in [n]$ and $\tau_1, \ldots, \tau_{l^*}$ are all distinct labels of $\bar{P}_1, \ldots, \bar{P}_l$.

Let $\bar{P} = \left(\bar{f}, \tau_1, \ldots, \tau_l\right)$ be a bitwisely described program such that $\bar{f} = (f_1, \ldots, f_n)$. We say $\bar{P}$ is *fully bound* if, for each $i \in [l]$, there is at least one path from the input wire corresponding to (τ_i, j) to the output wire among all the paths in the circuits $f_1, \ldots, f_n$ for any $j \in [n]$. For any bitwisely described program, $\bar{P} = (\bar{f}, \tau_1, \ldots, \tau_l)$, using a graph traversal algorithm, we can always find $l' \in [l]$ such that, without loss of generality, $i > l'$ if and only if each of the input wire of (τ_i, j) in $f_1, \ldots, f_n$ does not have a path that ends with its output wire for $j \in [n]$. For $k \in [n]$, let f'_k be the circuit f_k without the wires that have a path from the input wire corresponds to (τ_i, j) for $i > l'$ and $j \in [n]$, and let $\bar{f}' = (f'_1, \ldots, f'_n)$. Then we say $\bar{P}' = (\bar{f}', \tau_1, \ldots, \tau_{l'})$ is the *fully bound bitwisely described sub-program* of $\bar{P}$. Also, $\bar{P} = \pi_1\left(\bar{P}', \bar{P}_0\right)$ for some $\bar{P}_0$.

3) Multi-labeled programs

A *multi-label* is a pair $(\Delta, \tau) \in \mathcal{D} \times \mathcal{T}$. Here, Δ is referred to as the *dataset identifier* and τ is referred to as the *data identifier*. Sometimes they are also referred to as the dataset label and the data label. In this paper, $\mathcal{D}$ is referred to as the *dataset identifier space* and $\mathcal{T}$ is referred to as the *data identifier space*. A *multi-labeled program* is a pair $P_\Delta = (\Delta, P)$, where $\Delta \in \mathcal{D}$ is a dataset identifier and $P = (f, \tau_1, \ldots, \tau_l)$ is a labeled program with data labels $\tau_1, \ldots, \tau_l \in \mathcal{T}$ as labels.

The idea is that we are considering multiple datasets, and individual data items belong to one of the datasets. For example, consider a class of students and imagine we have data on their exam scores. Imagine you have to compute the average of the exam scores. The notion 'average' corresponds to a labeled program $AVG = (average, \text{"student 1"}, \text{"student 2"}, \ldots, \text{"student } l\text{"})$, where $average$ is the mathematical function $average(m_1, \ldots, m_l) = (m_1 + \cdots + m_l)/l$, and the data label "student i" refers to the exam score of the ith student. Then, for example, the multi-labeled program $AVG_{\text{"midterm"}} = (\text{"midterm"}, AVG)$ is the average of the midterm scores, while $AVG_{\text{"final"}}$ is the average of the final exam scores.

III. HOMOMORPHIC AUTHENTICATED ENCRYPTION

Here we describe the syntax of the homomorphic authenticated encryption.

A. SYNTAX

1) Homomorphic authenticated encryption

Definition 1 (MDHAE)**.** *A multi-dataset homomorphic authenticated encryption (MDHAE) is a tuple of PPT algorithms* $(\mathrm{KeyGen}, \mathrm{Enc}, \mathrm{Eval}, \mathrm{Dec})$ *which can be described as follows:*

- $(ek, sk) \leftarrow \mathrm{KeyGen}(1^\lambda)$*: outputs a public evaluation key* ek *and a secret key* sk *for the given security parameter* λ*.*
- $c \leftarrow \mathrm{Enc}(sk, \Delta, \tau, m)$*: given* sk*, outputs a ciphertext* $c \in \mathcal{C}$ *of the message* $m \in \mathcal{M}$ *with respect to the dataset identifier* $\Delta \in \mathcal{D}$ *and the data identifier* $\tau \in \mathcal{T}$*.*
- $c \leftarrow \mathrm{Eval}(ek, f, c_1, \ldots, c_l)$*: given* ek*, deterministically outputs the homomorphically evaluated ciphertext* $c \in \mathcal{C}$ *with respect to an admissible function* $f \in \mathcal{F}$ *of arity* l *and ciperhtexts* $c_1, \ldots, c_l \in \mathcal{C}$*.*
- m *or* $\perp \leftarrow \mathrm{Dec}(sk, \Delta, P, c)$*: given* sk*, deterministically outputs the decrypted value* m *of the ciphertext* $c \in \mathcal{C}$ *with respect to the dataset identifier* $\Delta \in \mathcal{D}$ *and the admissible labeled program* P *of arity* l*, or outputs* $\perp$ *if something is wrong.*

We assume that evaluation key ek implicitly contains the information about the message space $\mathcal{M}$, the ciphertext space $\mathcal{C}$, the dataset identifier space $\mathcal{D}$, the data identifier space $\mathcal{T}$ and the admissible function space $\mathcal{F}$. As mentioned above, we assume both Eval and Dec are deterministic algorithms.

2) Compactness

An MDHAE is required to have compact ciphertexts, where the compactness is defined as follows. If $(ek, sk) \leftarrow \mathrm{KeyGen}(1^\lambda)$ for a security parameter λ, then the output size of algorithms $\mathrm{Enc}(sk, \cdot, \cdot, \cdot)$ and $\mathrm{Eval}(ek, \ldots)$ must be bounded by a polynomial in λ regardless of the choice of their inputs. This way, the ciphertext size becomes independent of the evaluated function.

3) Correctness

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

- *Correctness of the evaluation*:

$$f(m_1, \ldots, m_l) = \mathrm{Dec}(sk, \Delta, P, \mathrm{Eval}(ek, f, c_1, \ldots, c_l))$$

  holds for any λ, any ciphertexts $c_1, \ldots, c_l$ that satisfy $m_i = \mathrm{Dec}(sk, \Delta, P_i, c_i)$ for some $m_i \in \mathcal{M}$ and admissible labeled program P_i for all $i \in [l]$, the labeled program $P = f(P_1, \ldots, P_l)$, any arity-l function $f : \mathcal{M}^l \to \mathcal{M}$ such that P is admissible and $(ek, sk) \leftarrow \mathrm{KeyGen}(1^\lambda)$.
- *Projection preservation*:

$$c_i = \mathrm{Eval}(ek, \pi_i, c_1, \ldots, c_l)$$

  for any λ and any ciphertexts $c_1, \ldots, c_l$ that satisfy $m_i = \mathrm{Dec}(sk, \Delta, P_i, c_i)$ for some $m_i \in \mathcal{M}$, admissible labeled program P_i for all $i \in [l]$ and $(ek, sk) \leftarrow \mathrm{KeyGen}(1^\lambda)$, where π_i is the ith projection function over $\mathcal{M}^l$.

Note that it follows that an MDHAE also satisfies the *correctness of encryption*:

$$m = \mathrm{Dec}(sk, \Delta, (\mathrm{id}, \tau), \mathrm{Enc}(sk, \Delta, \tau, m))$$

holds for the identity function id.

Remark 1. *Suppose* $m = \mathrm{Dec}(sk, \Delta, P, c)$ *for an MDHAE where* $(ek, sk) \leftarrow \mathrm{KeyGen}(1^\lambda)$*. If we let* P' *be the fully bound sub-program of* P*, then from the correctness properties of an MDHAE, we see that* $m = \mathrm{Dec}(sk, \Delta, P', c)$*. Similarly, if* $\perp = \mathrm{Dec}(sk, \Delta, P, c)$ *then* $\perp = \mathrm{Dec}(sk, \Delta, P', c)$*.*

4) Efficient decryption

We say that an MDHAE *supports efficient decryption*, if there exist two additional algorithms Prep and EffDec such that:

- $sk_P \leftarrow \mathrm{Prep}(sk, P)$: given sk and the admissible program P, deterministically outputs a decryption key sk_P for P. Note that this does not involve any dataset identifier $\Delta \in \mathcal{D}$.
- m or $\perp \leftarrow \mathrm{EffDec}(sk_P, \Delta, c)$: deterministically outputs the decrypted value $m \in \mathcal{M}$ of the ciphertext $c \in \mathcal{C}$ with respect to the decryption key sk_P for an admissible program P and the dataset identifier $\Delta \in \mathcal{D}$, or outputs $\perp$ if something is wrong.

The above algorithms are required to satisfy the following properties.

- Correctness: $\mathrm{EffDec}(sk_P, \Delta, c) = \mathrm{Dec}(sk, \Delta, P, c)$ for any dataset identifier $\Delta \in \mathcal{D}$ and ciphertext $c \in \mathcal{C}$, when $(ek, sk) \leftarrow \mathrm{KeyGen}(1^\lambda)$ and $sk_P \leftarrow \mathrm{Prep}(sk, P)$ for some admissible program P.
- Amortized efficiency: let P be an admissible program. Let $(m_1, \ldots, m_l) \in \mathcal{M}^l$ be an input for P, and let $t(l)$ be the time required to compute $P(m_1, \ldots, m_l)$. For $sk_P \leftarrow \mathrm{Prep}(sk, P)$, the time required for $\mathrm{EffDec}(sk_P, \Delta, c)$ is $o(t(l))$.

B. A NEW SECURITY NOTION FOR MULTI-DATASET HOMOMORPHIC AUTHENTICATED ENCRYPTIONS

We define a new security notion of an MDHAE H using the security game $\mathbf{Game}^{\mathrm{MDHAE}}$ below. Unlike the original definitions given in Joo and Yun [5], this is an 'all-in-one' definition, combining both privacy and authenticity of an MDHAE.

Let $\$(\cdot, \cdot, \cdot, \cdot)$ be an algorithm such that $\$(sk, \Delta, \tau, m)$ samples and outputs a ciphertext c from a distribution $C_{sk,\Delta,\tau}$ over the ciphertext space $\mathcal{C}$ of an HAE H. So, the output of $\$(sk, \Delta, \tau, m)$ is independent of m. Let $\perp(\cdot, \cdot, \cdot)$ be the trivial algorithm that outputs $\perp$ for any input.

The challenger only answers *non-redundant* decryption queries. The definition of a non-redundant decryption query comes after the main definition.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3100852, IEEE Access

$\mathbf{Game}^{\mathrm{MDHAE}}_{H,\$,A}(\lambda)$:

Initialization

A key pair $(ek, sk) \leftarrow H.\mathrm{KeyGen}(1^\lambda)$ is generated, and ek is given to A. The challenger initializes a set S with $\emptyset$, and flips a coin $b \xleftarrow{\$} \{0,1\}$. If $b = 0$, then the challenger defines oracles E and D as $E = H.\mathrm{Enc}(sk, \cdot, \cdot, \cdot)$ and $D = H.\mathrm{Dec}(sk, \cdot, \cdot, \cdot)$. Otherwise, the challenger lets $E = \$(sk, \cdot, \cdot, \cdot)$ and $D = \perp(\cdot, \cdot, \cdot)$. The adversary is given oracle access to E and D. For convenience, E-queries and D-queries are also called as encryption queries and decryption queries, respectively.

Queries

A makes encryption and decryption queries adaptively. The queries are handled as follows.

- For each encryption query (Δ, τ, m), if $(\Delta, \tau, \cdot, \cdot) \notin S$, then the challenger returns the answer $c \leftarrow E(\Delta, \tau, m)$ to A, and update $S \leftarrow S \cup \{(\Delta, \tau, m, c)\}$. Otherwise, the challenger rejects the query.
- For each decryption query (Δ, P, c), if P is admissible and the query is non-redundant with respect to S, then the challenger returns the answer m or $\perp \leftarrow D(\Delta, P, c)$ to A. Otherwise, the challenger rejects the query.

Finalization

A outputs a bit b'. The challenger returns 1 if $b = b'$, and 0 otherwise.

The *advantage* of the adversary A in the game $\mathbf{Game}^{\mathrm{MDHAE}}$ for the scheme H with respect to the algorithm \$ is defined as

$$\mathbf{Adv}^{\mathrm{MDHAE}}_{H,\$,A}(\lambda) := \left| \mathbf{Pr}\left[\mathbf{Game}^{\mathrm{MDHAE}}_{H,\$,A}(\lambda) = 1\right] - \frac{1}{2} \right|.$$

We say that an MDHAE H is *secure* if there exists an algorithm \$ such that the advantage $\mathbf{Adv}^{\mathrm{MDHAE}}_{H,\$,A}(\lambda)$ is negligible for any PPT adversary A.

Let (Δ, P, c) be a decryption query. Let $P' = (f', \tau'_1, \ldots, \tau'_{l'})$ be the fully bound sub-program of P. The decryption query (Δ, P, c) is *redundant* with respect to S if $(\Delta, \tau'_i, m'_i, c'_i) \in S$ for some (unique) $m'_i \in \mathcal{M}$ and $c'_i \in \mathcal{C}$ for all $i \in [l']$ and $c = H.\mathrm{Eval}(ek, f', c'_1, \ldots, c'_{l'})$. And we say that a decryption query is *non-redundant* if it is not redundant.

In other words, the decryption queries that can trivially distinguishes $H.\mathrm{Dec}(sk, \cdot, \cdot, \cdot)$ and $\perp(\cdot, \cdot, \cdot)$ are not allowed; for a redundant query (Δ, P, c) as above, it is guaranteed that the response for the query is $f(m'_1, \ldots, m'_{l'}) \neq \perp$, when $D = H.\mathrm{Dec}(sk, \cdot, \cdot, \cdot)$.

IV. HOMOMORPHIC ENCRYPTION

A. SYNTAX

1) Homomorphic encryption

Definition 2 (HE). *A homomorphic encryption (HE) is a tuple of PPT algorithms* $(\mathrm{KeyGen}, \mathrm{Enc}, \mathrm{Eval}, \mathrm{Dec})$ *as follows:*

- $(ek, sk) \leftarrow \mathrm{KeyGen}(1^\lambda)$*: outputs a public evaluation key* ek *and a secret key* sk *for the given security parameter* λ*.*
- $c \leftarrow \mathrm{Enc}(sk, m)$*: given* sk*, outputs a ciphertext* $c \in \mathcal{C}$ *of the message* $m \in \mathcal{M}$*.*
- $c \leftarrow \mathrm{Eval}(ek, f, c_1, \ldots, c_l)$*: given* ek*, deterministically outputs the homomorphically evaluated ciphertext* $c \in \mathcal{C}$ *with respect to an admissible function* $f \in \mathcal{F}$ *of arity* l *and* $c_1, \ldots, c_l \in \mathcal{C}$*.*
- $m \leftarrow \mathrm{Dec}(sk, c)$*: given* sk*, deterministically outputs the decrypted value* m *of the ciphertext* $c \in \mathcal{C}$*.*

We assume that evaluation key ek implicitly contains the information about the message space $\mathcal{M}$, the ciphertext space $\mathcal{C}$ and the admissible function space $\mathcal{F}$. As mentioned above, we assume both Eval and Dec are deterministic algorithms.

2) Compactness

An HE is required to have compact ciphertexts, where the compactness is defined as follows. If $(ek, sk) \leftarrow \mathrm{KeyGen}(1^\lambda)$ for a security parameter λ, then the output size of algorithms $\mathrm{Enc}(sk, \cdot)$ and $\mathrm{Eval}(ek, \ldots)$ must be bounded by a polynomial in λ regardless of the choice of their inputs. This way, the ciphertext size becomes independent of the evaluated function.

3) Correctness

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

- *Correctness of the evaluation*:

  $$f(m_1, \ldots, m_l) = \mathrm{Dec}(sk, \mathrm{Eval}(ek, f, c_1, \ldots, c_l))$$

  holds for any λ, any ciphertexts $c_1, \ldots, c_l$ such that $c_i \leftarrow \mathrm{Eval}(ek, f_i, c_{i,1}, \ldots, c_{i,t_i})$ for some admissible circuit $f_i \in \mathcal{F}$ and ciphertexts $c_{i,j} \leftarrow \mathrm{Enc}(sk, m_{i,j})$ that satisfies $m_i = f_i(m_{i,1}, \ldots, m_{i,t_i})$ for some $m_{i,j} \in \mathcal{M}$ for $(i,j) \in [l] \times [t_i]$ and any circuit $f \in \mathcal{F}$ such that $f^* \in \mathcal{F}$ where f^* is a circuit that evaluates f on the outputs of circuits $f_1, \ldots, f_l$ when $(ek, sk) \leftarrow \mathrm{KeyGen}(1^\lambda)$.
- *Projection preservation*:

  $$c_i = \mathrm{Eval}(ek, \pi_i, c_1, \ldots, c_l)$$

  and the circuit for $\mathrm{Eval}(ek, \pi_i, \ldots)$ is also the bitwisely described ith projection over $\mathcal{C}^l$ for any λ and any ciphertexts $c_1, \ldots, c_l$ that satisfy $m_i = \mathrm{Dec}(sk, c_i)$ for some $m_i \in \mathcal{M}$ for all $i \in [l]$ when $(ek, sk) \leftarrow \mathrm{KeyGen}(1^\lambda)$ and π_i is the ith projection function over $\mathcal{M}^l$.

Note that it follows that an HE also satisfies the *correctness of encryption*: $m = \mathrm{Dec}(sk, \mathrm{Enc}(sk, m))$ holds for the identity function id since $\mathrm{Eval}(ek, \mathrm{id}, \mathrm{Enc}(sk, m)) = \mathrm{Enc}(sk, m)$.

Remark 2. *Unlike conventional correctness property, we additionally require projection preservation. But some (leveled)*

fully homomorphic encryption already satisfies the projection preservation [6].

B. A SECURITY NOTION FOR HOMOMORPHIC ENCRYPTIONS

We define a security notion of an HE K using the security game $\mathbf{Game}^{\mathrm{HE}}$ below. For ease of comparison, we change some formalisms from conventional IND-CPA security definition, but the equivalence of two formulations can be proven.

Let $\$(\cdot,\cdot)$ be an algorithm such that $\$(sk, m)$ samples and outputs a ciphertext c from a distribution C_{sk} over the ciphertext space $\mathcal{C}$ of an HE K. So, the output of $\$(sk, m)$ is independent of m.

$\mathbf{Game}^{\mathrm{HE}}_{K,\$,A}(\lambda)$:

Initialization

A key pair $(ek, sk) \leftarrow K.\mathrm{KeyGen}(1^\lambda)$ is generated, and ek is given to A. The challenger flips a coin $b \xleftarrow{\$} \{0,1\}$. If $b = 0$, then the challenger defines an oracle E as $E = K.\mathrm{Enc}(sk,\cdot)$. Otherwise, the challenger lets $E = \$(sk,\cdot)$. The adversary is given oracle access to E, and for convenience, E-queries are also called as encryption queries.

Queries

A makes encryption queries adaptively. For each encryption query $m \in \mathcal{M}$, the challenger returns the answer $c \leftarrow E(m)$ to A.

Finalization

A outputs a bit b'. The challenger returns 1 if $b = b'$, and 0 otherwise.

The *advantage* of the adversary A in the game $\mathbf{Game}^{\mathrm{HE}}$ for the scheme K with respect to the algorithm $\$$ is defined as

$$\mathbf{Adv}^{\mathrm{HE}}_{K,\$,A}(\lambda) := \left|\mathbf{Pr}\left[\mathbf{Game}^{\mathrm{HE}}_{K,\$,A}(\lambda) = 1\right] - \frac{1}{2}\right|.$$

We say that an HE K is *secure* if there exists an algorithm $\$$ such that the advantage $\mathbf{Adv}^{\mathrm{HE}}_{K,\$,A}(\lambda)$ is negligible for any PPT adversary A.

V. HOMOMORPHIC AUTHENTICATOR

A. SYNTAX

Definition 3 (MDHA)**.** *A multi-dataset homomorphic authenticator (MDHA) is a tuple of PPT algorithms* $(\mathrm{KeyGen}, \mathrm{Auth}, \mathrm{Eval}, \mathrm{Verify})$ *as follows:*

- $(ek, sk) \leftarrow \mathrm{KeyGen}(1^\lambda)$: *outputs a public key pk and a secret key sk for the given security parameter λ.*
- $\sigma \leftarrow \mathrm{Auth}(sk, \Delta, \tau, m)$: *given sk, outputs an authentication tag (or sometimes just called a tag) $\sigma \in \Sigma$ of the message $m \in \mathcal{M}$ with respect to the dataset identifier $\Delta \in \mathcal{D}$ and the data identifier $\tau \in \mathcal{T}$.*
- $\sigma \leftarrow \mathrm{Eval}(ek, f, (m_1, \sigma_1), \ldots, (m_l, \sigma_l))$: *given ek, deterministically outputs the homomorphically evaluated authentication tag $\sigma \in \Sigma$ of the message $m = f(m_1, \ldots, m_l)$ with respect to an admissible function $f \in \mathcal{F}$ of arity l and tags $\sigma_1, \ldots, \sigma_l \in \Sigma$ of the messages $m_1, \ldots, m_l \in \mathcal{M}$, respectively.*
- $b \leftarrow \mathrm{Verify}(sk, \Delta, P, m, \sigma)$*: given sk, deterministically outputs the acceptance bit b of a tag $\sigma \in \Sigma$ with respect to the dataset identifier $\Delta \in \mathcal{D}$, the message $m \in \mathcal{M}$ and the admissible labeled program P of arity l.*

We assume that keys ek and sk implicitly contains the information about the message space $\mathcal{M}$, the tag space Σ, the dataset identifier space $\mathcal{D}$, the data identifier space $\mathcal{T}$ and the admissible function space $\mathcal{F}$.

1) Compactness

An MDHA is required to have compact authentication tags, where the compactness is defined as follows. If $(ek, sk) \leftarrow \mathrm{KeyGen}(1^\lambda)$ for a security parameter λ, then the output size of algorithms $\mathrm{Auth}(sk, \ldots)$ and $\mathrm{Eval}(ek, \ldots)$ must be bounded by a polynomial in λ regardless of the choice of their inputs. This way, the tag size becomes independent of the evaluated function.

2) Correctness

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

- *Correctness of the evaluation*: For a homomorphically evaluated tag $\sigma \leftarrow \mathrm{Eval}(ek, f, (m_1, \sigma_1), \ldots, (m_l, \sigma_l))$,

$$1 \leftarrow \mathrm{Verify}(sk, \Delta, P, f(m_1, \ldots, m_l), \sigma)$$

holds for any λ, any tags $\sigma_1, \ldots, \sigma_l \in \Sigma$ such that $1 = \mathrm{Verify}(sk, \Delta, P_i, m_i, \sigma_i)$ for a dataset identifier Δ, a message $m_i \in \mathcal{M}$ and an admissible program P_i for $i \in [l]$, the labeled program $P = f(P_1, \ldots, P_l)$ and any arity-l function $f : \mathcal{M}^l \to \mathcal{M}$ such that P is admissible when $(ek, sk) \leftarrow \mathrm{KeyGen}(1^\lambda)$.

- *Projection preservation*:

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

for any λ and any tags $\sigma_1, \ldots, \sigma_l \in \Sigma$ that satisfy $1 = \mathrm{Verify}(sk, \Delta, P_i, m_i, \sigma_i)$ for some $m_i \in \mathcal{M}$ and some admissible labeled program P_i for all $i \in [l]$ when $(ek, sk) \leftarrow \mathrm{KeyGen}(1^\lambda)$, where π_i is the ith projection function over $\mathcal{M}^l$.

3) Efficient verification

We say that an MDHA *supports efficient verification*, if there exist two additional deterministic algorithms Prep and $\mathrm{EffVerify}$ such that:

- $sk_P \leftarrow \mathrm{Prep}(sk, P)$: given sk and the admissible program P, deterministically outputs a verification key sk_P for P. Note that this does not involve any dataset identifier $\Delta \in \mathcal{D}$.
- $b \leftarrow \mathrm{EffVerify}(sk_P, \Delta, m, \sigma)$: deterministically outputs the acceptance bit b of the authentication tag σ with respect to the verification key sk_P for P, dataset identifier $\Delta \in \mathcal{D}$ and a message $m \in \mathcal{M}$.

The above algorithms are required to satisfy the following properties.

- Correctness: For any dataset identifier $\Delta \in \mathcal{D}$ and a tag $\sigma \in \Sigma$, $\mathrm{EffVerify}(sk_P, \Delta, m, \sigma) = \mathrm{Verify}(sk, \Delta, P, m, \sigma)$ when $(ek, sk) \leftarrow \mathrm{KeyGen}(1^\lambda)$ and $sk_P \leftarrow \mathrm{Prep}(sk, P)$ for some admissible program P.
- Amortized efficiency: let P be an admissible program. Let $(m_1, \ldots, m_l) \in \mathcal{M}^l$ be an input for P, and let $t(l)$ be the time required to compute $P(m_1, \ldots, m_l)$. For $sk_P \leftarrow \mathrm{Prep}(sk, P)$, the time required for $\mathrm{EffVerify}(sk_P, \Delta, m, \sigma)$ is $o(t(l))$.

4) Multi-dataset fully homomorphic authenticator

We say that T is multi-dataset leveled fully homomorphic authenticator (MDFHA) if the admissible function space $\mathcal{F}$ is defined as

$$\mathcal{F} = \{f \mid f : \mathcal{M}^l \to \mathcal{M} \text{ is a circuit of depth at most } d \text{ for some } l = \mathsf{poly}(\lambda)\}$$

for some $d = \mathsf{poly}(\lambda)$.

5) Bitwisely evaluable MDHA

We say that an MDHA T is *bitwisely evaluable* (or T is a BE-MDHA) if the algorithms $T.\mathrm{Eval}$ and $T.\mathrm{Verify}$ takes a bitwisely described circuit and a bitwisely described program as a part of their inputs instead of an ordinary circuit and an ordinary program, respectively. Also, selective security, correctness and the support for efficient verification can be defined as above by replacing circuits and programs to bitwisely described counterparts. Moreover, we say that T is a bitwisely evaluable multi-dataset leveled fully homomorphic authenticator (BE-MDFHA) if the admissible function space $\mathcal{F}$ is defined as

$$\mathcal{F} = \{\bar{f} \mid \bar{f} : \mathcal{M}^l \to \mathcal{M} \text{ is a bitwisely described circuit of depth at most } d \text{ for some } l = \mathsf{poly}(\lambda)\}$$

for some $d = \mathsf{poly}(\lambda)$.

Remark 3. *Since most homomorphic encryption has exponential ciphertext space and our generic construction for FHAE uses encrypt-then-authenticate structure, we need an homomorphic authenticator scheme that can authenticate messages with exponential size. But, to our knowledge, there is no direct MDHA construction that has exponentially large message space. Therefore, we define BE-MDHA as above and suggest a generic construction of BE-MDHA using existing MDHA with message space $\{0,1\}$ in Section VII.*

6) OR-homomorphic MDHA

We say that an MDHA T is *OR-homomorphic* if the admissible function space $\mathcal{F}$ is defined as

$$\mathcal{F} = \{f \mid f : \mathcal{M}^l \to \mathcal{M} \text{ is a circuit of depth } d \text{ for some } l = \mathsf{poly}(\lambda) \text{ such that all the gates of } f \text{ are OR gates}\}$$

for some $d = \mathsf{poly}(\lambda)$.

B. A SECURITY NOTION FOR MULTI-DATASET HOMOMORPHIC AUTHENTICATORS

We define a security notion of an MDHA T using the security game $\mathbf{Game}^{\overline{\mathrm{MDHA}}}$ below. Unlike the security definition of a homomorphic signature in Gorbunov, Vaikuntanathan, and Wichs [4], our security definition is over MDHA that can authenticate for freely chosen data identifiers. Also, unlike previous security notions above, we define selective security of an MDHA; for our purposes, the selective security of MDHA is enough.

$\mathbf{Game}^{\overline{\mathrm{MDHA}}}_{T,A}(\lambda)$:

Selective Queries

A makes authentication queries selectively. A selects polynomially many queries $((\Delta_i, \tau_i, m_i))_{i\in[q]}$ and sends $((\Delta_i, \tau_i, m_i))_{i\in[q]}$ to the challenger.

Initialization and Response

If $(\Delta_i, \tau_i) = (\Delta_j, \tau_j)$ for some $i \neq j$, then the challenger rejects the query. If $(\Delta_i, \tau_i) \neq (\Delta_j, \tau_j)$ for any $i \neq j$, then the challenger generates a key pair $(ek, sk) \leftarrow T.\mathrm{KeyGen}(1^\lambda)$ and computes $\sigma_i \leftarrow T.\mathrm{Auth}(sk, \Delta_i, \tau_i, m_i)$ and sends (ek, S) to A where $S = \{(\Delta_i, \tau_i, m_i, \sigma_i)\}_{i\in[q]}$.

Finalization

A outputs a forgery attempt $(\Delta^*, P^*, m^*, \sigma^*)$ such that for the fully bound sub-program $P^{*\prime} = (f^*, \tau_1^*, \ldots, \tau_l^*)$ of P^* where $f^* \in \mathcal{F}$, there is $(\Delta^*, \tau_i^*, m_i^*, \sigma_i^*) \in S$ for some (unique) $m_i^* \in \mathcal{M}$ and $\sigma_i^* \in \Sigma$ for all $i \in [l]$. If $1 \leftarrow T.\mathrm{Verify}(sk, \Delta^*, P^{*\prime}, m^*, \sigma^*)$ and $(m^*, \sigma^*) \neq (m^{**}, \sigma^{**})$ where $m^{**} = f^*(m_1^*, \ldots, m_l^*)$ and $\sigma^{**} \leftarrow T.\mathrm{Eval}(ek, f^*, (m_1^*, \sigma_1^*), \ldots, (m_l^*, \sigma_l^*))$, then the challenger outputs 1. Otherwise, the challenger outputs 0.

The *advantage* of the adversary A in the game $\mathbf{Game}^{\overline{\mathrm{MDHA}}}$ for the scheme T is defined as

$$\mathbf{Adv}^{\overline{\mathrm{MDHA}}}_{T,A}(\lambda) := \mathbf{Pr}\left[\mathbf{Game}^{\overline{\mathrm{MDHA}}}_{T,A}(\lambda) = 1\right].$$

We say that an MDHA T is *selectively secure* if the advantage $\mathbf{Adv}^{\overline{\mathrm{MDHA}}}_{T,A}(\lambda)$ is negligible for any PPT adversary A.

VI. GENERIC CONSTRUCTION OF SECURE FULLY HOMOMORPHIC AUTHENTICATED ENCRYPTION

A. OVERVIEW OF OUR CONSTRUCTION

In this section, we construct a secure FHAE. Our FHAE scheme is a generic construction using one FHE scheme and two HA schemes, one fully homomorphic and one OR-homomorphic. Overall, our construction can be regarded as following the encrypt-then-authenticate paradigm, but with one additional HA to meet our new security notion.

Before giving construction directly, we give a high-level overview of how we created our construction and some preliminary definitions.

First, let H_0 be a straightforward encrypt-then-authenticate construction made out of a FHE K and an MDFHA $\bar{T}$, as

well as additional random functions $\bar{F}_0$ and $\bar{G}_0$: H_0 encrypts an input (Δ, τ, m) as $(\mathring{c}, \bar{\sigma}) \leftarrow H_0.\mathrm{Enc}(sk_0, \Delta, \tau, m)$, where $\mathring{c} \leftarrow K.\mathrm{Enc}(\mathring{sk}, m)$ and $\bar{\sigma} \leftarrow \bar{T}.\mathrm{Auth}(\bar{sk}, \bar{\Delta}, \bar{\tau}, \mathring{c})$ for $\bar{\Delta} := \bar{F}_0(\Delta)$ and $\bar{\tau} := \bar{G}_0(\tau)$.

For homomorphic evaluation of H_0, we define $H_0.\mathrm{Eval}(ek_0, f, \dots)$ as applying the homomorphic evaluation algorithm $K.\mathrm{Eval}(\mathring{ek}, f, \dots)$ to the first parts of its input ciphertexts, encryptions of K, and the algorithm $\bar{T}.\mathrm{Eval}(\bar{ek}, K.\mathrm{Eval}(\mathring{ek}, f, \dots), \dots)$ to the second parts, the tags of the first parts. The decryption algorithm $H_0.\mathrm{Dec}(sk_0, \Delta, P, c_0)$ parses $P = (f, \tau_1, \dots, \tau_l)$ and $c_0 = (\mathring{c}, \bar{\sigma})$, and outputs $K.\mathrm{Dec}(\mathring{sk}, \mathring{c})$ when $1 \leftarrow \bar{T}.\mathrm{Verify}(\bar{sk}, \bar{\Delta}, \bar{P}, \mathring{c}, \bar{\sigma})$, $\perp$ otherwise where $\bar{P} = (K.\mathrm{Eval}(\mathring{ek}, f, \dots), \bar{\tau}_1, \dots, \bar{\tau}_l)$.

The construction H_0 is comparable to the generic construction of adaptively secure fully homomorphic signature given by Gorbunov, Vaikuntanathan and Wichs [4]. Their basic scheme satisfies only the selective security, but, by using their homomorphic trapdoor function as a homomorphic version of a chameleon hashing [12], they constructed an adaptively secure fully homomorphic signature scheme. In our construction, the inner FHE serves two purposes: the first is to encrypt the message to achieve privacy, the second is, like a chameleon hashing, to upgrade the authentication security from selective to adaptive.

Now, let us observe the security of H_0. If an FHE K is secure, then there is an algorithm $\$_K$ that is indistinguishable from $K.\mathrm{Enc}$. Therefore, if we define H_0' to be the same as H_0 except using $\$_K$ instead of $K.\mathrm{Enc}$, then H_0' and H_0 are also indistinguishable. Note that $H_0'.\mathrm{Enc}$ randomizes its inputs with $\bar{F}_0$, $\bar{G}_0$ and $\$_K$, and is independent of its message input. Using lazy sampling on random functions $\bar{F}_0$ and $\bar{G}_0$, one can answer adaptive queries of $H_0'.\mathrm{Enc}$ using selective queries of $\bar{T}.\mathrm{Auth}$ with random inputs $(\bar{\Delta}, \bar{\tau}, \$_K(\mathring{ek}, \cdot))$. One can view the attacks on H_0' distinguishing $H_0'.\mathrm{Dec}(\cdot)$ and $\perp(\cdot)$ as attacks on $\bar{T}$ with randomly chosen selective queries. Therefore, the security of $\bar{T}$ ensures that it is hard for adversaries to output an accepting decryption query (Δ, P, c) such that $P = (f, \tau_1, \dots, \tau_l)$ and there has been encryption queries with respect to (Δ, τ_i) for all $i \in [l]$. In other words, we have a form of adaptive security for authenticity, as long as the 'forgery attempt' $(\Delta, (f, \tau_1, \dots, \tau_l), c)$ does not have any 'empty slot'. However, we need also to take care of the cases when some slots are empty. One way to do this is to ensure that there are no empty slots. We might use a hash tree for this purpose, as in [3] and [5], but that technique is not compatible with multi-dataset. Instead, we will adapt a technique using an OR-homomorphic authenticator given by Catalano, Fiore, and Nizzardo [10].

The idea is to modify the plain encrypt-then-authenticate construction H_0 so that for each ciphertext we add an authentication tag of 0. When we perform homomorphic evaluation, we need also to homomorphically evaluate the authentication tag as well. When we evaluate a unary gate, we will homomorphically evaluate the identity gate for the authentication of 0. When we evaluate a binary gate, we will homomorphically evaluate logical OR of the corresponding two zeroes. Therefore, for unaltered ciphertexts and their homomorphic evaluation, the additional tags are all authentication tags of zeroes. During the decryption, we verify the additional authentication tag as well. The MDHA $\check{T}$ we use for the authentication of 0 can be OR-homomorphic, and since the message 0 is fixed and we may randomize the labels as in the construction of H_0, $\check{T}$ needs only to be selectively secure.

Now, suppose that the adversary makes a decryption query (Δ, P, c) with $P = (f, \tau_1, \dots, \tau_l)$. Without loss of generality, let us assume that P is fully bound. The basic intuition is that, if any of the labels τ_{i^*} is empty, that is, no message was encrypted with respect to (Δ, τ_{i^*}), then the reduction algorithm may guess the position i^*, and replace the additional authentication of 0 at (Δ, τ_{i^*}) with the authentication of 1. Since we will always evaluate ORs of inputs, the final value of this circuit made out of OR gates will be 1. Then, any such decryption query (with the additional tag for 0 verified) would produce a forgery of $\check{T}$. Hence, if $\check{T}$ is secure, then it is infeasible to produce a valid decryption query with an empty slot.

Our MDFHAE can be instantiated with existing schemes. For example, some popular FHE schemes like [2], [6] can be used as K, and the first fully homomorphic signature [4] can be used as $\check{T}$. As for $\bar{T}$, since there is no dedicated MDFHA scheme with exponential message space, one can follow our generic construction, Construction 3, for BE-MDHA in Section VII to construct such scheme. Using the first fully homomorphic signature [4] and Construction 3, one can construct a secure BE-MDFHA and it can be used as $\bar{T}$.

Before describing the construction, we need a preparation. Suppose a circuit $f : \mathcal{M}^l \to \mathcal{M}$ is given. We define the corresponding boolean circuit $\check{f} : \{0,1\}^l \to \{0,1\}$ as the circuit obtained by replacing each unary gate of f with the identity gate (sending a bit b to b itself), and each binary gate of f with the OR gate.

B. GENERIC CONSTRUCTION

Construction 1. *Let K be an HE scheme, $\bar{T}$ be a BE-MDHA scheme and $\check{T}$ be an OR-homomorphic MDHA. Let $\mathring{\mathcal{M}}$, $\mathring{\mathcal{C}}$, $\mathring{\mathcal{F}}$ be the message space, the ciphertext space, and the admissible function space of K, where every element in $\mathring{\mathcal{C}}$ is given as an n bit binary encoding. Similarly, let $\bar{\mathcal{M}} := \mathring{\mathcal{C}}$, $\bar{\Sigma}$, $\bar{\mathcal{D}}$, $\bar{\mathcal{T}}$, $\bar{\mathcal{F}}$ be the message space, the tag space, the dataset identifier space, the data identifier space, and the admissible (bitwisely described) function space of $\bar{T}$. Also, let $\check{\mathcal{M}} := \{0,1\}$, $\check{\Sigma}$, $\check{\mathcal{D}}$, $\check{\mathcal{T}}$, $\check{\mathcal{F}}$ be the corresponding ones of $\check{T}$. Let $\mathcal{M} := \mathring{\mathcal{M}}$, $\mathcal{C} := \mathring{\mathcal{C}} \times \bar{\Sigma} \times \check{\Sigma}$, $\mathcal{D}$, $\mathcal{T}$, $\mathcal{F}$ be the message space, the ciphertext space, the dataset identifier space, the data identifier space, and the admissible function space of the FHAE H below where*

$$\mathcal{F} := \{ f \in \mathring{\mathcal{F}} \mid \check{f} \in \check{\mathcal{F}} \text{ and } \bar{f} \in \bar{\mathcal{F}} \text{ where } \bar{f} \text{ is the bitwisely described circuit of } K.\mathrm{Eval}(\mathring{ek}, f, \dots) \text{ for any choices of } (\mathring{ek}, \mathring{sk}) \leftarrow K.\mathrm{KeyGen}(1^\lambda) \}.$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3100852, IEEE Access

Let $\bar{F} : \{0,1\}^\lambda \times \mathcal{D} \to \bar{\mathcal{D}}$, $\bar{G} : \{0,1\}^\lambda \times \mathcal{T} \to \bar{\mathcal{T}}$, $\check{F} : \{0,1\}^\lambda \times \mathcal{D} \to \check{\mathcal{D}}$ and $\check{G} : \{0,1\}^\lambda \times \mathcal{T} \to \check{\mathcal{T}}$ be secure PRFs.

Using K, $\bar{T}$, $\check{T}$, $\bar{F}$, $\bar{G}$, $\check{F}$, $\check{G}$ we construct an MDHAE H as follows.

- $H.\mathrm{KeyGen}(1^\lambda)$: *let* $(\mathring{ek}, \mathring{sk}) \leftarrow K.\mathrm{KeyGen}(1^\lambda)$, $(\bar{ek}, \bar{sk}) \leftarrow \bar{T}.\mathrm{KeyGen}(1^\lambda)$, $(\check{ek}, \check{sk}) \leftarrow \check{T}.\mathrm{KeyGen}(1^\lambda)$, $k_{\bar{F}}, k_{\bar{G}}, k_{\check{F}}, k_{\check{G}} \leftarrow \{0,1\}^\lambda$, *and output* $(ek, sk) := (\mathring{ek}\|\bar{ek}\|\check{ek}, \mathring{ek}\|\mathring{sk}\|\bar{sk}\|\check{sk}\|k_{\bar{F}}\|k_{\bar{G}}\|k_{\check{F}}\|k_{\check{G}})$.
- $H.\mathrm{Enc}(sk, \Delta, \tau, m)$: *parse given input* $sk = \mathring{ek}\|\mathring{sk}\|\bar{sk}\|\check{sk}\|k_{\bar{F}}\|k_{\bar{G}}\|k_{\check{F}}\|k_{\check{G}}$, *let* $\mathring{c} \leftarrow K.\mathrm{Enc}(\mathring{sk}, m)$, $\bar{\Delta} \leftarrow \bar{F}(k_{\bar{F}}, \Delta)$, $\bar{\tau} \leftarrow \bar{G}(k_{\bar{G}}, \tau)$, $\check{\Delta} \leftarrow \check{F}(k_{\check{F}}, \Delta)$, $\check{\tau} \leftarrow \check{G}(k_{\check{G}}, \tau)$, $\bar{\sigma} \leftarrow \bar{T}.\mathrm{Auth}(\bar{sk}, \bar{\Delta}, \bar{\tau}, \mathring{c})$, $\check{\sigma} \leftarrow \check{T}.\mathrm{Auth}(\check{sk}, \check{\Delta}, \check{\tau}, 0)$. *Output* $c := (\mathring{c}, \bar{\sigma}, \check{\sigma})$.
- $H.\mathrm{Eval}(ek, f, c_1, \ldots, c_l)$: *parse given input* $ek = \mathring{ek}\|\bar{ek}\|\check{ek}$. *For* $i = 1, \ldots, l$, *parse* $c_i = (\mathring{c}_i, \bar{\sigma}_i, \check{\sigma}_i)$. *Evaluate a ciphertext* $\mathring{c} \leftarrow K.\mathrm{Eval}(\mathring{ek}, f, \mathring{c}_1, \ldots, \mathring{c}_l)$, *and authentication tags* $\bar{\sigma} \leftarrow \bar{T}.\mathrm{Eval}(\bar{ek}, \bar{f}, \bar{\sigma}_1, \ldots, \bar{\sigma}_l)$, $\check{\sigma} \leftarrow \check{T}.\mathrm{Eval}(\check{ek}, \check{f}, \check{\sigma}_1, \ldots, \check{\sigma}_l)$ *where* $\bar{f}$ *is the bitwisely described circuit of the deterministic algorithm* $K.\mathrm{Eval}(\mathring{ek}, f, \ldots)$. *Output* $c := (\mathring{c}, \bar{\sigma}, \check{\sigma})$.
- $H.\mathrm{Dec}(sk, \Delta, P, c)$: *parse given inputs* $sk = \mathring{ek}\|\mathring{sk}\|\bar{sk}\|\check{sk}\|k_{\bar{F}}\|k_{\bar{G}}\|k_{\check{F}}\|k_{\check{G}}$, $P = (f, \tau_1, \ldots, \tau_l)$ *and* $c = (\mathring{c}, \bar{\sigma}, \check{\sigma})$. *Let* $\bar{\Delta} \leftarrow \bar{F}(k_{\bar{F}}, \Delta)$, $\check{\Delta} \leftarrow \check{F}(k_{\check{F}}, \Delta)$. *For* $i \in [l]$, *let* $\bar{\tau}_i \leftarrow \bar{G}(k_{\bar{G}}, \tau_i)$, $\check{\tau}_i \leftarrow \check{G}(k_{\check{G}}, \tau_i)$. *Now, let* $\bar{P} = (\bar{f}, \bar{\tau}_1, \ldots, \bar{\tau}_l)$, $\check{P} = (\check{f}, \check{\tau}_1, \ldots, \check{\tau}_l)$ *where* $\bar{f}$ *is the bitwisely described circuit of the deterministic algorithm* $K.\mathrm{Eval}(\mathring{ek}, f, \ldots)$. *If* $1 \leftarrow \bar{T}.\mathrm{Verify}(\bar{sk}, \bar{\Delta}, \bar{P}, \mathring{c}, \bar{\sigma})$ *and* $1 \leftarrow \check{T}.\mathrm{Verify}(\check{sk}, \check{\Delta}, \check{P}, 0, \check{\sigma})$, *then output* $m \leftarrow K.\mathrm{Dec}(\mathring{sk}, \mathring{c})$. *Otherwise, output* $\perp$.

Remark 4. *Construction 1 satisfies the correctness properties of an MDHAE. We can prove the correctness as follows:*

- *Correctness of the evaluation: From description of* $H.\mathrm{Eval}$ *and the correctness of* K, $\bar{T}$ *and* $\check{T}$, H *satisfies the correctness of evaluation.*
- *Projection preservation: From the correctness of* K, *if* f *is a projection, then* $\check{f}$ *is also a projection and* $\bar{f}$ *is a bitwisely described projection where* $\bar{f}$ *is the bitwisely described circuit of the deterministic algorithm* $K.\mathrm{Eval}(\mathring{ek}, f, \ldots)$. *Then, from the description of* $H.\mathrm{Eval}$ *and the correctness of* $\bar{T}$ *and* $\check{T}$, H *satisfies the projection preservation.*

Remark 5. *If* $\bar{T}$ *and* $\check{T}$ *supports efficient verification, then Construction 1 supports efficient decryption. We define* $H.\mathrm{Prep}$ *and* $H.\mathrm{EffDec}$ *as follows:*

- $H.\mathrm{Prep}(sk, P)$: *parse given two inputs* $sk = \mathring{ek}\|\mathring{sk}\|\bar{sk}\|\check{sk}\|k_{\bar{F}}\|k_{\bar{G}}\|k_{\check{F}}\|k_{\check{G}}$ *and* $P = (f, \tau_1, \ldots, \tau_l)$. *For* $i \in [l]$, *let* $\bar{\tau}_i \leftarrow \bar{G}(k_{\bar{G}}, \tau_i)$, $\check{\tau}_i \leftarrow \check{G}(k_{\check{G}}, \tau_i)$. *Now, let* $\bar{P} = (\bar{f}, \bar{\tau}_1, \ldots, \bar{\tau}_l)$, $\check{P} = (\check{f}, \check{\tau}_1, \ldots, \check{\tau}_l)$ *where* $\bar{f}$ *is the bitwisely described circuit of the deterministic algorithm* $K.\mathrm{Eval}(\mathring{ek}, f, \ldots)$. *Compute* $\bar{sk}_{\bar{P}} \leftarrow \bar{T}.\mathrm{Prep}(\bar{sk}, \bar{P})$ *and* $\check{sk}_{\check{P}} \leftarrow \bar{T}.\mathrm{Prep}(\check{sk}, \check{P})$ *and output* $sk_P := \mathring{sk}\|\bar{sk}_{\bar{P}}\|\check{sk}_{\check{P}}$.
- $H.\mathrm{EffDec}(sk_P, \Delta, c)$: *parse* $sk_P = \mathring{sk}\|\bar{sk}_{\bar{P}}\|\check{sk}_{\check{P}}$ *and* $c = (\mathring{c}, \bar{\sigma}, \check{\sigma})$. *Let* $\bar{\Delta} \leftarrow \bar{F}(k_{\bar{F}}, \Delta)$ *and* $\check{\Delta} \leftarrow \check{F}(k_{\check{F}}, \Delta)$. *If* $1 \leftarrow \bar{T}.\mathrm{EffVerify}(\bar{sk}_{\bar{P}}, \bar{\Delta}, \mathring{c}, \bar{\sigma})$ *and* $1 \leftarrow \check{T}.\mathrm{EffVerify}(\check{sk}_{\check{P}}, \check{\Delta}, 0, \check{\sigma})$, *then output* $m \leftarrow K.\mathrm{Dec}(\mathring{sk}, \mathring{c})$. *Otherwise, output* $\perp$.

Then $H.\mathrm{EffDec}(H.\mathrm{Prep}(sk, P), \Delta, c) = H.\mathrm{Dec}(sk, \Delta, P, c)$ *where* $(ek, sk) \leftarrow H.\mathrm{KeyGen}(1^\lambda)$. *Since the complexity of* $H.\mathrm{EffDec}$ *is independent of the time complexity of computing* f, *the above algorithms satisfy amortized efficiency.*

C. SECURITY

Theorem 1. *Suppose $\bar{F}$, $\bar{G}$, $\check{F}$ and $\check{G}$ are pseudorandom functions such that, $\bar{\mathcal{D}}$, $\bar{\mathcal{T}}$, $\check{\mathcal{D}}$ and $\check{\mathcal{T}}$ are superpolynomially large: $|\bar{\mathcal{D}}|, |\bar{\mathcal{T}}|, |\check{\mathcal{D}}|, |\check{\mathcal{T}}| \geq 2^{\omega(\log\lambda)}$. If $\bar{T}$ and $\check{\mathcal{T}}$ are selectively secure and K is secure, then H is a secure MDHAE.*

Proof. We use the same notations defined in Construction 1.

We define $\mathbf{Adv}_{\bar{F}}(\lambda)$, $\mathbf{Adv}_{\bar{G}}(\lambda)$, $\mathbf{Adv}_{\check{F}}(\lambda)$ and $\mathbf{Adv}_{\check{G}}(\lambda)$ to be distinguishing advantages of $\bar{F}(k_{\bar{F}}, \cdot)$, $\bar{G}(k_{\bar{G}}, \cdot)$, $\check{F}(k_{\check{F}}, \cdot)$ and $\check{G}(k_{\check{G}}, \cdot)$ from random functions $\bar{F}' : \mathcal{D} \to \bar{\mathcal{D}}$, $\bar{G}' : \mathcal{T} \to \bar{\mathcal{T}}$, $\check{F}' : \mathcal{D} \to \check{\mathcal{D}}$ and $\check{G}' : \mathcal{T} \to \check{\mathcal{T}}$, respectively, for $k_{\bar{F}}, k_{\bar{G}}, k_{\check{F}}, k_{\check{G}} \xleftarrow{\$} \{0,1\}^\lambda$.

We assume that there is an upper bound $\bar{l} = \mathsf{poly}(\lambda)$ on the number of inputs of the admissible function.

Let A be any PPT adversary against H in $\mathbf{Game}^{\mathrm{MDHAE}}$ with at most q queries for at most d different datasets. Then there are PPT adversaries $\mathring{B}$, $\mathring{B}'$, $\bar{B}$ and $\check{B}$ with at most q, q, $d\bar{l}q$ and $d\bar{l}q$ queries, respectively, such that

$$\begin{aligned}
&\mathbf{Adv}^{\mathrm{MDHAE}}_{H,\$_H,A}(\lambda) \\
&\leq \mathbf{Adv}_{\bar{F}}(\lambda) + \mathbf{Adv}_{\bar{G}}(\lambda) + \mathbf{Adv}_{\check{F}}(\lambda) + \mathbf{Adv}_{\check{G}}(\lambda) \\
&\quad + \frac{2q^2}{|\bar{\mathcal{D}}|} + \frac{2q^2}{|\check{\mathcal{D}}|} + \frac{2q^2\bar{l}^2}{|\bar{\mathcal{T}}|} + \frac{2q^2\bar{l}^2}{|\check{\mathcal{T}}|} + \mathbf{Adv}^{\mathrm{HE}}_{K,\$_K,\mathring{B}}(\lambda) \\
&\quad + \mathbf{Adv}^{\mathrm{HE}}_{K,\$_K,\mathring{B}'}(\lambda) + q\,\mathbf{Adv}^{\overline{\mathrm{MDHA}}}_{\bar{T},\bar{B}}(\lambda) \\
&\quad + (d+1)\left(\bar{l}q+1\right)\mathbf{Adv}^{\overline{\mathrm{MDHA}}}_{\check{T},\check{B}}(\lambda).
\end{aligned}$$

for some algorithms $\$_H$ and $\$_K$.

Suppose $\$_K$ be an algorithm such that $\mathbf{Adv}^{\mathrm{HE}}_{K,\$_K,B}(\lambda)$ is negligible for any PPT algorithm B. Then, in the rest of the proof, we specify $\$_H$ with respect to $\$_K$ as follows:

- $\$_H(sk, \Delta, \tau, \cdot)$: parse given secret key $sk = \mathring{ek}\|\mathring{sk}\|\bar{sk}\|\check{sk}\|k_{\bar{F}}\|k_{\bar{G}}\|k_{\check{F}}\|k_{\check{G}}$, let $\mathring{c} \leftarrow \$_K(\mathring{sk}, \cdot)$, $\bar{\Delta} \leftarrow \bar{F}(k_{\bar{F}}, \Delta)$, $\bar{\tau} \leftarrow \bar{G}(k_{\bar{G}}, \tau)$, $\check{\Delta} \leftarrow \check{F}(k_{\check{F}}, \Delta)$, $\check{\tau} \leftarrow \check{G}(k_{\check{G}}, \tau)$, $\bar{\sigma} \leftarrow \bar{T}.\mathrm{Auth}(\bar{sk}, \bar{\Delta}, \bar{\tau}, \mathring{c})$, $\check{\sigma} \leftarrow \check{T}.\mathrm{Auth}(\check{sk}, \check{\Delta}, \check{\tau}, 0)$. Output $c := (\mathring{c}, \bar{\sigma}, \check{\sigma})$.

In short, $\$_H$ is the same as $H.\mathrm{Enc}$ but instead of $K.\mathrm{Enc}$, $\$_H$ uses $\$_K$.

Before constructing adversaries $\mathring{B}$, $\mathring{B}$, $\bar{B}$ and $\check{B}$, we first switch PRFs with random functions using the games defined as follows:

Game $0(\lambda)$:
The original security game $\mathbf{Game}^{\mathrm{MDHAE}}_{H,\$_H,A}(\lambda)$.

Game $1(\lambda)$:
The security game $\mathbf{Game}^{\mathrm{MDHAE}}_{H',\$'_{H'},A}(\lambda)$ where H' and $\$'_{H'}$ are the same as H and $\$_H$, respectively,

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3100852, IEEE Access

except for the parts that use PRFs. In this game, $H'.\mathrm{KeyGen}(1^\lambda)$ samples random functions $\bar{F}' : \mathcal{D} \to \bar{\mathcal{D}}$, $\bar{G}' : \mathcal{T} \to \bar{\mathcal{T}}$, $\check{F}' : \mathcal{D} \to \check{\mathcal{D}}$ and $\check{G}' : \mathcal{T} \to \check{\mathcal{T}}$, and lets $sk := \mathring{ek} \| \mathring{sk} \| \bar{sk} \| \check{sk} \| \bar{F}' \| \bar{G}' \| \check{F}' \| \check{G}'$ as a secret key. Also, H', Enc, H', Dec and $\$'_{H'}$ use $\bar{F}'$, $\bar{G}'$, $\check{F}'$ and $\check{G}'$ instead of $\bar{F}(k_{\bar{F}}, \cdot)$, $\bar{G}(k_{\bar{G}}, \cdot)$, $\check{F}(k_{\check{F}}, \cdot)$ and $\check{G}(k_{\check{G}}, \cdot)$, respectively.

Using the security of the PRFs, we can bound

$$\begin{aligned}\mathbf{Adv}_A^{\textbf{Game } 0, \textbf{Game } 1}(\lambda) \le\ & \mathbf{Adv}_{\bar{F}}(\lambda) + \mathbf{Adv}_{\bar{G}}(\lambda) \\ & + \mathbf{Adv}_{\check{F}}(\lambda) + \mathbf{Adv}_{\check{G}}(\lambda).\end{aligned}$$

To bound $\mathbf{Adv}_A^{\textbf{Game } 1}(\lambda)$, we construct a PPT adversary $\mathring{B}$ against the challenger of the game $\mathbf{Game}^{\mathrm{HE}}_{K,\$_K,\mathring{B}}(\lambda)$ that runs A internally as follows (written in A's perspective):

Initialization

The challenger generates $(\mathring{ek}, \mathring{sk}) \leftarrow K.\mathrm{KeyGen}(1^\lambda)$ and send $\mathring{ek}$ to $\mathring{B}$. The challenger flips a coin $b \xleftarrow{\$} \{0,1\}$. If $b = 0$, then the challenger lets $E_K = K.\mathrm{Enc}(\mathring{sk}, \cdot)$. Otherwise, the challenger lets $E_K = \$_K(\cdot)$. Then, $\mathring{B}$ generates $(\bar{pk}, \bar{sk}) \leftarrow \bar{T}.\mathrm{KeyGen}(1^\lambda)$, $(\check{pk}, \check{sk}) \leftarrow \check{T}.\mathrm{KeyGen}(1^\lambda)$ and send $ek = \mathring{ek} \| \bar{ek} \| \check{ek}$ to A. $\mathring{B}$ initializes a set S as $\emptyset$.

Queries

$\mathring{B}$ responds to the queries of A as follows:

- For every encryption query (Δ, τ, m) that A makes, $\mathring{B}$ checks if $(\Delta, \tau, \cdot, \cdot) \in S$. If $(\Delta, \tau, \cdot, \cdot) \in S$, then $\mathring{B}$ rejects the query. Otherwise, $\mathring{B}$ queries m to the challenger, and gets $\mathring{c} = E_K(m)$ as the response. Then, $\mathring{B}$ lets $\bar{\Delta} = \bar{F}'(\Delta)$, $\bar{\tau} = \bar{G}'(\tau)$, $\check{\Delta} = \check{F}'(\Delta)$, $\check{\tau} = \check{G}'(\tau)$, $\bar{\sigma} \leftarrow \bar{T}.\mathrm{Auth}(\bar{sk}, \bar{\Delta}, \bar{\tau}, \mathring{c})$ and $\check{\sigma} \leftarrow \check{T}.\mathrm{Auth}(\check{sk}, \check{\Delta}, \check{\tau}, 0)$, and sends $c := (\mathring{c}, \bar{\sigma}, \check{\sigma})$ to A.
- For every decryption query (Δ, P, c) that A makes, $\mathring{B}$ checks if the query is redundant. If the query is redundant, $\mathring{B}$ rejects the query. If the query is non-redundant, $\mathring{B}$ checks if $H'.\mathrm{Dec}(sk, \Delta, P, c) \neq \perp$ using the knowledge of $\bar{F}'$, $\bar{G}'$, $\check{F}'$, $\check{G}'$ $\mathring{ek}$, $\bar{sk}$ and $\check{sk}$. If $H'.\mathrm{Dec}(sk, \Delta, P, c) \neq \perp$, then $\mathring{B}$ outputs "Bad" and halts. Otherwise, $\mathring{B}$ sends $\perp$ to A.

Finalization

A outputs a bit b'. Receiving b', $\mathring{B}$ also outputs b'.

Note that $\mathring{B}$'s responses to A are identical to the responses of the challenger of **Game** $1(\lambda)$ until A makes a decryption query (Δ, P, c) such that $\mathring{B}$ outputs "Bad" and halts (in other words, $H'.\mathrm{Dec}(sk, \Delta, P, c) \neq \perp$). We call such a query as a *bad* query. Also, $\mathring{B}$ wins the game $\mathbf{Game}^{\mathrm{HE}}_{K,\$_K,\mathring{B}}(\lambda)$ if and only if A does not make any bad queries on $\mathring{B}$ and outputs the winning bit $b' = b$ in the **Finalization** phase. Now, define two following events:

- $\mathcal{E}^{\textbf{Game } 1} = \{A \text{ makes a bad query on } \textbf{Game } 1(\lambda)\}$
- $\mathcal{E}^{\mathring{B}} = \{A \text{ makes a bad query on } \mathring{B}\}$

Then we see that

$$\begin{aligned}&\mathbf{Pr}\left[\textbf{Game } 1(\lambda) = 1\right] \\ \le\ &\mathbf{Pr}\left[\mathcal{E}^{\textbf{Game } 1}\right] \\ &+ \mathbf{Pr}\left[\left(\mathcal{E}^{\textbf{Game } 1}\right)^{\mathsf{C}}\right] \mathbf{Pr}\left[A \text{ outputs } b' = b \,\middle|\, \left(\mathcal{E}^{\textbf{Game } 1}\right)^{\mathsf{C}}\right] \\ =\ &\mathbf{Pr}\left[\mathcal{E}^{\mathring{B}}\right] \\ &+ \mathbf{Pr}\left[\left(\mathcal{E}^{\mathring{B}}\right)^{\mathsf{C}}\right] \mathbf{Pr}\left[\mathbf{Game}^{\mathrm{HE}}_{K,\$_K,\mathring{B}}(\lambda) = 1 \,\middle|\, \left(\mathcal{E}^{\mathring{B}}\right)^{\mathsf{C}}\right] \\ \le\ &\mathbf{Pr}\left[\mathcal{E}^{\mathring{B}}\right] + \mathbf{Adv}^{\mathrm{HE}}_{K,\$_K,\mathring{B}}(\lambda) \\ =\ &\mathbf{Pr}\left[\mathcal{E}^{\textbf{Game } 1}\right] + \mathbf{Adv}^{\mathrm{HE}}_{K,\$_K,\mathring{B}}(\lambda).\end{aligned}$$

Note that

$$\begin{aligned}&\mathbf{Pr}\left[\mathcal{E}^{\textbf{Game } 1}\right] \\ =\ &\mathbf{Pr}\left[\mathcal{E}^{\textbf{Game } 1} \mid b = 0\right] \mathbf{Pr}[b = 0] \\ &+ \mathbf{Pr}\left[\mathcal{E}^{\textbf{Game } 1} \mid b = 1\right] \mathbf{Pr}[b = 1] \\ \le\ &\frac{1}{2}\left|\mathbf{Pr}\left[\mathcal{E}^{\textbf{Game } 1} \mid b = 0\right] - \mathbf{Pr}\left[\mathcal{E}^{\textbf{Game } 1} \mid b = 1\right]\right| \\ &+ \mathbf{Pr}\left[\mathcal{E}^{\textbf{Game } 1} \mid b = 1\right].\end{aligned}$$

To bound the probability

$$\frac{1}{2}\left|\mathbf{Pr}\left[\mathcal{E}^{\textbf{Game } 1} \mid b = 0\right] - \mathbf{Pr}\left[\mathcal{E}^{\textbf{Game } 1} \mid b = 1\right]\right|,$$

we construct a PPT adversary $\mathring{B}'$ against the challenger of the game $\mathbf{Game}^{\mathrm{HE}}_{K,\$_K,\mathring{B}'}(\lambda)$. $\mathring{B}'$ is the same as $\mathring{B}$ except for the **Finalization** phase and the **Queries** phase. In the **Finalization** phase, $\mathring{B}'$ always outputs 0 instead of outputting A's output b' as $\mathring{B}$. In **Queries** phase, when A makes a query (Δ, P, c) such that $H'.\mathrm{Dec}(sk, \Delta, P, c) \neq \perp$, $\mathring{B}'$ outputs 1 and halts instead of outputting "Bad" as $\mathring{B}$. In A's perspective, $\mathring{B}$ and $\mathring{B}'$ are identical to each other. Therefore, we can write

$$\begin{aligned}&\mathbf{Adv}^{\mathrm{HE}}_{K,\$_K,\mathring{B}'}(\lambda) \\ =\ &\frac{1}{2}\left|\mathbf{Pr}\left[\mathring{B}' \text{ outputs } 1 \mid b = 0\right] - \mathbf{Pr}\left[\mathring{B}' \text{ outputs } 1 \mid b = 1\right]\right| \\ =\ &\frac{1}{2}\left|\mathbf{Pr}\left[\mathcal{E}^{\textbf{Game } 1} \mid b = 0\right] - \mathbf{Pr}\left[\mathcal{E}^{\textbf{Game } 1} \mid b = 1\right]\right|\end{aligned}$$

Now, we need to bound $\mathbf{Pr}[\mathcal{E}^{\textbf{Game } 1} \mid b = 1]$.

We first define **Game** $2(\lambda)$, which is the same as **Game** $1(\lambda)$ with slight changes. In **Game** $2(\lambda)$, the challenger always sets $b = 1$ instead of choosing b randomly as in **Game** $1(\lambda)$. Also, if A makes a bad query in the **Queries** phase, the challenger returns 1 and the game ends. If A does not output any bad queries, then the challenger returns 0 and the game ends. From the definition of **Game** 2, we see that $\mathbf{Pr}[\mathcal{E}^{\textbf{Game } 1} \mid b = 1] = \mathbf{Pr}[\textbf{Game } 2(\lambda) = 1]$. When A outputs a bad query (Δ^*, P^*, c^*), from the correctness of $\bar{T}$ and $\check{T}$, we see that $H'.\mathrm{Dec}(sk, \Delta^*, P^{*\prime}, c^*) \neq \perp$ where $P^{*\prime}$ is the fully bound sub-program of P^*. Also, if we let $P^{*\prime} = (f^*, \tau_1^*, \ldots, \tau_l^*)$, then the bad query (Δ^*, P^*, c^*) falls into one of the following two case:

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3100852, IEEE Access

Type 1 bad query:
$(\Delta^*, \tau_i^*, \cdot, \cdot) \notin S$ for at least one $i \in [l]$.
Type 2 bad query:
For $i \in [l]$, $(\Delta^*, \tau_i^*, m_i^*, c_i^*) \in S$ for some (unique) $m_i^* \in \mathcal{M}$, $c_i^* \in \mathcal{C}$, and $c^* \neq c^{**}$

where $c^{**} \leftarrow H'.\mathrm{Eval}(ek, f^*, c_1^*, \ldots, c_l^*)$. Also, if the bad query (Δ^*, P^*, c^*) is of Type 2, then it also falls into one of the following two cases:

Type $\bar{\mathbf{2}}$ bad query:
$(\mathring{c}^*, \bar{\sigma}^*) \neq (\mathring{c}^{**}, \bar{\sigma}^{**})$
Type $\check{\mathbf{2}}$ bad query:
$\check{\sigma}^* \neq \check{\sigma}^{**}$

where $c^* = (\mathring{c}^*, \bar{\sigma}^*, \check{\sigma}^*)$ and $c^{**} = (\mathring{c}^{**}, \bar{\sigma}^{**}, \check{\sigma}^{**})$. If we let

$$\check{\mathcal{E}} := \{\text{The first bad query of } A \text{ is of Type 1 in } \textbf{Game } 2(\lambda)\} \cup \{\text{The first bad query of } A \text{ is of Type } \check{2} \text{ in } \textbf{Game } 2(\lambda)\},$$
$$\bar{\mathcal{E}} := \{\text{The first bad query of } A \text{ is of Type } \bar{2} \text{ in } \textbf{Game } 2(\lambda)\},$$

then we see that

$$\mathbf{Pr}[\textbf{Game } 2(\lambda) = 1] \leq \mathbf{Pr}[\check{\mathcal{E}}] + \mathbf{Pr}[\bar{\mathcal{E}}].$$

To bound $\mathbf{Pr}[\bar{\mathcal{E}}]$, we construct a PPT adversary $\bar{B}$ of the game $\textbf{Game}^{\overline{\mathrm{MDHA}}}_{\bar{T},\bar{B}}(\lambda)$ as follows:

Initialization

$\bar{B}$ generates $(\mathring{ek}, \mathring{sk}) \leftarrow K.\mathrm{KeyGen}(1^\lambda)$, $(\check{ek}, \check{sk}) \leftarrow \check{T}.\mathrm{KeyGen}(1^\lambda)$, and prepares the selective queries as follows:

1) Sample $q^* \xleftarrow{\$} [q]$
2) Sample $\bar{\Delta}_i \xleftarrow{\$} \bar{\mathcal{D}}$, $\check{\Delta}_i \xleftarrow{\$} \check{\mathcal{D}}$ for $i \in [d]$ and $\bar{\tau}_j \xleftarrow{\$} \bar{\mathcal{T}}$, $\check{\tau}_j \xleftarrow{\$} \check{\mathcal{T}}$ for $j \in [\bar{l}q]$
3) Compute $\mathring{c}_{i,j} \leftarrow \$_K(\mathring{sk}, \cdot)$ for $(i,j) \in [d] \times [\bar{l}q]$
4) Submit $\left((\bar{\Delta}_i, \bar{\tau}_j, \mathring{c}_{i,j})\right)_{(i,j)\in[d]\times[\bar{l}q]}$ to the challenger

If $(\bar{\Delta}_i, \bar{\tau}_j) = (\bar{\Delta}_{i'}, \bar{\tau}_{j'})$ for some $(i,j) \neq (i',j')$, then the challenger rejects the query. Otherwise, the challenger generates $(\bar{ek}, \bar{sk}) \leftarrow \bar{T}.\mathrm{KeyGen}(1^\lambda)$, computes $\bar{\sigma}_{i,j} \leftarrow \bar{T}.\mathrm{Auth}(\bar{sk}, \bar{\Delta}_i, \bar{\tau}_j, \mathring{c}_{i,j})$ for $(i,j) \in [d] \times [\bar{l}q]$ and sends $(\bar{ek}, S_{\bar{T}})$ to B where $S_{\bar{T}} := \left\{(\bar{\Delta}_i, \bar{\tau}_j, \mathring{c}_{i,j}, \bar{\sigma}_{i,j})\right\}_{(i,j)\in[d]\times[\bar{l}q]}$. Then, $\bar{B}$ sends $ek := \mathring{ek} \| \bar{ek} \| \check{ek}$ to A and initializes a set S as $\emptyset$. $\bar{B}$ also programs $\bar{F}' : \mathcal{D} \to \bar{\mathcal{D}}$, $\check{F}' : \mathcal{D} \to \check{\mathcal{D}}$, $\bar{G}' : \mathcal{T} \to \bar{\mathcal{T}}$, $\check{G}' : \mathcal{T} \to \check{\mathcal{T}}$ to be functions that output $\bar{\Delta}_i$, $\check{\Delta}_i$, $\bar{\tau}_i$, $\check{\tau}_i$, respectively for the ith new input.

Queries

Among the queries of A, let $\Delta_i \in \mathcal{D}$ be the ith new dataset identifier and $\tau_j \in \mathcal{T}$ be the jth new data identifier. $\bar{B}$ handles queries of A as follows:

- For an encryption query $(\Delta_i, \tau_j, m_{i,j})$ that A makes, $\bar{B}$ checks if $(\Delta_i, \tau_j, \cdot, \cdot) \in S$. If $(\Delta_i, \tau_j, \cdot, \cdot) \in S$, then $\bar{B}$ rejects the query. Otherwise, $\bar{B}$ responds to the query $(\Delta_i, \tau_j, m_{i,j})$ with $c_{i,j} := (\mathring{c}_{i,j}, \bar{\sigma}_{i,j}, \check{\sigma}_{i,j})$ where $\check{\sigma}_{i,j} \leftarrow \check{T}.\mathrm{Auth}(\check{sk}, \check{\Delta}_i, \check{\tau}_j, 0)$ for $\check{\Delta}_i = \check{F}'(\Delta_i)$ and $\check{\tau}_j = \check{G}'(\tau_j)$ (also, $\bar{B}$ asks Δ_i and τ_j to $\bar{F}'$ and $\bar{G}'$, respectively, to set $\bar{F}'(\Delta_i) = \bar{\Delta}_i$ and $\bar{G}'(\tau_j) = \bar{\tau}_j$). After responding to the encryption query of A, $\bar{B}$ updates S with $S \leftarrow S \cup \{(\Delta_i, \tau_j, m_{i,j}, c_{i,j})\}$.
- For every decryption query (Δ^*, P^*, c^*) that A makes, $\bar{B}$ checks if the query is redundant. If the query (Δ^*, P^*, c^*) is redundant, then $\bar{B}$ rejects the query. If the query (Δ^*, P^*, c^*) is non-redundant decryption query before the q^*th query, then $\bar{B}$ gives $\perp$ as the response. If the q^*th query is an encryption query or a rejected query, then $\bar{B}$ outputs nothing and halts. If the q^*th query is the decryption query (Δ^*, P^*, c^*), then $\bar{B}$ parses $c^* = (\mathring{c}^*, \bar{\sigma}^*, \check{\sigma}^*)$ and finds $P^{*\prime} := (f^*, \tau_1^*, \ldots, \tau_l^*)$, the fully bound sub-program of P^*. If $(\Delta^*, \tau_i^*, m_i^*, c_i^*) \in S$ for some (unique) $m_i^* \in \mathcal{M}$ and $c_i^* \in \mathcal{C}$ for all $i \in [l]$ and $(\mathring{c}^*, \bar{\sigma}^*) \neq (\mathring{c}^{**}, \bar{\sigma}^{**})$ where $(\mathring{c}^{**}, \bar{\sigma}^{**}, \check{\sigma}^{**}) = H'.\mathrm{Eval}(ek, f^*, c_1^*, \ldots, c_l^*)$, then $\bar{B}$ outputs $(\bar{\Delta}^*, \bar{P^*}', \mathring{c}^*, \bar{\sigma}^*)$ as a forgery attempt and halts where $\bar{\Delta}^* = \bar{F}'(\Delta^*)$, $\bar{\tau}_i^* = \bar{G}'(\tau_i^*)$ for $i \in [l]$, $\bar{f}^*$ is the bitwisely described circuit of the deterministic algorithm $K.\mathrm{Eval}(ek, f^*, \ldots)$ and $\bar{P^*}' = (\bar{f}^*, \bar{\tau}_1^*, \ldots, \bar{\tau}_l^*)$. Otherwise, $\bar{B}$ outputs nothing and halts.

Finalization

$\bar{B}$ does not reach this phase.

Note that If A makes a Type $\bar{2}$ bad query on the q^*th query, then $\bar{B}$ wins the game $\textbf{Game}^{\overline{\mathrm{MDHA}}}_{\bar{T},\bar{B}}(\lambda)$.

For simplicity of the definitions below, we first let

$$prms = \Big(coins, (\bar{\Delta}_i)_{i\in[d]}, (\check{\Delta}_i)_{i\in[d]}, (\bar{\tau}_j)_{j\in[\bar{l}q]}, (\check{\tau}_j)_{j\in[\bar{l}q]}, r_K, r_{\bar{T}}, r_{\check{T}}, (e_{i,j})_{(i,j)\in[d]\times[\bar{l}q]}, (\bar{a}_{i,j})_{(i,j)\in[d]\times[\bar{l}q]}, (\check{a}_{i,j})_{(i,j)\in[d]\times[\bar{l}q]}\Big)$$

For ease of comparison, we define the following two games:

- **Game** $2(\lambda; prms)$: a deterministic **Game** $2(\lambda)$ that samples PRFs as $\bar{B}$ with deterministic A using randomness $coins$. More precisely, the challenger samples $(\bar{\Delta}_i)_{i\in[d]}$, $(\check{\Delta}_i)_{i\in[d]}$, $(\bar{\tau}_j)_{j\in[\bar{l}q]}$, $(\check{\tau}_j)_{j\in[\bar{l}q]}$ and programs $\bar{F}' : \mathcal{D} \to \bar{\mathcal{D}}$, $\check{F}' : \mathcal{D} \to \check{\mathcal{D}}$, $\bar{G}' : \mathcal{T} \to \bar{\mathcal{T}}$, $\check{G}' : \mathcal{T} \to \check{\mathcal{T}}$ to be functions that output $\bar{\Delta}_i$, $\check{\Delta}_i$, $\bar{\tau}_i$, $\check{\tau}_i$, respectively, for the ith new input. To run algorithms $K.\mathrm{KeyGen}$, $\bar{T}.\mathrm{KeyGen}$, $\check{T}.\mathrm{KeyGen}$, $\$_K(\mathring{sk}, \cdot)$, $\bar{T}.\mathrm{Auth}(\bar{sk}, \bar{\Delta}_i, \bar{\tau}_j, \cdot)$, $\check{T}.\mathrm{Auth}(\check{sk}, \check{\Delta}_i, \check{\tau}_j, 0)$, the challenger uses the randomness r_K, $r_{\bar{T}}$, $r_{\check{T}}$, $e_{i,j}$, $\bar{a}_{i,j}$, $\check{a}_{i,j}$ for $(i,j) \in [d] \times [\bar{l}q]$.
- **Game** $\bar{B}(\lambda; prms)$: a partially deterministic game $\textbf{Game}^{\overline{\mathrm{MDHA}}}_{\bar{T},\bar{B}}(\lambda)$ with deterministic A using randomness $coins$. More precisely, except for the choice of $q^* \in [q]$, the other parts of $\bar{B}$ and the challenger of this game

are deterministic. In A's perspective, the simulated challenger, consisting of $\bar{B}$ and the challenger of this game, samples $(\bar{\Delta}_i)_{i\in[d]}$, $(\check{\Delta}_i)_{i\in[d]}$, $(\bar{\tau}_j)_{j\in[\bar{l}q]}$, $(\check{\tau}_j)_{j\in[\bar{l}q]}$ and programs $\bar{F}' : \mathcal{D} \to \bar{\mathcal{D}}$, $\check{F}' : \mathcal{D} \to \check{\mathcal{D}}$, $\bar{G}' : \mathcal{T} \to \bar{\mathcal{T}}$, $\check{G}' : \mathcal{T} \to \check{\mathcal{T}}$ to be functions that output $\bar{\Delta}_i$, $\check{\Delta}_i$, $\bar{\tau}_i$, $\check{\tau}_i$, respectively for the ith new input. To run algorithms $K.\mathrm{KeyGen}$, $\bar{T}.\mathrm{KeyGen}$, $\check{T}.\mathrm{KeyGen}$, $\$_K(\mathring{sk}, \cdot)$, $\bar{T}.\mathrm{Auth}(\bar{sk}, \bar{\Delta}_i, \bar{\tau}_j, \cdot)$, $\check{T}.\mathrm{Auth}(\check{sk}, \check{\Delta}_i, \check{\tau}_j, 0)$, the simulated challenger uses the randomness r_K, $r_{\bar{T}}$, $r_{\check{T}}$, $e_{i,j}$, $\bar{a}_{i,j}$, $\check{a}_{i,j}$ for $(i,j) \in [d] \times [\bar{l}q]$.

Since **Game** $2(\lambda; prms)$ is **Game** $2(\lambda)$ with certain implementation of PRFs, $\mathbf{Pr}[\bar{\mathcal{E}}] = \mathbf{Pr}[\bar{\mathcal{E}}']$ where

$$\bar{\mathcal{E}}' := \{prms \mid \text{The first bad query that } A \text{ outputs is of Type } \bar{2} \text{ in } \textbf{Game } 2(\lambda; prms)\}.$$

Let

$$Coll := \{prms \mid \bar{\Delta}_i = \bar{\Delta}_{i'} \text{ or } \check{\Delta}_i = \check{\Delta}_{i'} \text{ for some } i \neq i' \text{ or } \bar{\tau}_j = \bar{\tau}_{j'} \text{ or } \check{\tau}_j = \check{\tau}_{j'} \text{ for some } j \neq j'\}.$$

Then for any fixed $prms \in Coll^{\complement} \cap \bar{\mathcal{E}}'$, suppose A made the first bad query on the q^{**}th query in **Game** $2(\lambda; prms)$. If $q^* = q^{**}$, **Game** $2(\lambda; prms)$ becomes the same as **Game** $\bar{B}(\lambda; prms)$ in A's perspective and $\bar{B}$ wins the game $\mathbf{Game}^{\overline{\mathrm{MDHA}}}_{\bar{T},\bar{B}}(\lambda)$. Therefore,

$$\frac{1}{q}\mathbf{Pr}\left[Coll^{\complement} \cap \bar{\mathcal{E}}'\right] \leq \mathbf{Pr}\left[\mathbf{Game}^{\overline{\mathrm{MDHA}}}_{\bar{T},\bar{B}}(\lambda) = 1\right] = \mathbf{Adv}^{\overline{\mathrm{MDHA}}}_{\bar{T},\bar{B}}(\lambda)$$

and

$$\begin{aligned}\mathbf{Pr}\left[\bar{\mathcal{E}}'\right] &= \mathbf{Pr}\left[Coll \cap \bar{\mathcal{E}}'\right] + \mathbf{Pr}\left[Coll^{\complement} \cap \bar{\mathcal{E}}'\right] \\ &\leq \mathbf{Pr}\left[Coll\right] + q\,\mathbf{Adv}^{\overline{\mathrm{MDHA}}}_{\bar{T},\bar{B}}(\lambda) \\ &\leq \frac{q^2}{|\bar{\mathcal{D}}|} + \frac{q^2}{|\check{\mathcal{D}}|} + \frac{q^2\bar{l}^2}{|\bar{\mathcal{T}}|} + \frac{q^2\bar{l}^2}{|\check{\mathcal{T}}|} + q\,\mathbf{Adv}^{\overline{\mathrm{MDHA}}}_{\bar{T},\bar{B}}(\lambda).\end{aligned}$$

To bound $\mathbf{Pr}[\check{\mathcal{E}}]$, we construct a PPT algorithm $\check{B}$ of the game $\mathbf{Game}^{\overline{\mathrm{MDHA}}}_{\bar{T},\check{B}}(\lambda)$ as follows:

Initialization

$\check{B}$ generates $(\mathring{sk}, \mathring{sk}) \leftarrow K.\mathrm{KeyGen}(\lambda)$, $(\bar{sk}, \bar{sk}) \leftarrow \bar{T}.\mathrm{KeyGen}(\lambda)$ and prepares the selective queries as follows:

1) Sample $q^* \xleftarrow{\$} [q]$, $i^* \xleftarrow{\$} [d+1]$, and $j^* \xleftarrow{\$} [\bar{l}q+1]$
2) Sample $\bar{\Delta}_i \xleftarrow{\$} \bar{\mathcal{D}}$, $\check{\Delta}_i \xleftarrow{\$} \check{\mathcal{D}}$ for $i \in [d]$ and $\bar{\tau}_j \xleftarrow{\$} \bar{\mathcal{T}}$, $\check{\tau}_j \xleftarrow{\$} \check{\mathcal{T}}$ for $j \in [\bar{l}q]$
3) For $i \in [d]$ and $j \in [\bar{l}q]$ such that $(i,j) \neq (i^*, j^*)$, let $b_{i,j} = 0$. If $(i,j) = (i^*, j^*)$, then let $b_{i^*,j^*} = b_{i,j} = 1$
4) Submit $\left((\check{\Delta}_i, \check{\tau}_j, b_{i,j})\right)_{(i,j)\in[d]\times[\bar{l}q]}$ to the challenger

If $(\check{\Delta}_i, \check{\tau}_j) = (\check{\Delta}_{i'}, \check{\tau}_{j'})$ for some $(i,j) \neq (i',j')$, then the challenger rejects the query. Otherwise, the challenger generates $(\check{ek}, \check{sk}) \leftarrow \check{T}.\mathrm{KeyGen}(1^\lambda)$, computes $\check{\sigma}_{i,j} \leftarrow \check{T}.\mathrm{Auth}(\check{sk}, \check{\Delta}_i, \check{\tau}_j, b_{i,j})$ for $(i,j) \in [d] \times [\bar{l}q]$ and sends $(\check{ek}, S_{\check{T}})$ to $\check{B}$ where $S_{\check{T}} := \left\{(\check{\Delta}_i, \check{\tau}_j, b_{i,j}, \check{\sigma}_{i,j})\right\}_{(i,j)\in[d]\times[\bar{l}q]}$. $\check{B}$ sends $ek := \mathring{ek} \| \bar{ek} \| \check{ek}$ to A and initializes a set S as $\emptyset$. $\check{B}$ also programs $\bar{F}' : \mathcal{D} \to \bar{\mathcal{D}}$, $\check{F}' : \mathcal{D} \to \check{\mathcal{D}}$, $\bar{G}' : \mathcal{T} \to \bar{\mathcal{T}}$, $\check{G}' : \mathcal{T} \to \check{\mathcal{T}}$ to be functions that output $\bar{\Delta}_i$, $\check{\Delta}_i$, $\bar{\tau}_i$, $\check{\tau}_i$, respectively for the ith new input.

Queries

Among the queries of A, let $\Delta_i \in \mathcal{D}$ be the ith new dataset identifier and $\tau_j \in \mathcal{T}$ be the jth new data identifier. $\check{B}$ handles queries of A as follows:

- For an encryption query $(\Delta_i, \tau_j, m_{i,j})$ that A makes, $\check{B}$ checks if $(\Delta_i, \tau_j, \cdot, \cdot) \in S$. If $(\Delta_i, \tau_j, \cdot, \cdot) \in S$, then $\check{B}$ rejects the encryption query. Otherwise, $\check{B}$ responds the query $(\Delta_i, \tau_j, m_{i,j})$ with $c_{i,j} := (\mathring{c}_{i,j}, \bar{\sigma}_{i,j}, \check{\sigma}_{i,j})$ where $\mathring{c}_{i,j} \leftarrow \$_K(\mathring{sk}, \cdot)$ and $\bar{\sigma}_{i,j} \leftarrow \bar{T}.\mathrm{Auth}(\bar{sk}, \bar{\Delta}_i, \bar{\tau}_j, \mathring{c}_{i,j})$ for $\bar{\Delta}_i = \bar{F}'(\Delta_i)$, $\bar{\tau}_j = \bar{G}'(\tau_j)$ ($\check{B}$ asks Δ_i and τ_j to $\check{F}'$ and $\check{G}'$, respectively, to set $\check{F}'(\Delta_i) = \check{\Delta}_i$ and $\check{G}'(\tau_j) = \check{\tau}_j$). After responding the encryption query of A, $\check{B}$ updates S with $S \leftarrow S \cup \{(\Delta_i, \tau_j, m_{i,j}, c_{i,j})\}$
- For every decryption query (Δ^*, P^*, c^*) that A makes, $\check{B}$ checks if the query is redundant. If the query (Δ^*, P^*, c^*) is redundant, then $\check{B}$ rejects the query. If the query (Δ^*, P^*, c^*) is non-redundant decryption query before the q^*th query, then $\check{B}$ returns $\perp$ as the response. If the q^*th query is an encryption query or a rejected query, then $\check{B}$ outputs nothing and halts. If the q^*th query is the decryption query (Δ^*, P^*, c^*), $\check{B}$ parses $c^* = (\mathring{c}^*, \bar{\sigma}^*, \check{\sigma}^*)$ and computes $P^{*\prime} := (f^*, \tau_1^*, \ldots, \tau_l^*)$, the fully bound sub-program of P^*. If $(\Delta^*, \tau_i^*, m_i^*, c_i^*) \in S$ for some (unique) $m_i^* \in \mathcal{M}$ and $c_i^* \in \mathcal{C}$ for all $i \in [l]$ and $\check{\sigma}^* \neq \check{\sigma}^{**}$ where $(\mathring{c}^{**}, \bar{\sigma}^{**}, \check{\sigma}^{**}) = H'.\mathrm{Eval}(ek, f^*, c_1^*, \ldots, c_l^*)$, then $\check{B}$ outputs $(\check{\Delta}^*, \check{P}^{*\prime}, 0, \check{\sigma}^*)$ as a forgery attempt and halts where $\check{P}^{*\prime} = (\check{f}^*, \check{\tau}_1^*, \ldots, \check{\tau}_l^*)$, $\check{\Delta}^* = \check{F}'(\Delta^*)$, $\check{\tau}_i^* = \check{G}'(\tau_i^*)$ for $i \in [l]$, $\check{f}^*$ is the boolean circuit obtained by replacing each unary gate of f^* with the identity gate, and each binary gate of f^* with the OR gate as defined in the beginning of the Section VI. If there is at least one $i \in [l]$ such that $(\Delta^*, \tau_i^*, \cdot, \cdot) \notin S$, then let l^* be the smallest number in the set $\{i \in [l] \mid (\Delta^*, \tau_i^*, \cdot, \cdot) \notin S\}$. If $\Delta^* = \Delta_{i^*}$ and $\tau_{l^*}^* = \tau_{j^*}$, then $\check{B}$ outputs $(\check{\Delta}^*, \check{P}^{*\prime}, 0, \check{\sigma}^*)$ as a forgery attempt and halts where $\check{\Delta}^*$ and $\check{P}^{*\prime}$ are defined as above. For other cases, $\check{B}$ outputs nothing and halts.

Finalization

$\check{B}$ does not reach this phase.

Note that on q^*th query, if A makes a Type 1 bad query while $\Delta^* = \Delta_{i^*}$ and $\tau^*_{l^*} = \tau_{j^*}$, or a Type $\check{2}$ bad query, then $\check{B}$ wins the game $\mathbf{Game}^{\overline{\mathrm{MDHA}}}_{\bar{T},\check{B}}(\lambda)$.

Let $prms$ be the tuple of randomness as defined above. For ease of comparison, we define the following game:

- **Game** $\check{B}(\lambda; prms)$: a partially deterministic game $\mathbf{Game}^{\overline{\mathrm{MDHA}}}_{\bar{T},\check{B}}(\lambda)$ with A using randomness $coins$. More precisely, except for the choices of $q^* \in [q]$, $i^* \in [d+1]$, $j^* \in [\bar{l}q+1]$, the other parts of $\check{B}$ and the challenger of this game are deterministic. In A's perspective, the simulated challenger, consisting of $\check{B}$ and the challenger of this game, samples $(\bar{\Delta}_i)_{i\in[d]}$, $(\check{\Delta}_i)_{i\in[d]}$, $(\bar{\tau}_j)_{j\in[\bar{l}q]}$, $(\check{\tau}_j)_{j\in[\bar{l}q]}$ and programs $\bar{F}' : \mathcal{D} \to \bar{\mathcal{D}}$, $\check{F}' : \mathcal{D} \to \check{\mathcal{D}}$, $\bar{G}' : \mathcal{T} \to \bar{\mathcal{T}}$, $\check{G}' : \mathcal{T} \to \check{\mathcal{T}}$ to be functions that output $\bar{\Delta}_i$, $\check{\Delta}_i$, $\bar{\tau}_i$, $\check{\tau}_i$, respectively for the ith new input. To run algorithms $K.\mathrm{KeyGen}$, $\bar{T}.\mathrm{KeyGen}$, $\check{T}.\mathrm{KeyGen}$, $\$_K(\mathring{sk}, \cdot)$, $\bar{T}.\mathrm{Auth}(\bar{sk}, \bar{\Delta}_i, \bar{\tau}_j, \cdot)$, $\check{T}.\mathrm{Auth}(\check{sk}, \check{\Delta}_i, \check{\tau}_j, \cdot)$, the simulated challenger uses the randomness r_K, $r_{\bar{T}}$, $r_{\check{T}}$, $e_{i,j}$, $\bar{a}_{i,j}$, $\check{a}_{i,j}$ for $(i,j) \in [d] \times [\bar{l}q]$.

If we define events $\check{\mathcal{E}}'$, $\check{\mathcal{E}}'_1$, $\check{\mathcal{E}}'_2$ as

$$\check{\mathcal{E}}' = \{prms \mid \text{The first bad query that } A \text{ outputs is of Type 1 or Type } \check{2} \text{ in } \mathbf{Game}\ 2(\lambda; prms)\},$$
$$\check{\mathcal{E}}'_1 = \{prms \mid \text{The first bad query that } A \text{ outputs is of Type 1 in } \mathbf{Game}\ 2(\lambda; prms)\},$$
$$\check{\mathcal{E}}'_2 = \{prms \mid \text{The first bad query that } A \text{ outputs is of Type } \check{2} \text{ in } \mathbf{Game}\ 2(\lambda; prms)\},$$

then we see that $\check{\mathcal{E}}' = \check{\mathcal{E}}'_1 \cup \check{\mathcal{E}}'_2$, $\check{\mathcal{E}}'_1 \cap \check{\mathcal{E}}'_2 = \emptyset$ and $\mathbf{Pr}[\check{\mathcal{E}}] = \mathbf{Pr}[\check{\mathcal{E}}']$.

For a fixed $prms \in Coll^{\complement} \cap \check{\mathcal{E}}'_1$, suppose A made, in $\mathbf{Game}\ 2(\lambda; prms)$, the first bad (Type 1) query (Δ^*, P^*, c^*) on the q^{**}th query where $P^{*\prime} = (f^*, \tau^*_1, \ldots, \tau^*_l)$ is the fully bound sub-program of P^*. If $q^* = q^{**}$, $\Delta^* = \Delta_{i^*}$ and $\tau^*_{l^*} = \tau_{j^*}$ where l^* is the smallest integer in the set $\{i \in [l] \mid (\Delta^*, \tau^*_i, \cdot, \cdot) \notin S\}$, then $\mathbf{Game}\ 2(\lambda; prms)$ becomes the same as $\mathbf{Game}\ \check{B}(\lambda; prms)$ in A's perspective and $\check{B}$ wins the game $\mathbf{Game}\ \check{B}(\lambda; prms)$. On the other hand, for fixed $prms \in Coll^{\complement} \cap \check{\mathcal{E}}'_2$, suppose A made the first bad (Type $\check{2}$) query on the q^{**}th query. If $q^* = q^{**}$, $i^* = d+1$ and $j^* = \bar{l}q+1$, then $\mathbf{Game}\ 2(\lambda; prms)$ becomes the same as $\mathbf{Game}\ \check{B}(\lambda; prms)$ in A's perspective and $\check{B}$ wins the game $\mathbf{Game}\ \check{B}(\lambda; prms)$. In other words, if $prms \in Coll^{\complement} \cap \check{\mathcal{E}}'$, then $\mathbf{Game}\ 2(\lambda; prms)$ and $\mathbf{Game}\ \check{B}(\lambda; prms)$ acts the same in A's perspective with probability greater than, or equal to $\frac{1}{(d+1)(\bar{l}q+1)}$. Therefore,

$$\begin{aligned}&\frac{1}{(d+1)(\bar{l}q+1)}\mathbf{Pr}\left[Coll^{\complement} \cap \check{\mathcal{E}}'\right]\\ &\le \mathbf{Pr}[\mathbf{Game}^{\overline{\mathrm{MDHA}}}_{\bar{T},\check{B}}(\lambda) = 1]\\ &= \mathbf{Adv}^{\overline{\mathrm{MDHA}}}_{\bar{T},\check{B}}(\lambda)\end{aligned}$$

and

$$\begin{aligned}\mathbf{Pr}\left[\check{\mathcal{E}}'\right] &= \mathbf{Pr}\left[Coll \cap \check{\mathcal{E}}'\right] + \mathbf{Pr}\left[Coll^{\complement} \cap \check{\mathcal{E}}'\right]\\ &\le \mathbf{Pr}\left[Coll\right] + (d+1)\left(\bar{l}q+1\right)\mathbf{Adv}^{\overline{\mathrm{MDHA}}}_{\bar{T},\check{B}}(\lambda)\\ &\le \frac{q^2}{|\bar{\mathcal{D}}|} + \frac{q^2}{|\check{\mathcal{D}}|} + \frac{q^2\bar{l}^2}{|\bar{\mathcal{T}}|} + \frac{q^2\bar{l}^2}{|\check{\mathcal{T}}|}\\ &\quad + (d+1)\left(\bar{l}q+1\right)\mathbf{Adv}^{\overline{\mathrm{MDHA}}}_{\bar{T},\check{B}}(\lambda).\end{aligned}$$

In conclusion, we can write

$$\begin{aligned}&\mathbf{Adv}^{\mathrm{MDHAE}}_{H,\$_H,A}(\lambda)\\ &\le \mathbf{Adv}_{\bar{F}}(\lambda) + \mathbf{Adv}_{\bar{G}}(\lambda) + \mathbf{Adv}_{\check{F}}(\lambda) + \mathbf{Adv}_{\check{G}}(\lambda)\\ &\quad + \frac{2q^2}{|\bar{\mathcal{D}}|} + \frac{2q^2}{|\check{\mathcal{D}}|} + \frac{2q^2\bar{l}^2}{|\bar{\mathcal{T}}|} + \frac{2q^2\bar{l}^2}{|\check{\mathcal{T}}|} + \mathbf{Adv}^{\mathrm{HE}}_{K,\$_K,\check{B}}(\lambda)\\ &\quad + \mathbf{Adv}^{\mathrm{HE}}_{K,\$_K,\hat{B}'}(\lambda) + q\,\mathbf{Adv}^{\overline{\mathrm{MDHA}}}_{\bar{T},\bar{B}}(\lambda)\\ &\quad + (d+1)\left(\bar{l}q+1\right)\mathbf{Adv}^{\overline{\mathrm{MDHA}}}_{\bar{T},\check{B}}(\lambda).\end{aligned}$$

□

VII. AN MDHA SCHEME AND A GENERIC CONSTRUCTION FOR BITWISELY EVALUABLE MDHA

In this section, we propose a selectively secure multi-dataset (leveled) fully homomorphic authenticator scheme and a generic construction for a selectively secure BE-MDHA. Our MDFHA scheme, Construction 2, can be directly used as $\check{T}$ in Construction 1. The generic construction we propose, Construction 3, can be used to construct a selectively secure BE-MDHA using a selectively secure MDHA such as Construction 2. Also, the result of Construction 3 can be used as $\bar{T}$ in Construction 1.

Like other (leveled) fully homomorphic authenticator schemes, our scheme is based on the first fully homomorphic signature scheme [4]. Our scheme is a slightly modified version of the first fully homomorphic signature [4], but our scheme supports multiple datasets without any additional transformation.

A. A SELECTIVELY SECURE MULTI-DATASET FULLY HOMOMORPHIC AUTHENTICATOR SCHEME

Before we introduce our secure MDFHA scheme, we go through some preliminaries.

1) Entropy

The *min-entropy* of a random variable X is defined as $\mathbf{H}_\infty(X) := -\log(\max_x \mathbf{Pr}[X = x])$. The *average conditional min-entropy* of X conditioned on Y is defined as

$$\mathbf{H}_\infty(X \mid Y) := -\log\left(\mathop{\mathbf{E}}_{y\leftarrow Y}\left[\max_x \mathbf{Pr}[X = x \mid Y = y]\right]\right).$$

Lemma 1. *Let X and Y be random variables defined on $\mathcal{X}$ and $\mathcal{Y}$, respectively. Then $\mathbf{H}_\infty(X \mid Y) \ge \mathbf{H}_\infty(X) - \log(|\mathcal{Y}|)$.*

2) The short integer solution problem

For a security parameter λ, suppose $n = \mathrm{poly}(\lambda)$, $k = \mathrm{poly}(\lambda)$, $q = 2^{\mathrm{poly}(\lambda)}$, $\beta = 2^{\mathrm{poly}(\lambda)}$ ($\beta \le q$) are given,

then the $\mathsf{SIS}_{n,k,q,\beta}$ hardness assumption means the following: For any PPT algorithm A that $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{n\times k}$ is given, the probability that A outputs $\mathbf{u} \in \mathbb{Z}_q^k$ such that $\|\mathbf{u}\|_\infty \le \beta$ and $\mathbf{Au} = \mathbf{0}$ is negligible. If A outputs $\mathbf{u}$ such that $\mathbf{Au} = \mathbf{0}$, then we say that A solved $\mathsf{SIS}_{n,k,q,\beta}$ problem.

Remark 6. *There are several versions of* SIS *hardness assumptions and the assumption above is believed to be true for some parameters [13]–[16].*

3) Lattice trapdoors

We can construct a matrix with a trapdoor using the following lemma.

Lemma 2. *([17]–[21]) For integers n and q, let $k_1 = n\lceil \log q \rceil$, $k_0 = O(n\log q) \ge n\log q + \omega(\log n)$, $k = k_0 + k_1$, $\beta_{sam} = O(n\sqrt{\log q})$, $\mathbf{G}_0 = \mathbf{I}_n \otimes \mathbf{g}^T \in \mathbb{Z}_q^{n\times nk_1}$ where $\mathbf{I}_n$ is the n-dimensional identity matrix and $\mathbf{g}^T = (1, 2, 2^2, \ldots, 2^{\lceil \log q\rceil - 1})$. Then for all $\bar{k} = \bar{k}(n) = \mathsf{poly}(n)$, there are efficient algorithms (halts within polynomial time with respect to their inputs)* Sam, TrapGen, SamPre, $\mathbf{H}_{alg}$ *satisfying the following:*

1) $\mathbf{U} \leftarrow \mathrm{Sam}(1^k, 1^{\bar{k}}, q)$ *samples a matrix* $\mathbf{U} \in \mathbb{Z}_q^{k\times\bar{k}}$ *such that* $\|\mathbf{U}\|_\infty \le \beta_{sam}$.
2) *For* $\mathbf{A}_0 \xleftarrow{\$} \mathbb{Z}_q^{n\times k_0}$ *and an invertible matrix* $\mathbf{H} \in \mathbb{Z}_q^{n\times n}$, $(\mathbf{A}, \mathbf{R}) \leftarrow \mathrm{TrapGen}(\mathbf{A}_0, \mathbf{H})$ *generates a random* $\mathbf{R} \leftarrow D_R$ *for certain distribution* D_R *over* $\mathbb{Z}_q^{k_0\times k_1}$ *and defines* $\mathbf{A}$ *as* $\mathbf{A} = [\mathbf{A}_0|\mathbf{HG}_0 - \mathbf{A}_0\mathbf{R}]$. *Moreover,* $(\mathbf{A}, \mathbf{R})$ *and the algorithm* SamPre *satisfies the following:*
   - *For* $\mathbf{A}' \xleftarrow{\$} \mathbb{Z}_q^{n\times k}$, $\mathbf{A} \overset{\text{stat}}{\approx} \mathbf{A}'$.
   - *For* $\mathbf{R} \leftarrow D_R$, $\max_{\|\mathbf{u}\|=1} \|\mathbf{Ru}\|_\infty \le O(n\log q)$ *except for negligible probability.*
   - *For* $\mathbf{U} \leftarrow \mathrm{Sam}(1^k, 1^{\bar{k}}, q)$, $\mathbf{V} = \mathbf{AU}$, $\mathbf{V}' \xleftarrow{\$} \mathbb{Z}_q^{n\times\bar{k}}$, $\mathbf{U}' \leftarrow \mathrm{SamPre}(\mathbf{A}_0, \mathbf{R}, \mathbf{H}, \mathbf{V}')$, $(\mathbf{A}, \mathbf{R}, \mathbf{U}, \mathbf{V}) \overset{\text{stat}}{\approx} (\mathbf{A}, \mathbf{R}, \mathbf{U}', \mathbf{V}')$. *Also,* $\mathbf{U}' \leftarrow \mathrm{SamPre}(\mathbf{A}_0, \mathbf{R}, \mathbf{H}, \mathbf{V}')$ *always satisfies* $\mathbf{AU}' = \mathbf{V}'$ *and* $\|\mathbf{U}'\|_\infty \le \beta_{sam}$.
   - *For any non-zero* $(\mathbf{u}_0, \mathbf{u}_1) \in \mathbb{Z}^{k_0} \times \mathbb{Z}^{k_1}$, *when* $\mathbf{A}_0$ *and* $\mathbf{A}_0\mathbf{R}$ *are given, the average min-entropy of* $\mathbf{Ru}_1$ *is at least* $\Omega(n)$.
3) *Let* $\mathbf{G} = [\mathbf{G}_0|\mathbf{0}] \in \mathbb{Z}_q^{n\times k}$, *then there exists a deterministic algorithm* $\mathbf{G}^{-1}$ *such that for any* $\mathbf{V} \in \mathbf{Z}_q^{n\times\bar{k}}$, $\mathbf{B} \leftarrow \mathbf{G}^{-1}(\mathbf{V})$ *such that* $\mathbf{B} \in \{0,1\}^{k\times\bar{k}}$ *and* $\mathbf{GB} = \mathbf{V}$.
4) *There is a deterministic algorithm* $\mathbf{H}'_{alg} : \mathbb{F}_{q^n} \to \mathbb{Z}_q^{n\times n}$ *such that for any distinct* $x, y \in \mathbb{F}_{q^n}$, $\mathbf{H}'_{alg}(x) - \mathbf{H}'_{alg}(y)$ *is an invertible matrix [21]. Therefore, there is a deterministic algorithm* $\mathbf{H}_{alg} : \mathbb{F}_{q^n} \setminus \{\mathbf{0}\} \to \mathbb{Z}_q^{n\times n}$ *such that for any* $\Delta \in \mathbb{F}_{q^n} \setminus \{\mathbf{0}\}$, $\mathbf{H}_{alg}(\Delta) := \mathbf{H}'_{alg}(\Delta) - \mathbf{H}'_{alg}(\mathbf{0}) \in \mathbb{Z}_q^{n\times n}$ *is an invertible matrix.*

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 $\beta_{max} = 2^{\omega(\log\lambda)d}$, $\beta_{SIS} = 2^{\omega(\log\lambda)}\beta_{max}$. Then, we choose $n = \mathsf{poly}(\lambda) = \omega(\log\lambda)$, a prime $q = 2^{\mathsf{poly}(\lambda)}$ so that $\mathsf{SIS}_{n,k,q,\beta_{SIS}}$ 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 given in Lemma 2, D_U be the distribution of the output of* $\mathrm{Sam}(1^k, 1^k, q)$, *and* $\beta_{init} = \beta_{sam} = \mathsf{poly}(\lambda)$. *Let $\mathcal{M} = \{0,1\}$, $\Sigma = \mathcal{D} \times \{\mathbf{U} \in \mathbb{Z}_q^{k\times k} \mid \|\mathbf{U}\|_\infty \le \beta_{max}\}$, $\mathcal{D} = \mathbb{F}_{q^n} \setminus \{\mathbf{0}\}$, $\mathcal{T}$ be any set and*

$$\begin{aligned}\mathcal{F} = \{f : \mathcal{M}^l \to \mathcal{M} \mid l = \mathsf{poly}(\lambda) \in \mathbb{Z},\ &\text{and } (\Delta, \mathbf{U}) \leftarrow \\ &T.\mathrm{Eval}(ek, f, (m_1, (\Delta, \mathbf{U}_1)), \ldots, (m_l, (\Delta, \mathbf{U}_l))) \\ &\text{satisfies } \|\mathbf{U}\|_\infty \le \beta_{max} \text{ for any choices of} \\ &\Delta \in \mathcal{D},\ (ek, sk) \leftarrow T.\mathrm{KeyGen}(1^\lambda) \text{ and } \mathbf{U}_i \leftarrow D_U, \\ &\text{for all } i \in [l], \text{ such that for some } \tau_i \in \mathcal{T} \text{ and } m_i \in \mathcal{M}, \\ &1 \leftarrow T.\mathrm{Verify}(sk, \Delta, (\mathrm{id}, \tau_i), m_i, (\Delta, \mathbf{U}_i))\}\end{aligned}$$

where id *is the identity function and* $T.$Eval *is defined below.*

Let $F : \{0,1\}^\lambda \times \mathcal{T} \to \mathbb{Z}_q^{n\times k}$ be a secure PRF.

We define a leveled fully homomorphic MDHA T as follows:

- $T.\mathrm{KeyGen}(1^\lambda)$*: sample* $\mathbf{A}_0 \xleftarrow{\$} \mathbb{Z}_q^{n\times k_0}$, $\mathbf{R} \leftarrow D_R$, $k_F \xleftarrow{\$} \{0,1\}^\lambda$ *and let* $\mathbf{A} := [\mathbf{A}_0|\mathbf{A}_1] = [\mathbf{A}_0| - \mathbf{A}_0\mathbf{R}]$. *Let* $ek := \mathbf{A}$ *and* $sk := (\mathbf{A}, \mathbf{R}, k_F)$, *and output* (ek, sk).
- $T.\mathrm{Auth}(sk, \Delta, \tau, m)$*: parse* $sk = (\mathbf{A}, \mathbf{R}, k_F)$ *and let* $\mathbf{A}_\Delta := [\mathbf{A}_0|\mathbf{H}_{alg}(\Delta)\mathbf{G} - \mathbf{A}_0\mathbf{R}] = [\mathbf{A}_0|\mathbf{H}_{alg}(\Delta)\mathbf{G} + \mathbf{A}_1]$ *and* $\mathbf{V} := F(k_F, \tau)$. *Compute* $\mathbf{U} \leftarrow \mathrm{SamPre}(\mathbf{A}_0, \mathbf{R}, \mathbf{H}_{alg}(\Delta), \mathbf{V})$, *and output* $\sigma = (\Delta, \mathbf{U})$.
- $T.\mathrm{Eval}(ek, f, (m_1, \sigma_1), \ldots, (m_l, \sigma_l))$*: parse* $ek = [\mathbf{A}_0|\mathbf{A}_1]$, $\sigma_i = (\Delta_i, \mathbf{U}_i)$ *for* $i \in [l]$. *Let* $\Delta = \Delta_1$, $\mathbf{A}_\Delta := [\mathbf{A}_0|\mathbf{H}_{alg}(\Delta)\mathbf{G} + \mathbf{A}_1]$, $\mathbf{V}_i := \mathbf{A}_\Delta\mathbf{U}_i + m_i\mathbf{G}$ *for* $i \in [l]$. *Evaluate* $\mathbf{U}$ *and* $\mathbf{V}$ *by following each gate of the circuit f where each gates are evaluated 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* $a \in \mathbb{Z}_q$,
  $$\mathbf{U} = a\mathbf{U}_1, \quad \mathbf{V} = a\mathbf{V}_1$$

  Finally, output $\sigma = (\Delta, \mathbf{U})$.
- $T.\mathrm{Verify}(sk, \Delta, P, m, \sigma)$*: parse* $sk = ([\mathbf{A}_0|\mathbf{A}_1], \mathbf{R}, k_F)$, $P = (f, \tau_1, \ldots, \tau_l)$ *and* $\sigma = (\Delta', \mathbf{U})$. *If* $\Delta \ne \Delta'$, *then output 0. Otherwise, let* $\mathbf{A}_\Delta := [\mathbf{A}_0|\mathbf{H}_{alg}(\Delta)\mathbf{G} + \mathbf{A}_1]$, $\mathbf{V} := \mathbf{A}_\Delta\mathbf{U} + m\mathbf{G}$, $\mathbf{V}_i := F(k_F, \tau_i)$ *for* $i \in [l]$. *From* $\mathbf{V}_1, \ldots, \mathbf{V}_l$, *evaluate* $\mathbf{V}'$ *by following each gates of f as* $T.$Eval. *If* $\mathbf{V} = \mathbf{V}'$, *then output 1. Otherwise, output 0.*

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3100852, IEEE Access

Remark 7. *Construction 2 satisfies the correctness property of an MDHA. We can prove the correctness as follows:*

- *Correctness of evaluation: Since* $T.\mathrm{Verify}$ *accepts the output of* $T.\mathrm{Eval}$ *when its inputs are also accepted by* $T.\mathrm{Verify}$*, we see that* T *satisfies the correctness of evaluation.*
- *Projection preservation: As*

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

for $(ek, sk) \leftarrow T.\mathrm{KeyGen}(1^\lambda)$ *and the* i*th projection function* π_i *over* $\mathcal{M}^l$*, we can say that* T *satisfies projection preservation.*

Remark 8. *Construction 2 supports efficient verification if we define* $T.\mathrm{Prep}$ *and* $T.\mathrm{EffVerify}$ *as follows:*

- $T.\mathrm{Prep}(sk, P)$*: parse* $sk = (\mathbf{A} = [\mathbf{A}_0|\mathbf{A}_1], \mathbf{R}, k_F)$*,* $P = (f, \tau_1, \ldots, \tau_l)$*. Let* $\mathbf{V}_i := F(k_F, \tau_i)$ *for* $i \in [l]$*. From* $\mathbf{V}_1, \ldots, \mathbf{V}_l$*, evaluate* $\mathbf{V}'$ *by following each gates of* f *as* $T.\mathrm{Eval}$*. Output* $sk_P = \mathbf{A}\|\mathbf{V}'$*.*
- $T.\mathrm{EffVerify}(sk_P, \Delta, m, \sigma)$*: parse* $sk_P = \mathbf{A}\|\mathbf{V}'$*,* $\mathbf{A} = [\mathbf{A}_0|\mathbf{A}_1]$ *and* $\sigma = (\Delta', \mathbf{U})$*. If* $\Delta \neq \Delta'$*, then output 0. Otherwise, let* $\mathbf{A}_\Delta := [\mathbf{A}_0|\mathbf{H}_{alg}(\Delta)\mathbf{G} + \mathbf{A}_1]$ *and* $\mathbf{V} := \mathbf{A}_\Delta\mathbf{U} + m\mathbf{G}$*. If* $\mathbf{V} = \mathbf{V}'$*, then output 1. Otherwise, output 0.*

Note that $T.\mathrm{EffVerify}\left(T.\mathrm{Prep}\left(sk, P\right), \Delta, m, \sigma\right) = T.\mathrm{Verify}\left(sk, P, \Delta, m, \sigma\right)$ *where* $(ek, sk) \leftarrow T.\mathrm{KeyGen}(1^\lambda)$*. Also, from the definition of* $T.\mathrm{EffVerify}$*, the complexity of* $T.\mathrm{EffVerify}$ *is independent of the time complexity of computing input* P*.*

Remark 9. *Construction 2 is (leveled) fully homomorphic from the following reasons. First, let* $f(x_1, x_2) = 1 - x_1 \cdot x_2$ *be a NAND gate. Consider two signatures* $\sigma_1 = (\Delta, \mathbf{U}_1)$*,* $\sigma_2 = (\Delta, \mathbf{U}_2)$ *such that* $1 \leftarrow T.\mathrm{Verify}(sk, \Delta, P_1, m_1, \sigma_1)$*,* $1 \leftarrow T.\mathrm{Verify}(sk, \Delta, P_2, m_2, \sigma_1)$*,* $\|\mathbf{U}_1\|_\infty \leq \beta$ *and* $\|\mathbf{U}_2\|_\infty \leq \beta$ *for some admissible programs* P_1, P_2*, messages* $m_1, m_2 \in \{0,1\}$ *and* $(ek, sk) \leftarrow T.\mathrm{KeyGen}(1^\lambda)$*. By following the definition of* $T.\mathrm{Eval}$*, we see that* $\|\mathbf{U}\|_\infty \leq (k+1)\beta$ *when* $\mathbf{U} \leftarrow T.\mathrm{Eval}(ek, f, (m_1, \sigma_1), (m_2, \sigma_2))$*. Therefore, for any freshly generated message-signature tuples* $(m'_1, \sigma'_1), \ldots, (m'_l, \sigma'_l)$ *and any depth* d *circuit* g *with arity* l *that consists of NAND gates,* $\|\mathbf{U}^*\|_\infty \leq (k+1)^d\beta_{init} \leq 2^{\omega(\log\lambda)d} \leq \beta_{max}$ *when* $\mathbf{U}^* \leftarrow T.\mathrm{Eval}(ek, g, (m'_1, \sigma'_1), \ldots, (m'_l, \sigma'_l))$*. In other words, any depth* d *circuit that consists of NAND gates is an admissible function of* T*.*

Theorem 2. *T on Construction 2 is selectively secure under the* $\mathsf{SIS}_{n,k,q,\beta_{SIS}}$ *hardness assumption.*

Proof. We define $\mathbf{Adv}_F(\lambda)$ to be the distinguishing advantage of $F(k_F, \cdot)$ from random function $F' : \mathcal{T} \to \mathbb{Z}_q^{n\times k}$ for $k_F \xleftarrow{\$} \{0,1\}^\lambda$.

Let A be any PPT adversary of T in $\mathbf{Game}_{T,A}^{\overline{\mathrm{MDHA}}}$ that makes at most q queries. Then there is a PPT algorithm B, running A internally, which solves $\mathsf{SIS}_{n,k,q,\beta_{SIS}}$ problem with probability $\mathbf{Adv}_B^{\mathsf{SIS}}(\lambda)$ such that

$$\mathbf{Adv}_{T,A}^{\overline{\mathrm{MDHA}}}(\lambda) \leq \mathbf{Adv}_B^{\mathsf{SIS}}(\lambda) + \mathbf{Adv}_F(\lambda) + \mathsf{negl}(\lambda)$$

Before constructing B, we define some games as follows:

Game $0(\lambda)$:

The original security game $\mathbf{Game}_{T,A}^{\overline{\mathrm{MDHA}}}(\lambda)$

Game $1(\lambda)$:

The security game $\mathbf{Game}_{T',A}^{\overline{\mathrm{MDHA}}}(\lambda)$ where T' is the same as T except for the parts that use PRF F. In this game, $T'.\mathrm{KeyGen}(1^\lambda)$ samples a random function $F' : \mathcal{T} \to \mathbb{Z}_q^{n\times k}$, and lets $sk := (\mathbf{A}, \mathbf{R}, F')$ as a secret key. Also, $T'.\mathrm{Auth}$ and $T'.\mathrm{Verify}$ uses F' instead of $F(k_F, \cdot)$.

Using the security of the PRF F, we can bound

$$\mathbf{Adv}_A^{\mathbf{Game}\ 0,\mathbf{Game}\ 1}(\lambda) \leq \mathbf{Adv}_F(\lambda).$$

To bound $\mathbf{Adv}_A^{\mathbf{Game}\ 1}(\lambda)$, we construct a PPT algorithm B that solves the SIS problem for $\mathbf{A}_0 \xleftarrow{\$} \mathbb{Z}_q^{n\times k_0}$ by running A internally as follows (written in A's perspective):

Selective Queries

A makes selective queries $((\Delta_i, \tau_i, m_i))_{i\in[q]}$.

Initializatoin and Response

B samples $i^* \xleftarrow{\$} [q]$, $\mathbf{R} \leftarrow D_R$ and let $\mathbf{A} = [\mathbf{A}_0|\mathbf{A}_1] = [\mathbf{A}_0| - \mathbf{H}_{alg}(\Delta_{i^*})\mathbf{G} - \mathbf{A}_0\mathbf{R}]$ and $\mathbf{A}_{\Delta_{i^*}} := [\mathbf{A}_0| - \mathbf{A}_0\mathbf{R}]$. B samples $\mathbf{U}_i \leftarrow D_U$ and program a random function $F' : \mathcal{T} \to \mathbb{Z}_q^{n\times k}$ to satisfy $F'(\tau_i) = \mathbf{V}_i := \mathbf{A}_{\Delta_{i^*}}\mathbf{U}_i + m_i\mathbf{G}$ for $i \in Ind^* := \{i \in [q] \mid \Delta_i = \Delta_{i^*}\}$. For $i \in [q] \setminus Ind^*$, B samples $\mathbf{U}_i \leftarrow \mathrm{SamPre}(\mathbf{A}_0, \mathbf{R}, \mathbf{H}_{alg}(\Delta_i) - \mathbf{H}_{alg}(\Delta_{i^*}), \mathbf{V}_i)$ where $\mathbf{A}_{\Delta_i} := [\mathbf{A}_0|\mathbf{H}_{alg}(\Delta_i)\mathbf{G} + \mathbf{A}_1]$ and $\mathbf{V}_i := F'(\tau_i)$. Then, B lets $ek = \mathbf{A}$, $\sigma_i := (\Delta_i, \mathbf{U}_i)$ for all $i \in [q]$, and sends (ek, S) to A where $S = \{(\Delta_i, \tau_i, m_i, \sigma_i)\}_{i\in[q]}$.

Finalization

A outputs a forgery attempt $(\Delta^*, P^*, m^*, \sigma^*)$.

In A's perspective, B's simulation above is indistinguishable to the original challenger of $\mathbf{Game}_{T,A}^{\overline{\mathrm{MDHA}}}(\lambda)$ except for negligible probability from Lemma 2. Also, regardless of the choice of i^*, B acts almost the same in A's perspective from Lemma 2. Therefore, when A outputs a forgery attempt $(\Delta^*, P^*, m^*, \sigma^* = (\Delta^*, \mathbf{U}^*))$, the probability that $\Delta^* = \Delta_{i^*}$ is at least $\frac{1}{q}$ except for negligible probability.

If $(\Delta^*, P^*, m^*, \sigma^*)$ is a forgery, then for the fully bound sub-program $P^{*\prime} = (f^*, \tau_1, \ldots, \tau_l^*)$ of P^*, there is $(\Delta^*, \tau_i^*, m_i^*, \sigma_i^*) \in S$ for some (unique) $m_i^* \in \mathcal{M}$, $\sigma_i^* \in \Sigma$ for $i \in [l]$. Also, $\mathbf{A}_{\Delta^*}\mathbf{U}^* + m^*\mathbf{G} = \mathbf{V}^*$ where $\mathbf{V}^*$ is evaluated as $T'.\mathrm{Eval}$ using $(\mathbf{V}_1^*, \ldots, \mathbf{V}_l^*) = (F'(\tau_1^*), \ldots, F'(\tau_l^*))$. If we let $\sigma^{**} = (\Delta^{**}, \mathbf{U}^{**}) \leftarrow T'.\mathrm{Eval}\left(ek, f^*, (m_1^*, \sigma_1^*), \ldots, (m_l^*, \sigma_l^*)\right)$ and $m^{**} = f^*(m_1^*, \ldots, m_l^*)$, then $(m^*, \sigma^*) \neq (m^{**}, \sigma^{**})$ also holds $(\Delta^* = \Delta^{**})$. In other words, If we let $m^\times := m^{**} - m^*$ and $\mathbf{U}^\times := [\mathbf{I}_{k_0}| - \mathbf{R}](\mathbf{U}^* - \mathbf{U}^{**})$, then

$$\mathbf{A}_{\Delta_{i^*}}(\mathbf{U}^* - \mathbf{U}^{**}) = \mathbf{A}_0\mathbf{U}^\times = m^\times\mathbf{G}.$$

Moreover, we have $\|\mathbf{U}^* - \mathbf{U}^{**}\|_\infty \leq 2\beta_{max}$ from $(\Delta^*, \mathbf{U}^*), (\Delta^{**}, \mathbf{U}^{**}) \in \Sigma$ and $\max_{\|\mathbf{u}\|=1} \|\mathbf{Ru}\|_\infty \leq O(n \log q)$ from Lemma 2. Thus, we can write $\|\mathbf{U}^\times\|_\infty \leq 2\beta_{max}(O(n \log q) + 1) \leq \beta_{SIS}$.

To solve the $\mathsf{SIS}_{n,k,q,\beta_{SIS}}$ problem, we consider the following cases:

- $m^\times = 0$: It is enough to show that $\mathbf{U}^\times \neq \mathbf{0}$ except for negligible probability. Let $\mathbf{U}^* - \mathbf{U}^{**} = \begin{bmatrix} \mathbf{U}_0^* \\ \mathbf{U}_1^* \end{bmatrix}$ such that $\mathbf{U}^\times = \mathbf{U}_0^* - \mathbf{R}\mathbf{U}_1^*$. If $\mathbf{U}_1^* = \mathbf{0}$, then $\mathbf{U}_0^* \neq \mathbf{0}$ from $\mathbf{U}^* - \mathbf{U}^{**} \neq \mathbf{0}$. Therefore, $\mathbf{U}^\times \neq \mathbf{0}$. If $\mathbf{U}_1^* \neq \mathbf{0}$, then from Lemma 2, we know that the min-entropy of $\mathbf{R}\mathbf{U}_1^*$ is at least $\Omega(n)$ when $\mathbf{A}_0$ and $\mathbf{AR}$ are given. Therefore, $\mathbf{U}^\times \neq \mathbf{0}$ except for negligible probability.
- $m^\times \neq 0$: B first samples $\mathbf{t} \xleftarrow{\$} \{0,1\}^{k_0}$ and computes $\mathbf{t}' = \mathbf{G}^{-1}(\mathbf{A}_0\mathbf{t}/m^\times) \in \{0,1\}^k$. Then we see that

$$\begin{aligned}\mathbf{A}_0((\mathbf{U}^* - \mathbf{U}^{**})\mathbf{t}' - \mathbf{t}) =& \mathbf{A}_0(\mathbf{U}^* - \mathbf{U}^{**})\mathbf{t}' - \mathbf{A}_0\mathbf{t} \\ =& m^\times \mathbf{G}\mathbf{t}' - \mathbf{A}_0\mathbf{t} \\ =& \mathbf{0}\end{aligned}$$

  If we let $\mathbf{u} = (\mathbf{U}^* - \mathbf{U}^{**})\mathbf{t}' - \mathbf{t}$, then we have $\|\mathbf{u}\|_\infty \leq 2k\beta_{max} + 1 \leq \beta_{SIS}$ and $\mathbf{A}_0\mathbf{u} = \mathbf{0}$. All we need to prove is that $\mathbf{u} \neq \mathbf{0}$ except for negligible probability. From the fact that $\mathbf{t}'$ is deterministic when $\mathbf{A}_0\mathbf{t}$ is given, we have

$$\begin{aligned}\mathbf{H}_\infty(\mathbf{t}|\mathbf{t}') \geq& \mathbf{H}_\infty(\mathbf{t}|\mathbf{A}_0\mathbf{t}) \\ \geq& \mathbf{H}_\infty(\mathbf{t}) - n \log q \\ =& k_0 - n \log q \\ =& O(n)\end{aligned}$$

  from Lemma 1. In other words, $\mathbf{u} = (\mathbf{U}^* - \mathbf{U}^{**})\mathbf{t}' - \mathbf{t} \neq \mathbf{0}$ except for negligible probability.

In conclusion, if A makes a forgery, then B can solve the $\mathsf{SIS}_{n,k,q,\beta_{SIS}}$ problem except for negligible probability. Therefore, we may write

$$\begin{aligned}&\mathbf{Adv}_{T,A}^{\overline{\mathrm{MDHA}}}(\lambda) \\ =& \mathbf{Adv}_A^{\mathbf{Game}\ 0}(\lambda) \\ \leq& \mathbf{Adv}_A^{\mathbf{Game}\ 0, \mathbf{Game}\ 1}(\lambda) + \mathbf{Adv}_A^{\mathbf{Game}\ 1}(\lambda) \\ \leq& \mathbf{Adv}_F(\lambda) + \mathbf{Adv}_B^{\mathsf{SIS}}(\lambda) + \mathsf{negl}(\lambda).\end{aligned}$$

□

B. GENERIC CONSTRUCTION OF BITWISELY EVALUABLE MULTI-DATASET HOMOMORPHIC AUTHENTICATOR

Here, we give a generic construction of a selectively secure BE-MDHA.

Construction 3. *Let T be an MDHA. Let $\mathcal{M} := \{0,1\}$, Σ, $\mathcal{D}$, $\mathcal{T}$, $\mathcal{F}$ be the message space, the tag space, the dataset identifier space, the data identifier space, and the admissible function space of T. Similarly, let $\bar{\mathcal{M}}$, $\bar{\Sigma}$, $\bar{\mathcal{D}}$, $\bar{\mathcal{T}}$, $\bar{\mathcal{F}}$ be the corresponding ones of $\bar{T}$ where every element in $\bar{\mathcal{M}}$ is given as an n-bit binary encoding and $\bar{\mathcal{F}}$ is defined as*

$$\bar{\mathcal{F}} := \left\{ \bar{f} = (f_1, \ldots, f_n) \in \mathcal{F}^n \mid \bar{f} \text{ is a bitwisely described circuit on } \mathcal{M}^l \text{ for some } l = \mathsf{poly}(\lambda) \right\}.$$

Without loss of generality, we let $\bar{\mathcal{T}} = \mathcal{T} \times [n]$. Using T, we construct a BE-MDHA $\bar{T}$ as follows.

- $\bar{T}.\mathrm{KeyGen}(1^\lambda)$*: let* $(ek, sk) \leftarrow T.\mathrm{KeyGen}(1^\lambda)$ *and outputs* $(\bar{ek}, \bar{sk}) := (ek, sk)$.
- $\bar{T}.\mathrm{Auth}\left(\bar{sk}, \Delta, \tau, \bar{m}\right)$*: parse* $\bar{m} = \langle m_1, \ldots, m_n \rangle$ *and let* $sk = \bar{sk}$, $\sigma_i \leftarrow T.\mathrm{Auth}(sk, \Delta, (\tau, i), m_i)$ *for* $i \in [n]$. *Output* $\bar{\sigma} := \sigma_1 \| \ldots \| \sigma_n$.
- $\bar{T}.\mathrm{Eval}\left(\bar{ek}, \bar{f}, (\bar{m}, \bar{\sigma}_1), \ldots, (\bar{m}_l, \bar{\sigma}_l)\right)$*: parse* $\bar{f} = (f_1, \ldots, f_n)$, $\bar{m}_i = \langle m_{i,1}, \ldots, m_{i,n} \rangle$, $\bar{\sigma}_i = \sigma_{i,1} \| \ldots, \| \sigma_{i,n}$ *for* $i \in [n]$. *For all* $i \in [n]$, *compute* $\sigma_i \leftarrow T.\mathrm{Eval}\left(ek, f_i, (m_{i',j'}, \sigma_{i',j'})_{(i',j') \in [l] \times [n]}\right)$. *Output* $\bar{\sigma} = \sigma_1 \| \ldots \| \sigma_n$.
- $\bar{T}.\mathrm{Verify}\left(\bar{sk}, \Delta, \bar{P}, \bar{m}, \bar{\sigma}\right)$*: parse* $\bar{P} = \left(\bar{f}, \tau_1, \ldots, \tau_l\right)$, $\bar{f} = (f_1, \ldots, f_n)$, $\bar{m} = \langle m_1, \ldots, m_n \rangle$ *and* $\bar{\sigma} = (\sigma_1, \ldots, \sigma_n)$. *Let* $P_i := \left(f_i, ((\tau_s, t))_{(s,t) \in [l] \times [n]}\right)$ *for* $i \in [n]$. *If* $1 \leftarrow T.\mathrm{Verify}\left(sk, \Delta, P_i, m_i, \sigma_i\right)$ *for* $i \in [n]$, *then output 1. Otherwise, output 0.*

Remark 10. *Construction 3 satisfies the correctness property of a BE-MDHA. We can prove the correctness as follows:*

- *Correctness of the evaluation: From descriptions of* $\bar{T}.\mathrm{Eval}, \bar{T}.\mathrm{Verify}$ *and the correctness of* T, $\bar{T}$ *satisfies the correctness of evaluation.*
- *Projection preservation: Let* $(ek, sk) \leftarrow T.\mathrm{KeyGen}(1^\lambda)$, $\bar{ek} = ek$, $\bar{sk} = sk$ *and* $\bar{\pi}_i = (\pi_{i,1}, \ldots, \pi_{i,n})$ *be a bitwisely described projection on ith coordinate on* $\mathcal{M}^l$ *for some* $l = \mathsf{poly}(\lambda)$. *If* $(\bar{m}_{j'}, \bar{\sigma}_{j'}) = \left(\langle m_{j',1}, \ldots, m_{j',n} \rangle, \sigma_{j',1} \| \ldots \| \sigma_{j',n}\right)$ *satisfies* $1 \leftarrow \bar{T}.\mathrm{Verify}\left(\bar{sk}, \Delta, \bar{P}_{j'}, \bar{m}_{j'}, \bar{\sigma}_{j'}\right)$ *for some admissible bitwisely described program* $\bar{P}_{j'}$ *for all* $j' \in [n]$, *then since* $\pi_{i,1}, \ldots, \pi_{i,n}$ *are projection circuits such that* $\pi_{i,j}\left((b_{i',j'})_{(i',j') \in [l] \times [n]}\right) = b_{i,j}$ *for* $j \in [n]$, *we see that* $\bar{T}.\mathrm{Eval}\left(ek, \Delta, \bar{\pi}_i, (\bar{m}_1, \bar{\sigma}_1), \ldots, (\bar{m}_l, \bar{\sigma}_l)\right) = \bar{\sigma}_i$ *from the projection preservation of* T.

Remark 11. *If T supports efficient verification, then Construction 3 also supports efficient verification with following* $\bar{T}.\mathrm{Prep}$ *and* $\bar{T}.\mathrm{EffVerify}$.

- $\bar{T}.\mathrm{Prep}\left(\bar{sk}, \bar{P}\right)$*: parse* $\bar{P} = \left(\bar{f}, \tau_1, \ldots, \tau_l\right)$ *and* $\bar{f} = (f_1, \ldots, f_n)$. *Let* $P_i := \left(f_i, ((\tau_s, t))_{(s,t) \in [l] \times [n]}\right)$ *for* $i \in [n]$. *Compute* $sk_{P_i} \leftarrow T.\mathrm{Prep}(sk, P_i)$ *for* $i \in [n]$. *Output* $\bar{sk}_{\bar{P}} := sk_{P_1} \| \ldots \| sk_{P_n}$.
- $\bar{T}.\mathrm{EffVerify}\left(\bar{sk}_{\bar{P}}, \Delta, \bar{m}, \bar{\sigma}\right)$*: parse given inputs* $\bar{sk}_{\bar{P}} := sk_{P_1} \| \ldots \| sk_{P_n}$, $\bar{m} = \langle m_1, \ldots, m_n \rangle$ *and* $\bar{\sigma} = \sigma_1 \| \ldots \| \sigma_n$. *If* $1 \leftarrow T.\mathrm{EffVerify}\left(sk_{P_i}, \Delta, m_i, \sigma_i\right)$ *for all* $i \in [n]$, *then output 1. Otherwise, output 0.*

Note that $\bar{T}.\mathrm{EffVerify}\left(T.\mathrm{Prep}\left(\bar{sk}, \bar{P}\right), \Delta, \bar{m}, \bar{\sigma}\right) = \bar{T}.\mathrm{Verify}\left(\bar{sk}, \bar{P}, \Delta, \bar{m}, \bar{\sigma}\right)$ *where* $\left(\bar{ek}, \bar{sk}\right) \leftarrow \bar{T}.\mathrm{KeyGen}(1^\lambda)$. *Also, from the amortized efficiency of* T, *the complexity of* $\bar{T}.\mathrm{EffVerify}$ *is independent of the time complexity of computing input* $\bar{P}$. *In other words,* $\bar{T}$ *supports efficient verification if* T *supports efficient verification.*

Theorem 3. *If T is a selectively secure MDHA, then $\bar{T}$ is a selectively secure BE-MDHA.*

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3100852, IEEE Access

Proof. Let A be any PPT adversary of $\bar{T}$ in $\mathbf{Game}^{\overline{\mathrm{MDHA}}}_{\bar{T},A}$ that makes at most q queries. Then there is a PPT algorithm B that makes nq queries such that

$$\mathbf{Adv}^{\overline{\mathrm{MDHA}}}_{\bar{T},A}(\lambda) \leq \mathbf{Adv}^{\overline{\mathrm{MDHA}}}_{T,B}(\lambda)$$

We construct such algorithm B that runs A internally as follows (written in A's perspective):

Selective Queries

A makes selective queries $((\Delta_i, \tau_i, \bar{m}_i))_{i\in[q]}$. If $(\Delta_i, \tau_i) = (\Delta_{i'}, \tau_{i'})$ for distinct $i, i' \in [q]$, then B rejects the queries. Otherwise, B parses $\bar{m}_i = \langle m_{i,1}, \ldots, m_{i,n}\rangle$ for $i \in [q]$ and makes selective queries $((\Delta_i, (\tau_i, j), m_{i,j}))_{(i,j)\in[q]\times[n]}$ to the challenger.

Initialization and Response

The challenger generates a key pair $(ek, sk) \leftarrow T.\mathrm{KeyGen}(1^\lambda)$, computes authentication tags $\sigma_{i,j} \leftarrow T.\mathrm{Auth}(sk, \Delta_i, (\tau_i, j), m_{i,j})$ for all $(i,j) \in [q]\times[n]$ and sends (ek, S_T) to B where $S_T := \{(\Delta_i, (\tau_i, j), m_{i,j}, \sigma_{i,j})\}_{(i,j)\in[q]\times[n]}$. Then, B lets $\bar{ek} := ek$, $\bar{\sigma}_i := (\sigma_{i,1}, \ldots, \sigma_{i,n})$ for $i \in [q]$ and sends $(\bar{ek}, S)$ to A where $S := \{(\Delta_i, \tau_i, \bar{m}_i, \bar{\sigma}_i)\}_{i\in[q]}$.

Finalization

A outputs a forgery attempt $\left(\Delta^*, \bar{P}^*, \bar{m}^*, \bar{\sigma}^*\right)$ such that $\bar{m}^* = \langle m^*_1, \ldots, m^*_n\rangle$, $\bar{\sigma}^* = (\sigma^*_1, \ldots, \sigma^*_n)$ and for the fully bound bitwisely described subprogram $\bar{P}^{*\prime} = \left(\bar{f}^* = (f^*_1, \ldots, f^*_n), \tau^*_1, \ldots, \tau^*_l\right)$ of $\bar{P}^*$, there is $(\Delta^*, \tau^*_i, \bar{m}^*_i, \bar{\sigma}^*_i) \in S$ for some (unique) $\bar{m}^*_i \in \bar{\mathcal{M}}$ and $\bar{\sigma}^*_i \in \bar{\Sigma}$ for all $i \in [l]$. B then computes $\bar{m}^{**} := \langle m^{**}_1, \ldots, m^{**}_n\rangle = \bar{P}^{*\prime}(\bar{m}^*_1, \ldots, \bar{m}^*_l)$ and $\bar{\sigma}^{**} := (\sigma^{**}_1, \ldots, \sigma^{**}_n) \leftarrow \bar{T}.\mathrm{Eval}\left(\bar{ek}, \bar{f}^*, (\bar{m}^*_1, \bar{\sigma}^*_1), \ldots, (\bar{m}^*_l, \bar{\sigma}^*_l)\right)$. If the calculated message-tag pair is different from the forgery attempt, $(\bar{m}^*, \bar{\sigma}^*) \neq (\bar{m}^{**}, \bar{\sigma}^{**})$, then B outputs

$$\left(\Delta^*, \left(f^*_{i^*}, ((\tau_s, t))_{(s,t)\in[l]\times[n]}\right), m^*_{i^*}, \sigma^*_{i^*}\right)$$

as a forgery attempt where i^* is the smallest number of the nonempty set $\{j \mid (m^*_j, \sigma^*_j) \neq (m^{**}_j, \sigma^{**}_j)\}$. Otherwise, B outputs nothing and halts.

Note that in A's perspective, B acts exactly the same as the adversary of $\mathbf{Game}^{\overline{\mathrm{MDHA}}}_{\bar{T},A}(\lambda)$. Also, if the forgery attempt $\left(\Delta^*, \bar{P}^*, \bar{m}^*, \bar{\sigma}^*\right)$ that A made is a forgery, then since $1 \leftarrow \bar{T}.\mathrm{Verify}(\bar{sk}, \Delta^*, \bar{P}^*, \bar{m}^*, \bar{\sigma}^*)$, we have $1 \leftarrow T.\mathrm{Verify}(sk, \Delta^*, P^*_{i^*}, m^*_{i^*}, \sigma^*_{i^*})$ where $P^*_{i^*} := \left(f^*_{i^*}, ((\tau_s, t))_{(s,t)\in[l]\times[n]}\right)$ for i^* defined as above. Therefore, $(\Delta^*, P^*_{i^*}, m^*_{i^*}, \sigma^*_{i^*})$ becomes a forgery of T. In other words, B wins if A outputs a forgery, and we may write

$$\mathbf{Adv}^{\overline{\mathrm{MDHA}}}_{\bar{T},A}(\lambda) \leq \mathbf{Adv}^{\overline{\mathrm{MDHA}}}_{T,B}(\lambda). \qquad \square$$

VIII. CONCLUSIONS

In this work, we have proposed a new security notion for HAEs that implies privacy and authenticity at the same time. Our new security notion satisfies all the previous security notions for HAEs. To make an FHAE that satisfies the new security notion, we have designed a generic construction that combines FHE and MDFHA.

Our construction is essentially an homomorphic version of the encrypt-then-authenticate construction, while we added another authentication independent of the message for our stronger security definition.

Fully homomorphic encryptions and fully homomorphic authenticators typically have very large ciphertext expansion, and their real-life, practical performance is sometimes not so satisfactory. Since our construction follows the 'encrypt-then-authenticate' paradigm, our construction has large ciphertext expansion and less-than-ideal performances, while it is true that existing FHA schemes supporting amortized efficiency and satisfying adaptive security have similar imperfections. Our FHAE gives extra data privacy for free, with asymptotically comparable performance as those FHA schemes.

It would be a very interesting open problem to construct more efficient FHAE schemes than our current generic composition.

REFERENCES

[1] R. L. Rivest, L. Adleman, and M. L. Dertouzos, "On data banks and privacy homomorphisms," Proc. FOCS, vol. 4, pp. 169–180, 1978.
[2] C. Gentry, "Fully homomorphic encryption using ideal lattices," in Proc. STOC, pp. 169–178, 2009.
[3] R. Gennaro and D. Wichs, "Fully homomorphic message authenticators," in Proc. ASIACRYPT, Part II, pp. 301–320, 2013.
[4] S. Gorbunov, V. Vaikuntanathan, and D. Wichs, "Leveled fully homomorphic signatures from standard lattices," in Proc. STOC, pp. 469–477, 2015.
[5] C. Joo and A. Yun, "Homomorphic authenticated encryption secure against chosen-ciphertext attack," in Proc. ASIACRYPT, Part II, pp. 173–192, 2014.
[6] C. Gentry, A. Sahai, and B. Waters, "Homomorphic encryption from learning with errors: conceptually-simpler, asymptotically-faster, attribute-based," in Proc. CRYPTO, Part I, pp. 75–92, 2013.
[7] X. Boyen, X. Fan, and E. Shi, "Adaptively secure fully homomorphic signatures based on lattices." Cryptology ePrint Archive, Report 2014/916, 2014. https://eprint.iacr.org/2014/916.
[8] F. Luo, F. Wang, K. Wang, and K. Chen, "A more efficient leveled strongly-unforgeable fully homomorphic signature scheme," Information Sciences, vol. 480, pp. 70–89, 2019.
[9] C. Wang, B. Wu, and H. Yao, "Leveled adaptively strong-unforgeable identity-based fully homomorphic signatures," IEEE Access, vol. 8, pp. 119431–119447, 2020.
[10] D. Catalano, D. Fiore, and L. Nizzardo, "On the security notions for homomorphic signatures," in Proc. ACNS, pp. 183–201, 2018.
[11] P. Rogaway and T. Shrimpton, "A provable-security treatment of the key-wrap problem," in Proc. EUROCRYPT, pp. 373–390, 2006.
[12] H. KRAWCZYK and T. Rabin, "Chameleon signatures," in Proc. NDSS, pp. 143–154, 2000.
[13] M. Ajtai, "Generating hard instances of lattice problems," in Proc. STOC, pp. 99–108, 1996.
[14] D. Micciancio, "Almost perfect lattices, the covering radius problem, and applications to Ajtai's connection factor," SIAM Journal on Computing, vol. 34, no. 1, pp. 118–169, 2004.
[15] D. Micciancio and O. Regev, "Worst-case to average-case reductions based on Gaussian measures," in Proc. FOCS, pp. 372–381, 2004.
[16] D. Micciancio and C. Peikert, "Hardness of SIS and LWE with small parameters," in Proc. CRYPTO, Part I, pp. 21–39, 2013.
[17] M. Ajtai, "Generating hard instances of the short basis problem," in Proc. ICALP, pp. 1–9, 1999.
[18] C. Gentry, C. Peikert, and V. Vaikuntanathan, "Trapdoors for hard lattices and new cryptographic constructions," in Proc. STOC, pp. 197–206, 2008.

[19] J. Alwen and C. Peikert, "Generating shorter bases for hard random lattices," Theory of Computing Systems, vol. 48, no. 3, pp. 535–553, 2011.
[20] D. Micciancio and C. Peikert, "Trapdoors for lattices: simpler, tighter, faster, smaller," in Proc. EUROCRYPT, pp. 700–718, 2012.
[21] S. Agrawal, D. Boneh, and X. Boyen, "Efficient lattice (H)IBE in the standard model," in Proc. EUROCRYPT, pp. 553–572, 2010.

JEONGSU KIM received the B.S. and M.S. degree in mathematical sciences from Korea Advanced Institute of Science and Technology (KAIST), Daejeon, Republic of Korea in 2014 and 2017, respectively. He is currently pursuing the Ph.D. degree in computer science and engineering at Ulsan National Institute of Science and Technology (UNIST), Ulsan, Republic of Korea.

His research interests include post-quantum cryptography, quantum security and homomorphic cryptography.

AARAM YUN received the B.S. degree in mathematics from Korea Advanced Institute of Science and Technology (KAIST), Daejeon, Republic of Korea in 1995, and the Ph.D. degree in mathematics from Yale University, New Haven, USA in 2001.

He is currently an associate professor with Department of Cyber Security, Ewha Womans University, Seoul, Republic of Korea. Before joining Ewha, he has been with Ulsan National Institute of Science and Technology (UNIST), Ulsan, Republic of Korea until 2018. His research interests include post-quantum cryptography, quantum security and homomorphic cryptography.