Verifiable Computation on Outsourced Encrypted Data

Junzuo Lai[1,2], Robert H. Deng[3], Hweehwa Pang[3], and Jian Weng[1,⋆]

[1] Department of Computer Science, Jinan University, China
{laijunzuo,cryptjweng}@gmail.com
[2] The State Key Laboratory of Integrated Services Networks, Xidian University, China
[3] School of Information Systems, Singapore Management University, Singapore 178902
{robertdeng,hhpang}@smu.edu.sg

Abstract. On one hand, homomorphic encryption allows a cloud server to perform computation on outsourced encrypted data but provides no verifiability that the computation is correct. On the other hand, homomorphic authenticator, such as homomorphic signature with public verifiability and homomorphic MAC with private verifiability, guarantees authenticity of computation over outsourced data but does not provide data confidentiality. Since cloud servers are usually operated by third-party providers which are almost certain to be outside the trust domain of cloud users, neither homomorphic encryption nor homomorphic authenticator suffices for verifiable computation on outsourced encrypted data in the cloud. In this paper, we propose *verifiable* homomorphic encryption (VHE), which enables *verifiable* computation on outsourced encrypted data.

We first introduce a new cryptographic primitive called homomorphic encrypted authenticator (HEA), which may be of independent interest. Informally, HEA can be viewed as a homomorphic authenticator in which the authenticator itself does not leak any information about the message it authenticates. Next, we show that the *fully* homomorphic MAC scheme, proposed by Gennaro and Wichs recently, is a *fully* HEA with *weak* unforgeability in the sense that an adversary is not allowed to make verification queries. We then propose a *linearly* HEA which can tolerate *any* number of malicious verification queries, i.e., it achieves (*strong*) unforgeability. Finally, we define VHE formally, and give a generic construction of VHE based on homomorphic encryption and HEA. Instantiating the generic construction, we derive a *fully* VHE with *weak* verifiability as well as a *linearly* VHE with (*strong*) verifiability.

Keywords: Cloud Computing, Outsourced Encrypted Data, Verifiable Homomorphic Encryption.

⋆ Corresponding author.

M. Kutyłowski and J. Vaidya (Eds.): ESORICS 2014, Part I, LNCS 8712, pp. 273–291, 2014.

1 Introduction

Cloud computing has become increasingly popular because it offers users the illusion of having infinite computing resources, of which they can use as much as they need, without having to worry about how those resources are provided and managed. Since cloud servers are usually operated by third-party providers which are almost certain to be outside the trust domain of cloud users, the cloud computing paradigm raises many security and privacy concerns. One problem is how can users securely outsource their data to the cloud, and later entrust it to perform computation over the data. In a nutshell, this problem can be described in the following scenario. A user Alice has a large collection of data $m_1, \ldots, m_n$ and intends to outsource her data to the cloud. In order to prevent leakage of sensitive information to the cloud service provider, Alice first encrypts the data to produce ciphertexts $c_1, \ldots, c_n$, where c_i is the encryption of m_i. She then uploads the ciphertexts to the cloud, without having to keep a copy of the data due to her limited local storage capacity. At some later point, Alice wishes to derive some information from her data, such as the sum or mean of $m_1, \ldots, m_n$. For this purpose, Alice sends an instruction to the cloud server, specifying a program $\mathcal{P}$ to be executed on her data. The cloud server executes $\mathcal{P}$ over the ciphertexts and returns the result of the execution, c, to Alice. Alice then retrieves $\mathcal{P}(m_1, \ldots, m_n)$ from c. This problem has been addressed by the recent ground-breaking development of *fully homomorphic encryption* [1], which allows a cloud server to perform *any* computation over outsourced encrypted data. Unfortunately, existing fully homomorphic encryption schemes provide no guarantee that the cloud server performed the computation correctly. In the cloud computing setting, there may be incentives for a cloud server to try to cheat and return an incorrect result to the client. This may be related to the nature of the computation being performed, e.g., if the cloud server wants to convince the client of a particular result because it will have beneficial consequences for the server or the server may simply be minimizing the use of its own computational overhead. Errors can also occur due to faulty algorithm implementation or system failure. Thus, the client needs some guarantee that the result returned from the server is correct. In particular, the cloud server needs to convince Alice that c is the correct result of the computation $\mathcal{P}$ over ciphertexts $c_1, \ldots, c_n$, i.e., the ciphertext of $\mathcal{P}(m_1, \ldots, m_n)$.

Consider another scenario in cloud computing. Alice outsources her data $m_1, \ldots, m_n$ in plaintext to a cloud server, and later asks the server to run a program $\mathcal{P}$ over the outsourced data $(m_1, \ldots, m_n)$. The server computes $\mathcal{P}(m_1, \ldots, m_n)$ and sends the result m to Alice. The problem now is that Alice wants to be sure that $m = \mathcal{P}(m_1, \ldots, m_n)$. Homomorphic authenticator, including homomorphic signature [2–12] (for public verification) and homomorphic MAC [13, 14] (for private verification), is the cryptographic primitive that addresses this problem. Roughly speaking, a homomorphic authenticator scheme enables Alice using her secret key to produce an authenticator (called signature in homomorphic signature, or tag in homomorphic MAC) which authenticates a data item so that later, given a set of data $m_1, \ldots, m_n$ and the corresponding

authenticators $\sigma_1, \ldots, \sigma_n$, anyone can perform a computation $\mathcal{P}$ over $(\sigma_1, \ldots, \sigma_n)$ to generate an authenticator σ that authenticates $\mathcal{P}(m_1, \ldots, m_n)$. However, homomorphic authenticator does not maintain *confidentiality* of outsourced data. That is, the cloud server has total access to user's data since they are not encrypted.

The above observations motivate us to consider *verifiable* homomorphic encryption (VHE), which enables *verifiable* computation on outsourced encrypted data. Informally, a VHE scheme allows a user using her secret key to encrypt data $m_1, \ldots, m_n$ and obtain independent ciphertexts $c_1, \ldots, c_n$ so that later, given ciphertexts $c_1, \ldots, c_n$, anyone can execute a program $\mathcal{P}$ over $(c_1, \ldots, c_n)$ to generate a ciphertext c. The user using her secret key can then decrypt ciphertext c to obtain plaintext m and check whether $m = \mathcal{P}(m_1, \ldots, m_n)$. There is a trivial solution to construct VHE in which the user authenticates the output of a computation $\mathcal{P}$ over $(c_1, \ldots, c_n)$ by accessing all the ciphertexts, i.e., $c_1, \ldots, c_n$. Thus, VHE is only interesting if authenticity of the output of $\mathcal{P}$ over the ciphertexts $c_1, \ldots, c_n$ can be verified with significantly lower communication cost than that of simply transmitting $c_1, \ldots, c_n$ to the user. This is particularly important where the outsourced data are large in size.

A naive approach to construct VHE is to combine homomorphic encryption and homomorphic authenticator directly. In the above cloud computing scenario, before outsourcing her data $m_1, \ldots, m_n$ to a cloud server, Alice first runs the encryption algorithm of a homomorphic encryption scheme and the authentication algorithm of a homomorphic authenticator scheme on m_i, $i = 1, \ldots, n$, then she uploads the ciphertext of m_i, $c_i = (\tilde{c}_i, \sigma_i)$, to the server, where $\tilde{c}_i$ and σ_i are the outputs of the encryption and authentication algorithms, respectively. Later, when the server is asked to execute a program $\mathcal{P}$ on the ciphertexts, it runs the evaluation algorithms of the homomorphic encryption scheme and homomorphic authenticator scheme on $((\tilde{c}_1, \ldots, \tilde{c}_n), \mathcal{P})$ and $((\sigma_1, \ldots, \sigma_n), \mathcal{P})$, respectively, and returns the result $c = (\tilde{c}, \sigma)$ to Alice, where $\tilde{c}$ and σ are the outputs from the evaluation algorithms of the homomorphic encryption scheme and homomorphic authenticator scheme, respectively. The client decrypts $\tilde{c}$ to obtain message m and checks that the server correctly applied $\mathcal{P}$ to the ciphertexts by verifying that the authenticator σ authenticates the message $m = \mathcal{P}(m_1, \ldots, m_n)$. Unfortunately, homomorphic authenticator schemes provide no guarantee that the authenticator σ_i on m_i does not leak information about m_i. Indeed, with a homomorphic signature scheme, the signature σ_i on message m_i always leaks information about m_i since, given a message m, anyone can check whether σ_i is a valid signature on m. Thus, the above naive construction of VHE does not guarantee that the outsourced data $m_1, \ldots, m_n$ is *semantically secure*.

Our Contributions. We first introduce a new cryptographic primitive called homomorphic encrypted authenticator (HEA), which may be of independent interest, and formally define its *semantic security* and *unforgeability*. Informally, a HEA can be viewed as a homomorphic authenticator in which the authenticator does not leak any information about the message that it authenticates. Then, we show that the *fully* homomorphic MAC scheme, proposed by Gennaro and

Wichs [13] recently, is a *fully* HEA scheme with *weak* unforgeability, where the adversary is not allowed to make verification queries. We emphasize that a homomorphic MAC scheme is not necessarily a HEA scheme, since the tag in a homomorphic MAC scheme may leak information about the message it authenticates. For example, the homomorphic MAC schemes tolerating any number of malicious verification queries, proposed by Catalano and Fiore [14] recently, are not HEA schemes because anyone can retrieve the message from its tag in these schemes. While a *fully* HEA scheme with weak unforgeability allows anyone to perform *any* authenticated computation on authenticated data, it is only secure in the setting where the adversary cannot make verification queries to test if a maliciously constructed authenticator verifies correctly. In practice, this means that the user needs to abort and completely stop using the scheme whenever she gets the first authenticator that doesn't verify correctly, and this motivates us to seek HEA schemes with (*strong*) unforgeability which allows an adversary to make arbitrarily many verification queries.

We observe that, in a homomorphic signature scheme, an adversary cannot obtain any additional information by making verification queries since, given a signature, anyone (including the adversary) can check whether it is a valid signature on a message. Thus, we resort to homomorphic signature for constructing HEA schemes with (strong) unforgeability. As mentioned above, a homomorphic signature scheme may not be a HEA scheme; so we have to adopt some techniques to convert the former into the latter. Drawing on a *linearly* homomorphic signature scheme proposed by Freeman [8], we propose a *linearly* HEA which can tolerate *any* number of malicious verification queries, i.e., it achieves (*strong*) unforgeability.

Finally, we formally introduce the notion and security requirements of VHE, and provide a generic construction of VHE from homomorphic encryption and HEA. Instantiating the generic construction, we obtain a *fully* VHE with *weak* verifiability, which allows anyone to perform *any* verifiable computation on outsourced encrypted data but does not tolerate malicious verification queries, as well as a *linearly* VHE with (*strong*) verifiability, which allows anyone to perform *linear* verifiable computations on outsourced encrypted data and can tolerate *any* number of malicious verification queries.

Organization. The rest of this paper is organized as follows. Section 2 reviews related work. Section 3 provides some preliminaries. Section 4 introduces the new cryptographic primitive HEA, shows that the fully homomorphic MAC scheme proposed by Gennaro and Wichs [13] is a fully HEA in a weaker security model, and proposes a linearly HEA scheme in a full security model. The formal definition of VHE and the generic construction of VHE from homomorphic encryption and HEA are given in Section 5. Section 6 concludes the paper.

2 Related Work

We review related literature including non-interactive verifiable computation, fully homomorphic encryption, homomorphic signature/MAC, linearly homomorphic

structure-preserving signature and homomorphic authenticator-encryption. We refer the reader to [15] for discussions on other related effort, such as succinct non-interactive arguments of knowledge [16].

Non-Interactive Verifiable Computation. The notion of non-interactive verifiable computation was introduced by Gennaro et al. [17]. Non-interactive verifiable computation enables a computationally weak client to outsource the computation of a function to a server, which returns the result of the function evaluation as well as a non-interactive proof that the computation was carried out correctly, with the crucial requirement that verification of the proof needs substantially less *computational* effort than computing the function by the client from scratch. The existing non-interactive verifiable computation schemes [17–22] focus on delegating general functions. In order to achieve higher efficiency, there exist non-interactive verifiable computation schemes which permit only a very limited class of functions, such as polynomials [23–26, 15, 22] and set operations [27]. Non-interactive verifiable computation schemes either do not protect privacy of outsourced data from a malicious server, or the functions to be evaluated must be known at system setup, or the outsourced data must be fixed a-priori (i.e., a client cannot outsource her data incrementally). VHE does not suffer from any of the limitations of non-interactive verifiable computation mentioned above.

Our goals are very different from non-interactive verifiable computation. We seek to save the clients from storing large amount of data as well as to save on *communication* cost. The key requirement that makes our definition of VHE interesting is that the output of the program $\mathcal{P}$ over the ciphertexts $c_1, \ldots, c_n$ be *succinct*; otherwise, there is a trivial solution in which a client can verify the output of a computation $\mathcal{P}$ by simply being provided with the ciphertexts $c_1, \ldots, c_n$. The succinctness requirement ensures that the client can verify the output of a computation $\mathcal{P}$ over encrypted data with much smaller *communication* overhead than that of simply transmitting the encrypted data from the server to the client. VHE is especially useful when verifying computations that require a large amount of encrypted data as input but have a short output (e.g., computing the median in a large database).

Fully Homomorphic Encryption. The notion of fully homomorphic encryption (FHE) was first put forward by Rivest et al. [28]. However, only in the past few years have candidate FHE schemes been proposed. The first such scheme was constructed by Gentry [1]; his work inspired a tremendous amount of research effort on improving the efficiency of his scheme [29–35], realizations of FHE based on different assumptions [36–39], and so on.

Homomorphic Signature and MAC. Homomorphic authenticator in both the asymmetric setting (i.e., homomorphic signature) and the symmetric setting (i.e., homomorphic MAC) has been considered in many prior works. The notion of homomorphic signature was introduced by Johnson et al. [40]. Since then, many homomorphic signature schemes [2–8] for *linear functions* have been proposed, mainly because of the important application to *network coding* [41, 42]. Linearly homomorphic authenticator has also been considered in the context of

proofs of data possession and retrievability [43–45]. The work of Ahn et al. [10] and Attrapadung et al. [11, 12] considered a new security requirement of homomorphic signature, i.e., *context hiding*, which requires that a derived signature be indistinguishable from a fresh signature on the same message. In a recent breakthrough, Boneh and Freeman [9] showed how to use ideal lattices to construct homomorphic signature for bounded degree polynomials; this scheme is currently the only one that goes beyond linear functions.

Gennaro and Wichs [13] introduced fully homomorphic MAC (i.e., homomorphic MAC for any computation) and gave a concrete construction which is only proven secure in a weaker model where an adversary cannot ask verification queries. We will show later in the paper that the fully homomorphic MAC scheme in [13] is in fact a fully HEA scheme. Catalano and Fiore [14] presented efficient realizations of homomorphic MAC that tolerate verification queries, for a restricted class of computations (i.e., arithmetic circuits with polynomially-bounded degree). The homomorphic MAC schemes proposed in [14] are not HEA schemes, and how to convert these schemes into HEA is an interesting open problem.

Linearly Homomorphic Structure-Preserving Signatures. Structure-preserving signatures (SPS) [46, 47] are signature schemes where public keys, messages and signatures all consist of elements of a group over which a bilinear map is efficiently computable. Recently, Libert et al. [48] introduced and realized the notion of linearly homomorphic structure-preserving signature (LHSPS), which is similar to SPS but equipped with a linearly homomorphic property. Catalano et al. [49] followed their work and proposed some new methodologies. Libert et al. [48] showed that LHSPS enables linear verifiable computations on outsourced encrypted data (i.e., linearly VHE), but their treatment is informal and decryption in their system takes polynomial time in the size of the message space (i.e., their system can only be used to encrypt short messages). In this paper, we present the notion and security models of VHE formally, give a generic construction for VHE, and derive a fully VHE scheme which is proven secure in a weaker security model and a linearly VHE scheme which is proven secure in a full security model.

Homomorphic Authenticator-Encryption. Gennaro and Wichs [13] showed that their proposed homomorphic MAC can be extended to homomorphic authenticator-encryption, also called homomorphic authenticated encryption in [50]. A homomorphic authenticator-encryption scheme is a homomorphic authenticator scheme with an additional decrypt algorithm, which allows a user with a secret key to retrieve the authenticated message from an authenticator. Our notion of HEA is different from homomorphic authenticator-encryption. A HEA scheme only requires that an authenticator not leak information about the authenticated message; thus HEA is a weaker cryptographic primitive than homomorphic authenticator-encryption. In fact, homomorphic authenticator-encryption, which also enables verifiable computation on outsourced encrypted data, is similar to our notion of VHE. However, the schemes proposed in [13, 50]

cannot tolerate malicious verification queries. Compared with their work, we investigate the relationships among homomorphic encryption, homomorphic signature/MAC and VHE, and provide a general method to construct VHE. We also propose a linearly VHE scheme which can tolerate any number of malicious verification queries.

3 Preliminaries

If S is a set, then $s \xleftarrow{\$} S$ denotes the operation of picking an element s uniformly at random from S. Let $\mathbb{N}$ denote the set of natural numbers. If $n \in \mathbb{N}$ then $[n]$ denotes the set $\{1, \ldots, n\}$. If $\lambda \in \mathbb{N}$ then 1^λ denotes the string of λ ones. Let $z \leftarrow \mathsf{A}(x, y, \ldots)$ denote the operation of running an algorithm A with inputs $(x, y, \ldots)$ and output z. A function $f(\lambda)$ is *negligible* if for every $c > 0$ there exists a λ_c such that $f(\lambda) < 1/\lambda^c$ for all $\lambda > \lambda_c$.

3.1 Bilinear Groups

Let $\mathcal{G}$ be an algorithm that takes as input a security parameter λ and outputs a tuple $(p, \mathbb{G}, \mathbb{G}_T, e)$, where $\mathbb{G}$ and $\mathbb{G}_T$ are multiplicative cyclic groups of prime order p, and $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ is a map that possesses the following properties:

1. **Bilinearity:** $e(g^a, h^b) = e(g, h)^{ab}$ for all $g, h \in \mathbb{G}$ and $a, b \in \mathbb{Z}_p^*$.
2. **Non-degeneracy:** $e(g, h) \neq 1$ whenever $g, h \neq 1_{\mathbb{G}}$.
3. **Computable:** efficient computability for any input pair.

We refer to the tuple $(p, \mathbb{G}, \mathbb{G}_T, e)$ as a *bilinear group*. We consider the following problems in bilinear groups.

q-Strong Diffie-Hellman (q-SDH) Problem. The q-SDH problem in $\mathbb{G}$ is defined as follows: Given a tuple $(\bar{g}, \bar{g}^\alpha, \ldots, \bar{g}^{\alpha^q})$ as input for randomly chosen $\bar{g} \xleftarrow{\$} \mathbb{G}$ and $\alpha \xleftarrow{\$} \mathbb{Z}_p^*$, output a pair $(\bar{g}^{1/(\alpha+\vartheta)}, \vartheta)$ where $\vartheta \in \mathbb{Z}_p^*$. The advantage of an algorithm $\mathcal{A}$ in solving q-SDH problem is defined as $|\mathsf{Pr}[\mathcal{A}(\bar{g}, \bar{g}^\alpha, \ldots, \bar{g}^{\alpha^q}) = (\bar{g}^{1/(\alpha+\vartheta)}, \vartheta)]|$, where the probability is over the random choices of $\bar{g} \in \mathbb{G}, \alpha \in \mathbb{Z}_p^*$, and the random bits of $\mathcal{A}$.

Definition 1. *We say that the q-SDH assumption holds in $\mathbb{G}$ if all probabilistic polynomial time algorithms have at most a negligible advantage in solving the q-SDH problem in $\mathbb{G}$.*

Decision Linear (DLN) Problem [51]. The DLN problem in $\mathbb{G}$ is defined as follows: Given a tuple $(\bar{g}, \tilde{g}, \hat{g}, \bar{g}^x, \tilde{g}^y, \hat{g}^z)$ as input, output 1 if $x + y = z$ and 0 otherwise. The advantage of an algorithm $\mathcal{A}$ in solving the DLN problem is defined as $|\mathsf{Pr}[\mathcal{A}(\bar{g}, \tilde{g}, \hat{g}, \bar{g}^x, \tilde{g}^y, \hat{g}^z) = 1 : \bar{g}, \tilde{g}, \hat{g} \xleftarrow{\$} \mathbb{G},\ x, y, z \xleftarrow{\$} \mathbb{Z}_p^*] - \mathsf{Pr}[\mathcal{A}(\bar{g}, \tilde{g}, \hat{g}, \bar{g}^x, \tilde{g}^y, \hat{g}^{x+y}) = 1 : \bar{g}, \tilde{g}, \hat{g} \xleftarrow{\$} \mathbb{G},\ x, y \xleftarrow{\$} \mathbb{Z}_p^*]|$, where the probability is over the random choices of $\bar{g}, \tilde{g}, \hat{g} \in \mathbb{G}$ and $x, y, z \in \mathbb{Z}_p^*$, and the random bits of $\mathcal{A}$.

Definition 2. *We say that the DLN assumption holds in $\mathbb{G}$ if all probabilistic polynomial time algorithms have at most a negligible advantage in solving the DLN problem in $\mathbb{G}$.*

4 Homomorphic Encrypted Authenticator

Informally, a homomorphic encrypted authenticator (HEA) can be viewed as a homomorphic authenticator, where the authenticator on a message does not leak any information about the message. Before presenting the definition of HEA formally, we need to establish some syntax for specifying which data is being authenticated and over which data a program $\mathcal{P}$ should be evaluated. We recall the notion of labeled data and programs introduced by Gennaro and Wichs in [13].

Labeled Data and Programs. Whenever a user wants to authenticate some data item, she chooses a *label* $\tau \in \{0,1\}^*$ for it, and the authentication algorithm authenticates the data with respect to the label τ. A labeled program $\mathcal{P}$ consists of a tuple $(f, \tau_1, \ldots, \tau_k)$ where $f : \mathbb{F}^k \to \mathbb{F}$ is a circuit/function, and $\tau_1, \ldots, \tau_k$ are the *labels* of the input nodes of f. Given some labeled programs $\mathcal{P}_1, \ldots, \mathcal{P}_t$ and a function $g : \mathbb{F}^k \to \mathbb{F}$, the *composed program*, denoted by $\mathcal{P}^* = g(\mathcal{P}_1, \ldots, \mathcal{P}_k)$, corresponds to evaluating g on the outputs of $\mathcal{P}_1, \ldots, \mathcal{P}_k$. The labeled inputs of $\mathcal{P}^*$ are all the distinct labeled inputs of $\mathcal{P}_1, \ldots, \mathcal{P}_k$. We denote by $\mathcal{I}_\tau = (g_{id}, \tau)$ the *identity program* with label τ where g_{id} is the canonical identity function and τ is some label. Notice that any program $\mathcal{P} = (f, \tau_1, \ldots, \tau_k)$ can be expressed as the composition of identity programs $\mathcal{P} = f(\mathcal{I}_{\tau_1}, \ldots, \mathcal{I}_{\tau_k})$.

Homomorphic Encrypted Authenticator. A homomorphic encrypted authenticator scheme consists of the following four algorithms:

- $\mathsf{Setup}(1^\lambda)$ takes as input a security parameter λ. It outputs a pair of public key PK and secret key SK. The public key PK defines a message space $\mathcal{M}$ and a set $\mathcal{F}$ of "admissible" functions $f : \mathcal{M}^k \to \mathcal{M}$.
- $\mathsf{Auth}(\mathsf{SK}, \tau, m)$ takes as input secret key SK, a label $\tau \in \{0,1\}^*$ and a message $m \in \mathcal{M}$. It outputs an authenticator σ.
- $\mathsf{Ver}(\mathsf{SK}, m, \mathcal{P}, \sigma)$ takes as input secret key SK, a message $m \in \mathcal{M}$, a labeled program $\mathcal{P}$ and an authenticator σ. It outputs either 0 (reject) or 1 (accept).
- $\mathsf{Eval}(\mathsf{PK}, f, \boldsymbol{\sigma})$ takes as input public key PK, a function $f \in \mathcal{F}$ and a vector of authenticators $\boldsymbol{\sigma} = (\sigma_1, \ldots, \sigma_k)$. It outputs a new authenticator σ.

For correctness, we require that for each $(\mathsf{PK}, \mathsf{SK})$ output by $\mathsf{Setup}(1^\lambda)$, the following properties hold:

1. For all labels $\tau \in \{0,1\}^*$ and all $m \in \mathcal{M}$, if $\sigma \leftarrow \mathsf{Auth}(\mathsf{SK}, \tau, m)$, then $\mathsf{Ver}(\mathsf{SK}, m, \mathcal{I}_\tau, \sigma) = 1$.
2. Given an admissible function $g : \mathcal{M}^k \leftarrow \mathcal{M}$ and any set of message/program/authenticator triples $\{(m_i, \mathcal{P}_i, \sigma_i)\}_{i=1}^k$ such that $\mathsf{Ver}(\mathsf{SK}, m_i, \mathcal{P}_i, \sigma_i) = 1$, if $m = g(m_1, \ldots, m_k)$, $\mathcal{P} = g(\mathcal{P}_1, \ldots, \mathcal{P}_k)$ and $\sigma = \mathsf{Eval}(\mathsf{PK}, g, (\sigma_1, \ldots, \sigma_k))$, then $\mathsf{Ver}(\mathsf{SK}, m, \mathcal{P}, \sigma) = 1$.

The above requirements capture the basic correctness of computing over freshly authenticated data, as well as the composability of computing over the authenticated outputs of prior computations. If the set of admissible functions $\mathcal{F}$ of a HEA scheme consists of *linear* functions (resp. *any* functions) from $\mathcal{M}^k$ to $\mathcal{M}$, then we say that the HEA scheme is a *linearly* (resp. *fully*) HEA scheme.

4.1 Security Model for Homomorphic Encrypted Authenticator

We now introduce the security requirements of HEA, including *semantic security* and *unforgeability*. Informally, *semantic security* requires that an authenticator σ of a message m with label τ should not leak any information about m. Let σ_i be an authenticator on message m_i with label τ_i for $i = 1, \ldots, k$. *Unforgeability* requires that given $(\sigma_1, \ldots, \sigma_k)$, it should be impossible to output an authenticator σ and an admissible function f such that σ is a valid authenticator on a message-program pair $(m, \mathcal{P} = (f, \tau_1, \ldots, \tau_k))$ and $m \neq f(m_1, \ldots, m_k)$.

The semantic security of HEA is defined in terms of the following game, played between a challenger and an adversary $\mathcal{A}$:

Setup. The challenger runs $\mathsf{Setup}(1^\lambda)$ to obtain a pair of public key PK and secret key SK. It gives the public key PK to adversary $\mathcal{A}$ and keeps SK to itself. It also initializes a list $T = \emptyset$.

Authentication queries. The adversary $\mathcal{A}$ adaptively queries the challenger for authenticators. $\mathcal{A}$ submits a message $m \in \mathcal{M}$. If there exists a tuple (τ, m) in T, the challenger computes $\sigma \leftarrow \mathsf{Auth}(\mathsf{SK}, \tau, m)$; otherwise, the challenger chooses a fresh label $\tau \in \{0,1\}^*$, updates the list $T = T \cup (\tau, m)$ and computes $\sigma \leftarrow \mathsf{Auth}(\mathsf{SK}, \tau, m)$. Then, the challenger gives the authenticator σ to $\mathcal{A}$.

Challenge. Adversary $\mathcal{A}$ submits a label $\tau^* \in \{0,1\}^*$ and two messages $m_0, m_1 \in \mathcal{M}$. The challenger selects a random bit $\beta \in \{0,1\}$, computes $\sigma^* \leftarrow \mathsf{Auth}(\mathsf{SK}, \tau^*, m_\beta)$ and sends σ^* to the adversary.

Guess. Adversary $\mathcal{A}$ outputs its guess $\beta' \in \{0,1\}$ for β and wins the game if $\beta = \beta'$.

The advantage of the adversary in this game is defined as $|\mathsf{Pr}[\beta = \beta'] - \frac{1}{2}|$ where the probability is taken over the random bits used by the challenger and the adversary.

Definition 3. *A HEA scheme is* semantically secure *if all probabilistic polynomial time adversaries have at most a negligible advantage in this security game.*

The unforgeability of HEA is defined in terms of the following game, played between a challenger and an adversary $\mathcal{A}$:

Setup. The challenger runs $\mathsf{Setup}(1^\lambda)$ to obtain a pair of public key PK and secret key SK. It gives the public key PK to adversary $\mathcal{A}$ and keeps SK to itself. It also initializes a list $T = \emptyset$.

Queries. Adversary $\mathcal{A}$ adaptively issues the following queries:

- *Authentication queries.* Adversary $\mathcal{A}$ submits a message $m \in \mathcal{M}$. If there exists a tuple (τ, m) in T, the challenger computes $\sigma \leftarrow \mathsf{Auth}(\mathsf{SK}, \tau, m)$; otherwise, the challenger chooses a fresh label $\tau \in \{0,1\}^*$, updates the list $T = T \cup (\tau, m)$ and computes $\sigma \leftarrow \mathsf{Auth}(\mathsf{SK}, \tau, m)$. Then, the challenger gives the authenticator σ to $\mathcal{A}$.
- *Verification queries.* Adversary $\mathcal{A}$ submits $(m, \mathcal{P}, \sigma)$ and the challenger replies with the output of $\mathsf{Ver}(\mathsf{SK}, m, \mathcal{P}, \sigma)$.

Output. Adversary $\mathcal{A}$ outputs a message m^*, a labeled program $\mathcal{P}^* = (f^*, \tau_1^*, \ldots, \tau_k^*)$ and an authenticator σ^*.

The adversary *wins* if $\mathsf{Ver}(\mathsf{SK}, m^*, \mathcal{P}^*, \sigma^*) = 1$ and one of the following conditions hold:

1. there exists $i \in \{1, \ldots, k\}$ such that $(\tau_i^*, \cdot) \notin T$ (a *Type 1 forgery*),
2. T contains tuples $(\tau_1^*, m_1), \ldots, (\tau_k^*, m_k)$, for some message $m_1, \ldots, m_k$, and $m^* \neq f^*(m_1, \ldots, m_k)$ (a *Type 2 forgery*).

Informally, in a Type 1 forgery the adversary produces a valid authenticator σ on a message-program pair $(m^*, \mathcal{P}^* = (f^*, \tau_1^*, \ldots, \tau_k^*))$ where no message was ever authenticated under the label τ_i^* involved in the forgery, whereas in a Type 2 forgery the adversary produces a valid authenticator σ on a message-program pair $(m^*, \mathcal{P}^* = (f^*, \tau_1^*, \ldots, \tau_k^*))$ where m^* is not the correct output of the labeled program $\mathcal{P}^*$ when executed on previously authenticated message $(m_1, \ldots, m_k)$.

The advantage of the adversary in this game is defined as $|\mathsf{Pr}[\mathcal{A}\ wins]|$ where the probability is taken over the random bits used by the challenger and the adversary.

Definition 4. *A HEA scheme is (*strongly*) unforgeable, or simply unforgeable, if all probabilistic polynomial time adversaries have at most a negligible advantage in this security game.*

We say that a HEA scheme is *weakly* unforgeable (or unforgeable without verification queries) if in the above security game the adversary cannot make verification queries.

4.2 Proposed HEA Constructions

In this subsection, we first show that the fully homomorphic authenticator scheme proposed by Gennaro and Wichs [13] recently, is a secure fully HEA with weak unforgeability. Drawing on the linearly homomorphic signature scheme proposed by Freeman [8], we then present a secure linearly HEA scheme which can tolerate any number of malicious verification queries, i.e., it achieves (strong) unforgeability.

A Secure Fully HEA with Weak Unforgeability. In [13], Gennaro and Wichs proposed a fully homomorphic authenticator scheme and proved that it is *unforgeable without verification queries*. Observe that in an authenticator $\sigma = (c_1, \ldots, c_\lambda, \nu)$ on a message m with label τ, $\nu = F_K(\tau)$ is a value independent of the message m and c_i, $i \in [\lambda]$, is a ciphertext of a homomorphic encryption scheme HE. If the underlying fully homomorphic encryption scheme HE is semantically secure, then for each $i \in [\lambda]$, c_i does not leak any information about m, thus σ will not leak any information about m. That is, the fully homomorphic authenticator proposed in [13] is also *semantically secure*. To sum up, the fully homomorphic authenticator proposed by Gennaro and Wichs [13], is a secure fully HEA with weak unforgeability.

As shown in [13], there is an efficient attack against the scheme in the setting of security *with* verification queries. That is, the fully HEA scheme with weak

unforgeability is only secure in the setting where the adversary cannot make verification queries to test if a maliciously constructed authenticator verifies correctly. In practice, this means that the user needs to abort and completely stop using the scheme whenever she gets the first authenticator that doesn't verify correctly. This motivates the need for HEA schemes with (*strong*) unforgeability that allow the adversary to make arbitrarily many verification queries.

A Secure Linearly HEA with (Strong) Unforgeability. Drawing on the linearly homomorphic signature scheme proposed by Freeman [8], which is based on the Boneh-Boyen (BB) signature scheme [52], we now show how to construct a *linearly* HEA scheme which can tolerate any number of malicious verification queries, i.e., it achieves (strong) unforgeability. We emphasize that a linearly homomorphic signature scheme is not necessarily a linearly HEA scheme; so we have to adopt some techniques to convert the former into the latter.

In the proposed linearly HEA construction, we also use the notion of "data set" introduced in [8]. That is, each set of messages is grouped together into a "data set" or "file", and each file is associated with a label τ that serves to bind the messages together in that file. Therefore, in our proposed construction, the algorithm Setup includes an additional input parameter, k, which indicates the maximum data size of a file; and the algorithm Auth includes an additional input parameter, an index i, which indicates that the authenticated message is the ith message in the file. Verifiable homomorphic operations in our scheme only apply to the data associated with the same label. That is, for an admissible labeled program $\mathcal{P} = (f, \tau_1, \ldots, \tau_k)$, we require that $\tau_1 = \cdots = \tau_k$; thus, for simplicity, we denote by (f, τ) a labeled program $\mathcal{P}$. Since the client can group as many related messages as possible into a file (i.e., associated with an identical label), the requirement of the admissible labeled program should not constrain the usability of a general storage service overly.

Concretely, the proposed linearly HEA scheme consists of the following algorithms:

$\mathsf{Setup}(1^\lambda, k)$ Given a security parameter λ and a maximum data size k, the setup algorithm runs $\mathcal{G}(1^\lambda)$ to obtain a bilinear group $(p, \mathbb{G}, \mathbb{G}_T, e)$. Next, it chooses $g, u, v_1, \ldots, v_k, h, h_0 \in \mathbb{G}$ and $\alpha, a, b \in \mathbb{Z}_p^*$ uniformly at random. Then, it sets $g_1 = g^\alpha, h_1 = h_0^{1/a}, h_2 = h_0^{1/b}$ (note that $h_1^a = h_2^b = h_0$), and chooses a collision-resistant hash function $H : \{0,1\}^* \to \mathbb{Z}_p^*$. The public key is published as $\mathsf{PK} = (g, g_1, u, v_1, \ldots, v_k, h, h_0, h_1, h_2, H)$ and the secret key is $\mathsf{SK} = (\alpha, a, b)$.

The message space of our proposed scheme is $\mathbb{F}_p$ and the set of admissible functions $\mathcal{F}$ is all $\mathbb{F}_p$-linear functions from $\mathbb{F}_p^k$ to $\mathbb{F}_p$. We represent a function $f \in \mathcal{F}$ as the vector $(c_1, \ldots, c_k) \in \mathbb{F}_p^k$, i.e., $f(m_1, \ldots, m_k) = \sum_{i=1}^k c_i m_i$.

$\mathsf{Auth}(\mathsf{SK}, \tau, m, i)$ Given secret key SK, a label $\tau \in \{0,1\}^*$, a message $m \in \mathbb{F}_p$ and an index $i \in \{1, \ldots, k\}$, it chooses $r_1, r_2, s \in \mathbb{Z}_p^*$ uniformly at random. Then, it outputs the authenticator $\sigma = (\tilde{C}, C_0, C_1, C_2, s)$, where

$$\tilde{C} = g^{1/(\alpha + H(\tau))},\ C_0 = h_0^{r_1 + r_2} (h^m u^s v_i)^{1/(\alpha + H(\tau))},\ C_1 = h_1^{r_1},\ C_2 = h_2^{r_2}.$$

Note that the index i indicates that m is the ith message in the file associated with label τ.

$\mathsf{Ver}(\mathsf{SK}, m, \mathcal{P} = (f, \tau), \sigma)$ Given secret key SK, a message $m \in \mathbb{F}_p$, a label τ, a function $f = (c_1, \ldots, c_k) \in \mathbb{F}_p^k$ and an authenticator $\sigma = (\tilde{C}, C_0, C_1, C_2, s)$, it checks whether $e(g_1 g^{H(\tau)},\ \tilde{C}) = e(g,\ g)$ and $e(C_0/(C_1^a C_2^b),\ g) = e(\tilde{C},\ h^m u^s \cdot (\prod_{i=1}^k v_i^{c_i}))$. If so, it outputs 1 (accept); otherwise it outputs 0 (reject).

$\mathsf{Eval}(\mathsf{PK}, f, \boldsymbol{\sigma})$ Given public key PK, a function $f = (c_1, \ldots, c_k) \in \mathbb{F}_p^k$ and a vector of authenticators $\boldsymbol{\sigma} = (\sigma_1, \ldots, \sigma_k)$ where $\sigma_i = (\tilde{C}^{(i)}, C_0^{(i)}, C_1^{(i)}, C_2^{(i)}, s^{(i)})$ for $i = 1, \ldots, k$, it outputs a new authenticator $\sigma = (\tilde{C}, C_0, C_1, C_2, s)$ where

$$\tilde{C} = \tilde{C}^{(1)},\ C_0 = \prod_{i=1}^k (C_0^{(i)})^{c_i},\ C_1 = \prod_{i=1}^k (C_1^{(i)})^{c_i},\ C_2 = \prod_{i=1}^k (C_2^{(i)})^{c_i},\ s = \sum_{i=1}^k c_i s^{(i)}.$$

Correctness. We show that the proposed homomorphic encrypted authenticator scheme satisfies the correctness properties of HEA.

1. Let $\tau \in \{0,1\}^*$ be a label, $m \in \mathbb{F}_p$ be a message and $i \in \{1, \ldots, k\}$ be an index. Suppose $\sigma = (\tilde{C}, C_0, C_1, C_2, s) \leftarrow \mathsf{Auth}(\mathsf{SK}, \tau, m, i)$. We now show that $\mathsf{Ver}(\mathsf{SK}, m, \mathcal{I}_\tau, \sigma) = 1$. Observe that $\tilde{C} = g^{1/(\alpha + H(\tau))},\ C_0/(C_1^a C_2^b) = h_0^{r_1 + r_2} (h^m u^s v_i)^{1/(\alpha+H(\tau))} / (h_1^{ar_1} h_2^{br_2}) = (h^m u^s v_i)^{1/(\alpha+H(\tau))}$.
Thus, we have $e(g_1 g^{H(\tau)}, \tilde{C}) = e(g_1 g^{H(\tau)}, g^{1/(\alpha+H(\tau))}) = e(g, g)$,

$$\begin{aligned} e(C_0/(C_1^a C_2^b), g) &= e((h^m u^s v_i)^{1/(\alpha+H(\tau))}, g) = e(h^m u^s v_i, g^{1/(\alpha+H(\tau))}) \\ &= e(\tilde{C}, h^m u^s v_i). \end{aligned}$$

It follows that $\mathsf{Ver}(\mathsf{SK}, m, \mathcal{I}_\tau, \sigma) = 1$.

2. Let $\tau \in \{0,1\}^*$ be a label, and $f', f_1, \ldots, f_k$ be linear functions represented as vectors in $\mathbb{F}_p^k$, with $f' = (c_1, \ldots, c_k)$ and $f_i = (d_{i1}, \ldots, d_{ik})$ for $i = 1, \ldots, k$. Suppose $\boldsymbol{\sigma} = (\sigma_1, \ldots, \sigma_k)$ is a vector of authenticators with $\sigma_i = (\tilde{C}^{(i)}, C_0^{(i)}, C_1^{(i)}, C_2^{(i)}, s^{(i)})$ for $i = 1, \ldots, k$, such that $\mathsf{Ver}(\mathsf{SK}, m_i, \mathcal{P}_i = (f_i, \tau), \sigma_i) = 1$ for some $m_i \in \mathbb{F}_p$. We show that $\mathsf{Ver}(\mathsf{SK}, f'(m_1, \ldots, m_k), \mathcal{P} = (f' \circ \boldsymbol{f}, \tau), \mathsf{Eval}(\mathsf{PK}, f', \boldsymbol{\sigma})) = 1$, where $f' \circ \boldsymbol{f}$ denotes the function that sends $\boldsymbol{x} = (x_1, \ldots, x_k)$ to $f'(f_1(\boldsymbol{x}), \ldots, f_k(\boldsymbol{x}))$. Note that $f' \circ \boldsymbol{f}$ can be represented as a vector $(d_1, \ldots, d_k) \in \mathbb{F}_p^k$ where $d_i = \sum_{j=1}^k c_j d_{ji}$ for $i = 1, \ldots, k$.
Since $\mathsf{Ver}(\mathsf{SK}, m_i, \mathcal{P}_i = (f_i, \tau), \sigma_i) = 1$ for $i = 1, \ldots, k$, we have

$$\begin{gathered} \tilde{C}^{(i)} = g^{1/(\alpha+H(\tau))},\ C_0^{(i)} = h_0^{r_{i1} + r_{i2}} (h^{m_i} u^{s^{(i)}} \cdot \textstyle\prod_{j=1}^k v_j^{d_{ij}})^{1/(\alpha+H(\tau))}, \\ C_1^{(i)} = h_1^{r_{i1}},\ C_2^{(i)} = h_2^{r_{i2}}, \end{gathered}$$

for some random $r_{i1}, r_{i2} \in \mathbb{Z}_p^*$. Let $\mathsf{Eval}(\mathsf{PK}, f', \boldsymbol{\sigma}) = \sigma = (\tilde{C}, C_0, C_1, C_2, s)$. We have $\tilde{C} = g^{1/(\alpha+H(\tau))},\ C_1 = h_1^{r_1},\ C_2^{(i)} = h_2^{r_2},\ C_0 = h_0^{r_1+r_2} (h^m u^s \cdot \prod_{i=1}^k v_i^{\sum_{j=1}^k c_j d_{ji}})^{1/(\alpha+H(\tau))} = h_0^{r_1+r_2} (h^m u^s \cdot \prod_{i=1}^k v_i^{d_i})^{1/(\alpha+H(\tau))}$, where $m = \sum_{i=1}^k c_i m_i, r_1 = \sum_{i=1}^k c_i r_{i1}, r_2 = \sum_{i=1}^k c_i r_{i2}$ and $s = \sum_{i=1}^k c_i s^{(i)}$. Observe that $C_0/(C_1^a C_2^b) = (h^m u^s \cdot \prod_{i=1}^k v_i^{d_i})^{1/(\alpha+H(\tau))}$. Thus,

$e(g_1 g^{H(\tau)}, \tilde{C}) = e(g_1 g^{H(\tau)}, g^{1/(\alpha+H(\tau))}) = e(g, g)$, and $e(C_0/(C_1^a C_2^b), g) = e((h^m u^s \prod_{i=1}^k v_i^{d_i})^{1/(\alpha+H(\tau))}, g) = e(\tilde{C}, h^m u^s \prod_{i=1}^k v_i^{d_i})$. It follows that $\mathsf{Ver}(\mathsf{SK}, m, \mathcal{P} = (f' \circ \boldsymbol{f}, \tau), \sigma) = 1$, where $m = \sum_{i=1}^k c_i m_i = f'(m_1, \ldots, m_k)$ and $\sigma = \mathsf{Eval}(\mathsf{PK}, f', \boldsymbol{\sigma})$.

Security. We state the security theorems of our proposed scheme, including semantic security and unforgeability. The proofs of the security theorems are given in the full version of this paper.

Theorem 1. *If the DLN assumption holds in $\mathbb{G}$, then the proposed HEA scheme is semantically secure.*

Theorem 2. *If the q-SDH assumption holds in $\mathbb{G}$, the BB signature scheme is strongly unforgeable against a weak chosen message attack and the hash function H is collision-resistant, then the proposed HEA scheme is unforgeable.*

5 Verifiable Homomorphic Encryption

Informally, a verifiable homomorphic encryption (VHE) is a symmetric-key homomorphic encryption which enables *verifiable* computation on outsourced encrypted data. In a VHE scheme, a user with secret key SK can encrypt messages $m_1, \ldots, m_k$ to obtain k independent ciphertexts $c_1, \ldots, c_k$. Given ciphertexts $c_1, \ldots, c_k$ and an admissible function f, anyone can compute a ciphertext c on the value $f(m_1, \ldots, m_k)$. The user can then decrypt ciphertext c to obtain message m with secret key SK, and check whether $m = f(m_1, \ldots, m_k)$.

We also use the syntax of *labeled data and programs* for specifying which data is being encrypted and which data a program $\mathcal{P}$ should be evaluated on. Formally, a VHE scheme consists of the following four algorithms:

- $\mathsf{Setup}(1^\lambda)$ takes as input a security parameter λ. It outputs the public parameters PP and a secret key SK. The public parameters PP defines a message space $\mathcal{M}$ and a set $\mathcal{F}$ of "admissible" functions $f : \mathcal{M}^k \to \mathcal{M}$.
- $\mathsf{Enc}(\mathsf{SK}, \tau, m)$ takes as input a secret key SK, a label $\tau \in \{0,1\}^*$ and a message $m \in \mathcal{M}$. It outputs a ciphertext c.
- $\mathsf{Dec}(\mathsf{SK}, \mathcal{P}, c)$ takes as input a secret key SK, a labeled program $\mathcal{P}$ and a ciphertext c. It outputs a message $m \in \mathcal{M}$ or an error symbol $\perp$.
- $\mathsf{Eval}(\mathsf{PP}, f, \boldsymbol{c})$ takes as input the public parameters PP, a function $f \in \mathcal{F}$ and a vector of ciphertexts $\boldsymbol{c} = (c_1, \ldots, c_k)$. It outputs a new ciphertext c.

For correctness, we require that for each $(\mathsf{PP}, \mathsf{SK})$ output by $\mathsf{Setup}(1^\lambda)$, the following properties hold:

1. For all labels $\tau \in \{0,1\}^*$ and all $m \in \mathcal{M}$, if $c \leftarrow \mathsf{Enc}(\mathsf{SK}, \tau, m)$ then $\mathsf{Dec}(\mathsf{SK}, \mathcal{I}_\tau, c) = m$.
2. Given an admissible function $g : \mathcal{M}^k \leftarrow \mathcal{M}$ and any set of message/program/ciphertext triples $\{(m_i, \mathcal{P}_i, c_i)\}_{i=1}^k$ such that $\mathsf{Dec}(\mathsf{SK}, \mathcal{P}_i, c_i) = m_i$, if $\mathcal{P} = g(\mathcal{P}_1, \ldots, \mathcal{P}_k)$ and $c = \mathsf{Eval}(\mathsf{PP}, g, (c_1, \ldots, c_k))$, then $\mathsf{Dec}(\mathsf{SK}, \mathcal{P}, c) = g(m_1, \ldots, m_k)$.

The above requirements capture the basic correctness of decrypting over freshly encrypted data, as well as the composability of decrypting over the outputs of prior computations. If the set of admissible functions $\mathcal{F}$ of a VHE scheme consists of *linear* functions (resp. *any* functions) from $\mathcal{M}^k$ to $\mathcal{M}$, then we say that the VHE scheme is a *linearly* (resp. *fully*) VHE scheme.

The security requirements of VHE, including *semantic security* and *verifiability*. Informally, *semantic security* requires that a ciphertext c of a message m with label τ should not leak any information about m. Let c_i be a ciphertext on a message m_i with label τ_i for $i = 1, \ldots, k$. *Verifiability* requires that given $(c_1, \ldots, c_k)$, it should be impossible to output a ciphertext c and an admissible function f such that $m' \leftarrow \mathsf{Dec}(\mathsf{SK}, \mathcal{P} = (f, \tau_1, \ldots, \tau_k), c)$ and $m' \neq \{\perp, f(m_1, \ldots, m_k)\}$. The formal definitions of the security requirements of VHE are given in the full version of this paper.

5.1 Generic Construction of VHE from HE and HEA

Given a homomorphic encryption scheme $\mathsf{HE} = (\mathsf{HE.Setup}, \mathsf{HE.Enc}, \mathsf{HE.Dec}, \mathsf{HE.Eval})$ and a homomorphic encrypted authenticator scheme $\mathsf{HEA} = (\mathsf{HEA.Setup}, \mathsf{HEA.Auth}, \mathsf{HEA.Ver}, \mathsf{HEA.Eval})$, we define the 4-tuple algorithms of a VHE scheme $(\mathsf{Setup}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})$ as follows. (Notice that, both public-key homomorphic encryption and secret-key homomorphic encryption can be used in the following generic construction, although the description uses a public-key homomorphic encryption scheme.)

- $\mathsf{Setup}(1^\lambda)$ Given a security parameter λ, the setup algorithm performs $(\mathsf{PK}^{\mathtt{HE}}, \mathsf{SK}^{\mathtt{HE}}) \leftarrow \mathsf{HE.Setup}(1^\lambda)$, $(\mathsf{PK}^{\mathtt{HEA}}, \mathsf{SK}^{\mathtt{HEA}}) \leftarrow \mathsf{HEA.Setup}(1^\lambda)$. Then, it sets the public parameters $\mathsf{PP} = (\mathsf{PK}^{\mathtt{HE}}, \mathsf{PK}^{\mathtt{HEA}})$ and the secret key $\mathsf{SK} = (\mathsf{SK}^{\mathtt{HE}}, \mathsf{SK}^{\mathtt{HEA}})$. We assume that $\mathsf{PK}^{\mathtt{HE}}$ and $\mathsf{PK}^{\mathtt{HEA}}$ are defined over the same message space $\mathcal{M}$ and the same set of admissible functions $\mathcal{F}$.
- $\mathsf{Enc}(\mathsf{SK}, \tau, m)$ Given the secret key, a label $\tau \in \{0,1\}^*$ and a message $m \in \mathcal{M}$, it runs $c^{\mathtt{HE}} \leftarrow \mathsf{HE.Enc}(\mathsf{PK}^{\mathtt{HE}}, m)$, $\sigma^{\mathtt{HEA}} \leftarrow \mathsf{HEA.Auth}(\mathsf{SK}^{\mathtt{HEA}}, \tau, m)$. Then, it outputs the ciphertext $c = (c^{\mathtt{HE}}, \sigma^{\mathtt{HEA}})$.
- $\mathsf{Dec}(\mathsf{SK}, \mathcal{P}, c)$ Given the secret key, a labeled program $\mathcal{P}$ and a ciphertext $c = (c^{\mathtt{HE}}, \sigma^{\mathtt{HEA}})$, it runs $m \leftarrow \mathsf{HE.Dec}(\mathsf{SK}^{\mathtt{HE}}, c^{\mathtt{HE}})$. Then, it checks whether $\mathsf{HEA.Ver}(\mathsf{SK}^{\mathtt{HEA}}, m, \mathcal{P}, \sigma^{\mathtt{HEA}}) = 1$. If so, it outputs the message m; otherwise, it outputs $\perp$.
- $\mathsf{Eval}(\mathsf{PP}, f, \boldsymbol{c})$ With the public parameters $\mathsf{PP} = (\mathsf{PK}^{\mathtt{HE}}, \mathsf{PK}^{\mathtt{HEA}})$, a function $f \in \mathcal{F}$ and a vector of ciphertexts $\boldsymbol{c} = (c_1, \ldots, c_k)$ where $c_i = (c_i^{\mathtt{HE}}, \sigma_i^{\mathtt{HEA}})$ for $i = 1, \ldots, k$, it runs $c^{\mathtt{HE}} \leftarrow \mathsf{HE.Eval}(\mathsf{PK}^{\mathtt{HE}}, f, \boldsymbol{c}^{\mathtt{HE}})$, $\sigma^{\mathtt{HEA}} \leftarrow \mathsf{HEA.Eval}(\mathsf{PK}^{\mathtt{HEA}}, f, \boldsymbol{\sigma}^{\mathtt{HEA}})$, where $\boldsymbol{c}^{\mathtt{HE}} = (c_1^{\mathtt{HE}}, \ldots, c_k^{\mathtt{HE}})$ and $\boldsymbol{\sigma}^{\mathtt{HEA}} = (\sigma_1^{\mathtt{HEA}}, \ldots, \sigma_k^{\mathtt{HEA}})$. Then, it outputs a ciphertext $c = (c^{\mathtt{HE}}, c^{\mathtt{HEA}})$.

Obviously, the above scheme satisfies the correctness of VHE if the underlying homomorphic encryption scheme and homomorphic encrypted authenticator scheme are correct. Now, we state the security theorems of our proposed VHE scheme.

Theorem 3. *If the homomorphic encryption scheme* HE *and homomorphic encrypted authenticator scheme* HEA *are semantically secure, then the proposed VHE scheme is semantically secure.*

Proof. Let $c = (c^{\mathtt{HE}}, \sigma^{\mathtt{HEA}})$ be a ciphertext of message m. Since the underlying homomorphic encryption scheme and homomorphic encrypted authenticator are semantically secure, $c^{\mathtt{HE}}$ and $\sigma^{\mathtt{HEA}}$ do not leak any information about m. Thus, the proposed VHE scheme is also semantically secure.

Theorem 4. *If the homomorphic encrypted authenticator scheme* HEA *is (resp. weakly) unforgeable, then the proposed VHE scheme is (resp. weakly) verifiable.*

Proof. Suppose there exists an adversary $\mathcal{A}$ that breaks the verifiability of the proposed VHE scheme. We can build an algorithm $\mathcal{B}$ that breaks the unforgeability of the underlying homomorphic encrypted authenticator scheme as follows.

Let $\mathcal{C}$ be the challenger corresponding to $\mathcal{B}$ in the unforgeability game of the underlying HEA scheme. $\mathcal{B}$ is given the public key $\mathsf{PK}^{\mathtt{HEA}}$ of the underlying HEA scheme and runs $\mathcal{A}$ executing the following steps.

Setup. $\mathcal{B}$ first runs $(\mathsf{PK}^{\mathtt{HE}}, \mathsf{SK}^{\mathtt{HE}}) \leftarrow \mathsf{HE.Setup}(1^{\lambda})$. Then, it sends the public parameters $\mathsf{PP} = (\mathsf{PK}^{\mathtt{HE}}, \mathsf{PK}^{\mathtt{HEA}})$ to adversary $\mathcal{A}$.

Queries. Since $\mathcal{B}$ knows the secret key $\mathsf{SK}^{\mathtt{HE}}$, with the help of the authentication and verification oracles of HEA provided by $\mathcal{C}$, it can answer $\mathcal{A}$'s ciphertext and verification queries.

Output. Finally, $\mathcal{A}$ outputs a labeled program $\mathcal{P}^*$ and a ciphertext $c^* = (c^{*\mathtt{HE}}, \sigma^{*\mathtt{HEA}})$. $\mathcal{B}$ runs $m^* \leftarrow \mathsf{HE.Dec}(\mathsf{SK}^{\mathtt{HE}}, c^{*\mathtt{HE}})$ and outputs $(m^*, \mathcal{P}^*, \sigma^{*\mathtt{HEA}})$.

Obviously, if $\mathcal{A}$ breaks the verifiability of the proposed VHE scheme with non-negligible advantage, $\mathcal{B}$ will win the unforgeable game of the underlying HEA scheme with non-negligible advantage. This completes the proof of Theorem 4.

Instantiating the above generic construction with existing fully homomorphic encryption (resp. linearly homomorphic encryption) and our proposed fully HEA with weak unforgeability (resp. linearly HEA with unforgeability), it is straightforward to derive a fully VHE with weak verifiability (reps. linearly VHE with verifiability); we omit the details of the constructions here in order to keep the paper compact.

6 Conclusion

In this paper, we study verifiable homomorphic encryption (VHE) which enables verifiable computation on outsourced encrypted data. In order to construct VHE schemes, we introduce a new cryptographic primitive called homomorphic encrypted authenticator (HEA), and show that a VHE scheme can be built upon a homomorphic encryption scheme and a HEA scheme. We observe that the fully homomorphic MAC scheme, proposed by Gennaro and Wichs [13] recently, is a fully HEA scheme in a weaker security model. Then, we present a linearly HEA scheme which is proven secure in a full security model. Instantiating the generic

construction of VHE, we can derive a fully VHE scheme which is secure in a weaker security model, and a linearly VHE scheme which is secure in a full security model. An open research problem is to find fully VHE constructions which is secure in a full security model, i.e., it allows *any* verifiable computation over outsourced encrypted data and tolerates *any* number of malicious verification queries.

Acknowledgement. We are grateful to the anonymous reviewers for their helpful comments. The work of Junzuo Lai was supported by the National Natural Science Foundation of China (Nos. 61300226, 61272534), the Research Fund for the Doctoral Program of Higher Education of China (No. 20134401120017), the Guangdong Provincial Natural Science Foundation (No. S2013040014826), and the Fundamental Research Funds for the Central Universities. The work of Jian Weng was supported by the National Science Foundation of China (Nos. 61272413, 61133014), the Fok Ying Tung Education Foundation (No. 131066), the Program for New Century Excellent Talents in University (No. NCET-12-0680), the Research Fund for the Doctoral Program of Higher Education of China (No. 20134401110011), and the Foundation for Distinguished Young Talents in Higher Education of Guangdong (No. 2012LYM 0027).

References

1. Gentry, C.: Fully homomorphic encryption using ideal lattices. In: STOC, pp. 169–178 (2009)
2. Boneh, D., Freeman, D.M., Katz, J., Waters, B.: Signing a linear subspace: Signature schemes for network coding. In: Jarecki, S., Tsudik, G. (eds.) PKC 2009. LNCS, vol. 5443, pp. 68–87. Springer, Heidelberg (2009)
3. Gennaro, R., Katz, J., Krawczyk, H., Rabin, T.: Secure network coding over the integers. In: Nguyen, P.Q., Pointcheval, D. (eds.) PKC 2010. LNCS, vol. 6056, pp. 142–160. Springer, Heidelberg (2010)
4. Boneh, D., Freeman, D.M.: Linearly homomorphic signatures over binary fields and new tools for lattice-based signatures. In: Catalano, D., Fazio, N., Gennaro, R., Nicolosi, A. (eds.) PKC 2011. LNCS, vol. 6571, pp. 1–16. Springer, Heidelberg (2011)
5. Attrapadung, N., Libert, B.: Homomorphic network coding signatures in the standard model. In: Catalano, D., Fazio, N., Gennaro, R., Nicolosi, A. (eds.) PKC 2011. LNCS, vol. 6571, pp. 17–34. Springer, Heidelberg (2011)
6. Catalano, D., Fiore, D., Warinschi, B.: Adaptive pseudo-free groups and applications. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 207–223. Springer, Heidelberg (2011)
7. Catalano, D., Fiore, D., Warinschi, B.: Efficient network coding signatures in the standard model. In: Fischlin, M., Buchmann, J., Manulis, M. (eds.) PKC 2012. LNCS, vol. 7293, pp. 680–696. Springer, Heidelberg (2012)
8. Freeman, D.M.: Improved security for linearly homomorphic signatures: A generic framework. In: Fischlin, M., Buchmann, J., Manulis, M. (eds.) PKC 2012. LNCS, vol. 7293, pp. 697–714. Springer, Heidelberg (2012)

9. Boneh, D., Freeman, D.M.: Homomorphic signatures for polynomial functions. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 149–168. Springer, Heidelberg (2011)
10. Ahn, J.H., Boneh, D., Camenisch, J., Hohenberger, S., Shelat, A., Waters, B.: Computing on authenticated data. In: Cramer, R. (ed.) TCC 2012. LNCS, vol. 7194, pp. 1–20. Springer, Heidelberg (2012)
11. Attrapadung, N., Libert, B., Peters, T.: Computing on authenticated data: New privacy definitions and constructions. In: Wang, X., Sako, K. (eds.) ASIACRYPT 2012. LNCS, vol. 7658, pp. 367–385. Springer, Heidelberg (2012)
12. Attrapadung, N., Libert, B., Peters, T.: Efficient completely context-hiding quotable and linearly homomorphic signatures. In: Kurosawa, K., Hanaoka, G. (eds.) PKC 2013. LNCS, vol. 7778, pp. 386–404. Springer, Heidelberg (2013)
13. Gennaro, R., Wichs, D.: Fully homomorphic message authenticators. In: Sako, K., Sarkar, P. (eds.) ASIACRYPT 2013, Part II. LNCS, vol. 8270, pp. 301–320. Springer, Heidelberg (2013)
14. Catalano, D., Fiore, D.: Practical homomorphic MACs for arithmetic circuits. In: Johansson, T., Nguyen, P.Q. (eds.) EUROCRYPT 2013. LNCS, vol. 7881, pp. 336–352. Springer, Heidelberg (2013)
15. Backes, M., Fiore, D., Reischuk, R.M.: Verifiable delegation of computation on outsourced data. In: ACM Conference on Computer and Communications Security, pp. 863–874 (2013)
16. Bitansky, N., Canetti, R., Chiesa, A., Tromer, E.: From extractable collision resistance to succinct non-interactive arguments of knowledge, and back again. In: ITCS, pp. 326–349 (2012)
17. Gennaro, R., Gentry, C., Parno, B.: Non-interactive verifiable computing: Outsourcing computation to untrusted workers. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 465–482. Springer, Heidelberg (2010)
18. Chung, K.-M., Kalai, Y., Vadhan, S.: Improved delegation of computation using fully homomorphic encryption. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 483–501. Springer, Heidelberg (2010)
19. Barbosa, M., Farshim, P.: Delegatable homomorphic encryption with applications to secure outsourcing of computation. In: Dunkelman, O. (ed.) CT-RSA 2012. LNCS, vol. 7178, pp. 296–312. Springer, Heidelberg (2012)
20. Parno, B., Raykova, M., Vaikuntanathan, V.: How to delegate and verify in public: Verifiable computation from attribute-based encryption. In: Cramer, R. (ed.) TCC 2012. LNCS, vol. 7194, pp. 422–439. Springer, Heidelberg (2012)
21. Goldwasser, S., Kalai, Y.T., Popa, R.A., Vaikuntanathan, V., Zeldovich, N.: Succinct functional encryption and applications: Reusable garbled circuits and beyond. IACR Cryptology ePrint Archive 2012, 733 (2012)
22. Gennaro, R., Pastro, V.: Verifiable computation over encrypted data in the presence of verification queries. Cryptology ePrint Archive, Report 2014/202 (2014), `http://eprint.iacr.org/`
23. Benabbas, S., Gennaro, R., Vahlis, Y.: Verifiable delegation of computation over large datasets. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 111–131. Springer, Heidelberg (2011)
24. Fiore, D., Gennaro, R.: Publicly verifiable delegation of large polynomials and matrix computations, with applications. In: ACM Conference on Computer and Communications Security, pp. 501–512 (2012)
25. Catalano, D., Fiore, D., Gennaro, R., Vamvourellis, K.: Algebraic (trapdoor) one way functions and their applications. IACR Cryptology ePrint Archive 2012, 434 (2012)

26. Papamanthou, C., Shi, E., Tamassia, R.: Signatures of correct computation. In: Sahai, A. (ed.) TCC 2013. LNCS, vol. 7785, pp. 222–242. Springer, Heidelberg (2013)
27. Papamanthou, C., Tamassia, R., Triandopoulos, N.: Optimal verification of operations on dynamic sets. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 91–110. Springer, Heidelberg (2011)
28. Rivest, R.L., Adleman, L., Dertouzos, M.L.: On data banks and privacy homomorphisms. Foundations of Secure Computation 32(4), 169–178 (1978)
29. Stehlé, D., Steinfeld, R.: Faster fully homomorphic encryption. In: Abe, M. (ed.) ASIACRYPT 2010. LNCS, vol. 6477, pp. 377–394. Springer, Heidelberg (2010)
30. Smart, N.P., Vercauteren, F.: Fully homomorphic encryption with relatively small key and ciphertext sizes. In: Nguyen, P.Q., Pointcheval, D. (eds.) PKC 2010. LNCS, vol. 6056, pp. 420–443. Springer, Heidelberg (2010)
31. Gentry, C., Halevi, S., Smart, N.P.: Fully homomorphic encryption with polylog overhead. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 465–482. Springer, Heidelberg (2012)
32. Gentry, C., Halevi, S., Smart, N.P.: Better bootstrapping in fully homomorphic encryption. In: Fischlin, M., Buchmann, J., Manulis, M. (eds.) PKC 2012. LNCS, vol. 7293, pp. 1–16. Springer, Heidelberg (2012)
33. Brakerski, Z., Gentry, C., Halevi, S.: Packed ciphertexts in LWE-based homomorphic encryption. In: Kurosawa, K., Hanaoka, G. (eds.) PKC 2013. LNCS, vol. 7778, pp. 1–13. Springer, Heidelberg (2013)
34. Cheon, J.H., Coron, J.-S., Kim, J., Lee, M.S., Lepoint, T., Tibouchi, M., Yun, A.: Batch fully homomorphic encryption over the integers. In: Johansson, T., Nguyen, P.Q. (eds.) EUROCRYPT 2013. LNCS, vol. 7881, pp. 315–335. Springer, Heidelberg (2013)
35. Gentry, C., Sahai, A., Waters, B.: Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part I. LNCS, vol. 8042, pp. 75–92. Springer, Heidelberg (2013)
36. van Dijk, M., Gentry, C., Halevi, S., Vaikuntanathan, V.: Fully homomorphic encryption over the integers. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 24–43. Springer, Heidelberg (2010)
37. Brakerski, Z., Vaikuntanathan, V.: Efficient fully homomorphic encryption from (standard) lwe. In: FOCS, pp. 97–106 (2011)
38. Brakerski, Z., Vaikuntanathan, V.: Fully homomorphic encryption from ring-LWE and security for key dependent messages. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 505–524. Springer, Heidelberg (2011)
39. Brakerski, Z., Gentry, C., Vaikuntanathan, V.: (leveled) fully homomorphic encryption without bootstrapping. In: ITCS, pp. 309–325 (2012)
40. Johnson, R., Molnar, D., Song, D., Wagner, D.: Homomorphic signature schemes. In: Preneel, B. (ed.) CT-RSA 2002. LNCS, vol. 2271, pp. 244–262. Springer, Heidelberg (2002)
41. Ahlswede, R., Cai, N., Li, S.Y.R., Yeung, R.W.: Network information flow. IEEE Transactions on Information Theory 46(4), 1204–1216 (2000)
42. Li, S.Y.R., Yeung, R.W., Cai, N.: Linear network coding. IEEE Transactions on Information Theory 49(2), 371–381 (2003)
43. Ateniese, G., Burns, R.C., Curtmola, R., Herring, J., Kissner, L., Peterson, Z.N.J., Song, D.X.: Provable data possession at untrusted stores. In: ACM Conference on Computer and Communications Security, pp. 598–609 (2007)

44. Juels, A., Kaliski Jr., B.S.: Pors: proofs of retrievability for large files. In: ACM Conference on Computer and Communications Security, pp. 584–597 (2007)
45. Shacham, H., Waters, B.: Compact proofs of retrievability. In: Pieprzyk, J. (ed.) ASIACRYPT 2008. LNCS, vol. 5350, pp. 90–107. Springer, Heidelberg (2008)
46. Groth, J.: Simulation-sound NIZK proofs for a practical language and constant size group signatures. In: Lai, X., Chen, K. (eds.) ASIACRYPT 2006. LNCS, vol. 4284, pp. 444–459. Springer, Heidelberg (2006)
47. Abe, M., Fuchsbauer, G., Groth, J., Haralambiev, K., Ohkubo, M.: Structure-preserving signatures and commitments to group elements. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 209–236. Springer, Heidelberg (2010)
48. Libert, B., Peters, T., Joye, M., Yung, M.: Linearly homomorphic structure-preserving signatures and their applications. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part II. LNCS, vol. 8043, pp. 289–307. Springer, Heidelberg (2013)
49. Catalano, D., Marcedone, A., Puglisi, O.: Linearly homomorphic structure preserving signatures: New methodologies and applications. IACR Cryptology ePrint Archive 2013, 801 (2013)
50. Joo, C., Yun, A.: Homomorphic authenticated encryption secure against chosen-ciphertext attack. IACR Cryptology ePrint Archive 2013, 726 (2013)
51. Boneh, D., Boyen, X., Shacham, H.: Short group signatures. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 41–55. Springer, Heidelberg (2004)
52. Boneh, D., Boyen, X.: Short signatures without random oracles and the sdh assumption in bilinear groups. J. Cryptology 21(2), 149–177 (2008)