

Singapore Management University Institutional Knowledge at Singapore Management University

Research Collection School Of Information Systems

School of Information Systems

5-2016

Efficient verifiable computation of linear and quadratic functions over encrypted data

Ngoc Hieu TRAN

Singapore Management University, nhtran.2013@phdis.smu.edu.sg

Hwee Hwa PANG


Singapore Management University, hhpang@smu.edu.sg

Robert H. DENG

Singapore Management University, robertdeng@smu.edu.sg

DOI: <https://doi.org/10.1145/2897845.2897892>

Follow this and additional works at: https://ink.library.smu.edu.sg/sis_research

 Part of the [Databases and Information Systems Commons](#), and the [Information Security Commons](#)

Citation

TRAN, Ngoc Hieu; Hwee Hwa PANG; and DENG, Robert H.. Efficient verifiable computation of linear and quadratic functions over encrypted data. (2016). *Asia CCS '16: Proceedings of the 11th ACM Asia Conference on Computer and Communications Security, Xi'an, China, May 30 - June 3*. 605-616. Research Collection School Of Information Systems.

Available at: https://ink.library.smu.edu.sg/sis_research/3351

This Conference Proceeding Article is brought to you for free and open access by the School of Information Systems at Institutional Knowledge at Singapore Management University. It has been accepted for inclusion in Research Collection School Of Information Systems by an authorized administrator of Institutional Knowledge at Singapore Management University. For more information, please email libIR@smu.edu.sg.

Efficient Verifiable Computation of Linear and Quadratic Functions over Encrypted Data

Ngoc Hieu Tran
School of Information Systems
Singapore Management
University
nhtran.2013@smu.edu.sg

HweeHwa Pang
School of Information Systems
Singapore Management
University
hhpang@smu.edu.sg

Robert H. Deng
School of Information Systems
Singapore Management
University
robertdeng@smu.edu.sg

ABSTRACT

In data outsourcing, a client stores a large amount of data on an untrusted server; subsequently, the client can request the server to compute a function on any subset of the data. This setting naturally leads to two security requirements: *confidentiality* of input data, and *authenticity* of computations. Existing approaches that satisfy both requirements simultaneously are built on fully homomorphic encryption, which involves expensive computation on the server and client and hence is impractical. In this paper, we propose two verifiable homomorphic encryption schemes that do not rely on fully homomorphic encryption. The first is a simple and efficient scheme for linear functions. The second scheme supports the class of multivariate quadratic functions, by combining the Paillier cryptosystem with a new homomorphic message authentication code (MAC) scheme. Through formal security analysis, we show that the schemes are semantically secure and unforgeable.

Keywords

Verifiable computation; homomorphic encryption; homomorphic MAC; data outsourcing

1. INTRODUCTION

In a cloud scenario, a client with limited resources outsources a large amount of data to a powerful server, and delegates to the server the role of performing computation on the data. If the server is untrusted, the client must be able to verify the correctness of computations returned by the server. Homomorphic message authenticator schemes provide this capability. Following Gennaro et al.'s work in [23], many such protocols have been proposed for both private verifiable settings (i.e., homomorphic MAC) [12, 4, 7, 14, 15] and public verifiable settings (i.e., homomorphic signature) [9, 20, 31, 32, 18]. However, homomorphic message authenticator by itself does not maintain the confidentiality of outsourced data. In particular, all the protocols mentioned above store data on the server in plaintext format.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

ASIA CCS '16, May 30-June 03, 2016, Xi'an, China

© 2016 ACM. ISBN 978-1-4503-4233-9/16/05...\$15.00

DOI: <http://dx.doi.org/10.1145/2897845.2897892>

If the client data is sensitive, a natural concern is how to ensure simultaneously the confidentiality of the outsourced data and the authenticity of the server's computations. We want to enable the client to encrypt the data before releasing them to the server, and for the latter to perform computation directly on the encrypted data. One approach is to combine data encryption and homomorphic message authenticator (either homomorphic signature or homomorphic MAC) by the Encrypt-and-MAC composition [6]. This approach, known as *verifiable homomorphic encryption*, was first introduced by Lai et al. [27]. It is similar to the homomorphic authenticator encryption notions of Gennaro and Wichs [24] as well as Joo and Yun [26].

While several solutions exist for verifiable computation of linear functions in private settings [1] and public settings [10, 16, 3, 17, 22], there are only a few recent studies on verifiable computation on encrypted data. In particular, the scheme of Libert et al. [29] works only for very small data domains because the client must perform a discrete logarithm operation to recover the computed result, while the scheme of Lai et al. [27] relies on fully homomorphic encryption (FHE). Both of the schemes require the client to perform expensive computations and are inefficient in practice. In another line of work, several studies consider how to support a larger class of computation, such as multivariate quadratic polynomials or univariate polynomials of high degree [26, 21, 5]. These schemes guarantee confidentiality and authenticity simultaneously, but they require fully homomorphic encryption. To the best of our knowledge, there is no scheme in the literature for verifiable computation of multivariate quadratic polynomials on encrypted data, without using FHE¹.

1.1 Our Contribution

In this work, we propose two schemes for verifiable computation on outsourced encrypted data. First, we introduce a very efficient scheme for linear functions. Our second scheme for multivariate quadratic functions only needs an additively homomorphic encryption technique such as the Paillier cryptosystem [30], rather than FHE. The details of our contributions are as follows.

Linear functions. We construct a practical verifiable homomorphic encryption scheme for linear functions that is provably secure under the pseudo-random function assumption. The scheme employs symmetric key homomorphic

¹The recent work reported in [2] proposes a verifiable computation scheme that does not make use of FHE. However, it assumes a model in which the client delegates its computation to multiple servers, at least one of which is honest.

encryption, based on one-time pad, to protect the confidentiality of input data, and combines it with the practical homomorphic authenticator of [12] to verify the correctness of computation. Compared to previous work, our scheme has the advantages of being simple and efficient.

Multivariate quadratic functions. We also propose a verifiable homomorphic encryption scheme for multivariate quadratic functions. In order to achieve the twin requirements of confidentiality and authenticity, we construct a new, efficient scheme that combines the Paillier cryptosystem and a homomorphic MAC, following the *Encrypt-and-MAC* composition [6].

From the technique of Backes et al. in [4] for constructing homomorphic MAC for quadratic functions, we observe that the Paillier cryptosystem can be used in place of an FHE scheme in computing quadratic polynomials. This idea is generalized in [13]. However, we emphasize that our scheme is the first that guarantees the confidentiality of input data and authenticity of outsourced computations on multivariate quadratic functions without using FHE.

To satisfy the authenticity requirement, existing homomorphic MAC schemes such as [12, 4] are not applicable in our construction since an authenticator leaks information about the message that it authenticates. Thus, we formulate a new homomorphic MAC scheme for multivariate quadratic functions. Our homomorphic MAC scheme differs from the one in [4], which is based on bilinear maps, in two ways: our authenticator does not leak any information about the message that it authenticates, and our scheme is more efficient as it does not require bilinear maps.

Although our scheme supports a limited class of computations (i.e., linear and multivariate quadratic functions), it is versatile enough to cover a wide range of statistical computations including sum, average, variance, standard deviation, root mean square, covariance, and linear regression [21].

1.2 Organization

The rest of this paper is organized as follows. Section 2 surveys related work. Section 3 gives some preliminary information and our generic framework. Sections 4 and 5 present our verifiable homomorphic encryption schemes for linear functions and quadratic functions, respectively. Section 6 then reports on an empirical evaluation of our schemes. Finally, Section 7 concludes the paper.

2. RELATED WORK

We review prior work on non-interactive verifiable computation, verifiable homomorphic encryption for linear functions, and verifiable homomorphic encryption for multivariate quadratic functions.

Non-interactive Verifiable Computation. This research direction was initiated by Gennaro et al. [23]. Non-interactive verifiable computation allows a client with limited resources to outsource a set of data, and the computation of functions on the data, to a server. The server returns the result of the function evaluation, along with a non-interactive proof that the result was carried out correctly.

In [12], Catalano and Fiore proposed a homomorphic MAC that supports polynomial functions computed from arithmetic circuits. In their construction, the authenticator of a message is a set of polynomial coefficients that evaluates to the message on a public point (e.g., 0) and a randomly chosen

value on a secret point (e.g., secret key). Their construction is efficient, and can tolerate any number of verification queries. Based on the scheme of Catalano and Fiore [12], Backes et al. proposed a practical protocol in [4] for evaluating degree-2 polynomial functions. They introduced a homomorphic MAC scheme with *efficient verification*, meaning that a client can verify the correctness of computation in constant time, independently of the number of inputs. This is helpful when the client frequently needs to compute over the same data set. Both the schemes in [12] and [4] do not protect data confidentiality, because the authenticator leaks information about the message that it authenticates.

Linear verifiable homomorphic encryption. To defend against pollution attack in network coding [28], Agrawal et al. [1] proposed linear homomorphic message authentication code in a private setting, while Boneh et al. [8] introduced linear homomorphic signature in a public setting. These studies do not consider the confidentiality of input data.

Recently, Libert et al. [29] introduced the notion of linearly homomorphic structure-preserving signatures. Their construction allows a client to apply additively homomorphic encryption on data to be stored on an untrusted remote server, and the client is able to verify linear combinations that the server performs over the encrypted data. This scheme has a limitation in that the client must carry out a discrete log operation to recover the plaintext from the encrypted data. Thus, the construction only works for very small data domains. In contrast, our scheme for linear functions supports large data domains. In [19], Catalano et al. introduced the notion of linear homomorphic authenticated encryption with public verifiability, and proposed a linear homomorphic signature scheme for outsourced encrypted data based on the Paillier cryptosystem. Different from that work, our scheme is constructed for linear functions in a private verifiable setting, and combines homomorphic MAC [12] and symmetric key homomorphic encryption to achieve a highly practical solution.

Verifiable homomorphic encryption for quadratic functions. In [24], Gennaro and Wichs introduced a fully homomorphic MAC scheme and extends it to verifiable homomorphic encryption. Their construction relies on FHE to preserve the confidentiality of input data and authenticity of computations. The scheme only satisfies the weak security model where an adversary is not allowed to issue verification queries. In [26], Joo et al. introduced a homomorphic authenticated encryption scheme for low-degree polynomials. Their construction relies on the (error-free) approximate-GCD assumption and the homomorphic message authentication code proposed by Catalano et al. [12].

Goldwasser [25] presented a generic protocol for achieving privacy of input data and verification of outsourced computation. Their main result is the construction of a succinct single-key functional encryption scheme for general functions. The function encryption in this scheme is constructed from FHE and attribute-based encryption. However, their scheme only works for binary functions, which is not efficient in practice. In addition, there is no formal definition and security proof [21].

In [21], Fiore et al. proposed verifiable homomorphic encryption schemes for linear functions, multivariate quadratic functions, and univariate polynomials of high degree. There, a data item is encrypted by the BGV homomorphic encryption

tion [11], and a result returned by the server is verified through the homomorphic MAC scheme of [4]. This work introduced a novel homomorphic hashing technique for compressing ciphertext. That helps the client to save storage cost and reduce computation cost on the outsourced server.

All the schemes above are built on FHE. The advantage of our work is that it requires only additively homomorphic encryption (i.e., Paillier cryptosystem) to achieve an efficient scheme supporting multivariate quadratic functions.

3. GENERIC FRAMEWORK

We begin by reviewing some notations and definitions that we use in this work. We denote $x \xleftarrow{\$} S$ as the operation of assigning to x an element selected uniformly at random from a set S . The notation $x \leftarrow A(\cdot)$ denotes the operation of running a procedure A with the given input and assigning the output to x .

3.1 Arithmetic Circuits

This section gives an overview of arithmetic circuits. (For more details, we refer the interested reader to [33]). An arithmetic circuit over a field \mathbb{F} is a directed acyclic graph. Each node in the graph is called a *gate*. A gate with in-degree 0 is an *input gate*. An input gate is labeled by either a variable from a set of variables $\mathcal{X} = (x_1, \dots, x_n) \in \mathbb{F}^n$ or a constant $\alpha \in \mathbb{F}$. A gate with in-degree and out-degree greater than 0 is an *internal gate*. An internal gate is either an addition gate with a $+$ label, or a multiplication gate with a \times label. A gate with out-degree 0 is an *output gate*. In this work, we allow a variable to undergo only one multiplication with another variable, but unlimited multiplications with constants and additions with other variables. Such a circuit can always be transformed into one in which every internal gate has two inputs. Therefore, we consider only circuits with one output gate, and in which each internal gate has in-degree 2. The final result of the circuit is the output of the output gate.

In the following, we define a ‘*level*’ for each gate, which will be used in our proposed scheme in Section 5. An input gate that is a variable has a level of ‘1’, whereas an input gate that is a constant has a level of ‘0’. The level of a multiplication gate is the sum of the levels of its two inputs. Moreover, the level of an addition gate is the maximum of the levels of its two inputs. The level of a circuit is the level of its output gate. In the scheme in Section 5, the level of the output gate is at most ‘2’.

3.2 Labeled Programs

In a cloud scenario, assume that a data owner wants to share the outsourced data with many other clients, and the clients are able to request different computations on various data items. We use the notions of labeled data and labeled program \mathcal{P} to indicate which data is being authenticated and how the data should be evaluated. For example, in a collection of files, the label of a data item may be a file name, or the date and time at which the file is created (e.g., “2015/10/04 - 00:00:00”); a labeled program may be the computation of the *median* of the values in all the files recorded in April. With structured data such as a database table, a labeled datum may comprise the identifier of a record and the name of an attribute, e.g., (“id”, “salary”), while a labeled program may compute the *sum* of the “salary” in all records with “id” in the range from 1 to 100.

We adopt the formal notion of labeled program introduced by Gennaro and Wichs in [24]. A labeled program \mathcal{P} is defined by a tuple $(f, \tau_1, \dots, \tau_n)$ where $f : \mathbb{F}^n \rightarrow \mathbb{F}$ is a circuit as defined above, and each label $\tau_i \in \{0, 1\}^*$ uniquely identifies the i -th input node of f . Labeled programs may be composed as follows. Given labeled programs $\mathcal{P}_1, \dots, \mathcal{P}_t$ and a circuit $f : \mathbb{F}^t \rightarrow \mathbb{F}$, the composed program $\mathcal{P}^* = f(\mathcal{P}_1, \dots, \mathcal{P}_t)$ evaluates a circuit f on the outputs of $\mathcal{P}_1, \dots, \mathcal{P}_t$. The labeled inputs of \mathcal{P}^* correspond to the distinct labeled inputs of $\mathcal{P}_1, \dots, \mathcal{P}_t$. We denote by $\mathcal{I}_\tau = (f_{id}, \tau)$ the identity program with input label $\tau \in \{0, 1\}^*$, where $f_{id} : \mathbb{F} \rightarrow \mathbb{F}$ is the canonical identity function. Notice that any program $\mathcal{P} = (f, \tau_1, \dots, \tau_n)$ can be written as a composition of identity programs $\mathcal{P} = f(\mathcal{I}_{\tau_1}, \dots, \mathcal{I}_{\tau_n})$.

3.3 Paillier Cryptosystem

Key Generation. Let p_1, p_2 be two large, independent prime numbers. Set $N = p_1 p_2$, and $\lambda' = \text{lcm}(p_1 - 1, p_2 - 1)$ where lcm represents the least common multiple function. Next, set function $L(x) = (x - 1)/N$ and select a random number $g' \xleftarrow{\$} \mathbb{Z}_{N^2}^*$ such that the order of g' is a non-zero multiple of N (e.g., $g' = 1 + N$ satisfies the condition and is easily calculated). Output (N, g') as the public key and (p_1, p_2) as the private key.

Encryption. Let $m \in \mathbb{Z}_N$ be a datum to be encrypted. Select a random number $u \xleftarrow{\$} \mathbb{Z}_N^*$, then encrypt m as $c = E(m) = g'^m \cdot u^N \bmod N^2$.

Decryption. Let $c \in \mathbb{Z}_{N^2}^*$ be a ciphertext to be decrypted. Compute $m = D(c) = \frac{L(c^{\lambda'} \bmod N^2)}{L(g'^{\lambda'} \bmod N^2)} \bmod N$.

3.4 Generic Verifiable Homomorphic Encryption Scheme

We now define our verifiable homomorphic encryption scheme for outsourced computation over encrypted data. Our scheme uses homomorphic encryption (HE) to protect the confidentiality of data, and homomorphic message authentication code (MAC) simultaneously to verify the correctness of outsourced computations. We denote our verifiable homomorphic encryption scheme as HEMAC. HEMAC works in a cloud scenario where the data owner runs a one-time key generation and encryption procedure to outsource a set of encrypted data to a server. Subsequently, a client who possesses a shared secret key with the data owner may request the server to perform computation over any subset of data (by specifying the set of labeled data and the labeled program), and receive the encrypted form of the result. The client can then decrypt and verify the correctness of the result returned by the server.

The generic HEMAC scheme is a tuple of probabilistic polynomial time algorithms defined as follows:

- **KeyGen**(1^λ) takes as input a security parameter λ , and outputs a secret key SK and a public parameter PP .
- **Enc**(SK, τ, m) takes as input a secret key SK , a label $\tau \in \{0, 1\}^*$ and a datum $m \in \mathcal{M}$. It outputs a ciphertext C .
- **Eval**($\text{PP}, \mathcal{P}, \vec{C}$) takes as input the public parameter PP , a labeled program $\mathcal{P} = (f, \tau_1, \dots, \tau_n)$ and a vector of ciphertexts $\vec{C} = (C_1, \dots, C_n)$. It outputs a new ciphertext C .

- $\text{Dec}(\text{SK}, \mathcal{P}, C)$ takes as input the secret key SK , a labeled program $\mathcal{P} = (f, \tau_1, \dots, \tau_n)$ and a ciphertext C . It outputs a datum $m \in \mathcal{M}$ or an error symbol \perp .

3.5 Correctness

We require HEMAC to satisfy *encryption correctness* and *evaluation correctness*. For a pair of secret key and public parameter (SK, PP) produced by the KeyGen procedure, the correctness properties are defined as follows.

Encryption correctness. For any datum $m \in \mathcal{M}$ and any label $\tau \in \{0, 1\}^*$, if $C \leftarrow \text{Enc}(\text{SK}, \tau, m)$, then $m \leftarrow \text{Dec}(\text{SK}, \mathcal{I}_\tau, C)$.

Informally, given any C computed by $\text{Enc}(\text{SK}, \tau, m)$, C must be the ciphertext of m with respect to the identity program $\mathcal{P}_{\mathcal{I}_\tau}$.

Evaluation correctness. Given a fixed pair of (SK, PP) , a circuit $f : \mathbb{F}^t \rightarrow \mathbb{F}$, and any set of data/program/ciphertext triples $\{(m_i, \mathcal{P}_i, C_i)\}_{i=1}^t$ such that $m_i \leftarrow \text{Dec}(\text{SK}, \mathcal{P}_i, C_i)$. If $\mathcal{P}^* = f(\mathcal{P}_1, \dots, \mathcal{P}_t)$, $m^* = f(m_1, \dots, m_t)$, and $C^* \leftarrow \text{Eval}(\text{PP}, \mathcal{P}^*, (C_1, \dots, C_t))$, then $m^* \leftarrow \text{Dec}(\text{SK}, \mathcal{P}^*, C^*)$.

Informally, for a given vector of ciphertexts $\vec{C} = (C_1, \dots, C_t)$ where each C_i is the ciphertext of some datum m_i as output by labeled program \mathcal{P}_i , a ciphertext C^* computed by $\text{Eval}(\text{PP}, \mathcal{P}^*, \vec{C})$ with a composed program $\mathcal{P}^* = f(\mathcal{P}_1, \dots, \mathcal{P}_t)$ must be the ciphertext of $m^* = f(m_1, \dots, m_t)$.

3.6 Security Models

The HEMAC scheme has to satisfy the requirements of *semantic security* and *unforgeability*. Let $\mathcal{H} = (\text{KeyGen}, \text{Enc}, \text{Eval}, \text{Dec})$ denote the generic scheme as defined above.

Attack 1. The semantic security of HEMAC is defined by the following security game between a challenger and an adversary \mathcal{A} .

Setup. The challenger runs $\text{KeyGen}(1^\lambda)$ to obtain a pair of secret key and public parameter (SK, PP) . It gives public parameter PP to adversary \mathcal{A} , and keeps secret key SK to itself. Next, the challenger initializes a list $T = \emptyset$ for tracking the queries from \mathcal{A} .

Queries. Adversary \mathcal{A} adaptively issues encryption queries to the challenger, each of the form (τ, m) where $\tau \in \{0, 1\}^*$ and $m \in \mathcal{M}$. The challenger then performs the following:

- If τ does not exist in T , the challenger computes $C \leftarrow \text{Enc}(\text{SK}, \tau, m)$, updates the list $T = T \cup \{\tau\}$, and gives ciphertext C to \mathcal{A} .
- If τ is found in T , the challenger rejects the query.

Challenge. Adversary \mathcal{A} submits a label $\tau^* \in \{0, 1\}^*$ and two data items $m_0, m_1 \in \mathcal{M}$, such that τ^* is not already in list T . The challenger selects a random bit $b \in \{0, 1\}$, computes $C^* \leftarrow \text{Enc}(\text{SK}, \tau^*, m_b)$, and sends C^* to \mathcal{A} .

Output. Adversary \mathcal{A} outputs b' representing its guess for b . \mathcal{A} wins the game if $b' = b$.

The advantage $\text{ss-adv}[\mathcal{A}, \mathcal{H}]$ of adversary \mathcal{A} with respect to the HEMAC scheme in this game is defined as

$$\left| \Pr[b' = b] - \frac{1}{2} \right|,$$

where the probability is taken over the random bits used by the challenger and adversary \mathcal{A} .

Definition 1. The HEMAC scheme \mathcal{H} is semantically secure if, for all probabilistic polynomial time adversary \mathcal{A} , the advantage $\text{ss-adv}[\mathcal{A}, \mathcal{H}]$ is negligible.

Attack 2. The unforgeability of the HEMAC scheme is defined by the following game between a challenger and an adversary \mathcal{A} .

Setup. The challenger runs $\text{KeyGen}(1^\lambda)$ to obtain a pair of secret key and public parameter (SK, PP) . It gives public parameter PP to adversary \mathcal{A} , and keeps secret key SK to itself. The challenger also initializes a list $T = \emptyset$ for tracking the queries from \mathcal{A} .

Ciphertext Queries. Adversary \mathcal{A} adaptively queries for ciphertexts on pairs of label and datum of its choice. Given a query (τ, m) where $\tau \in \{0, 1\}^*$ and $m \in \mathcal{M}$, the challenger performs the following:

- If τ does not exist in T , the challenger computes $C \leftarrow \text{Enc}(\text{SK}, \tau, m)$, updates the list $T = T \cup \{\tau\}$, and gives ciphertext C to \mathcal{A} .
- If τ is found in T , the challenger rejects the query.

Verification queries. Adversary \mathcal{A} adaptively issues verification queries as follows: Given a query (\mathcal{P}, C) , the challenger responds with the output of $\text{Dec}(\text{SK}, \mathcal{P}, C)$.

Output. Adversary \mathcal{A} outputs a forgery tuple of ciphertext C^* and labeled program $\mathcal{P} = (f^*, \tau_1^*, \dots, \tau_n^*)$. The adversary wins the game if $m^* \leftarrow \text{Dec}(\text{SK}, \mathcal{P}, C^*)$ and one of the following conditions holds:

- Type 1 forgery: There exists some $i \in \{1, \dots, n\}$ such that $\tau_i^* \notin T$, i.e., as least one label τ_i^* has not been queried during the game.
- Type 2 forgery: List T contains all the labels $\tau_1^*, \dots, \tau_n^*$ for data items m'_1, \dots, m'_n , and $m^* \neq f^*(m'_1, \dots, m'_n)$, i.e., m^* is not the correct output of program \mathcal{P} when executed on (m'_1, \dots, m'_n) .

The advantage $\text{uf-cmva}[\mathcal{A}, \mathcal{H}]$ of adversary \mathcal{A} with respect to the HEMAC scheme in this game is defined as the probability that \mathcal{A} wins Attack 2.

Definition 2. The HEMAC scheme is existentially unforgeable under adaptive chosen message and query verification attack if, for all probabilistic polynomial time adversaries \mathcal{A} , the advantage $\text{uf-cmva}[\mathcal{A}, \mathcal{H}]$ is negligible.

We note that the adversary can pose a verification query on the tuple $(\mathcal{P} = (f^*, \tau_1^*, \dots, \tau_n^*), C^*)$. The adversary can also terminate the **Verification Queries** phase if the response by the challenger is not \perp and any of the two types of forgery happens.

4. VERIFIABLE HOMOMORPHIC ENCRYPTION FOR LINEAR FUNCTIONS

In this section, we introduce a verifiable homomorphic encryption scheme for linear functions, denoted as l -HEMAC. The construction combines additive homomorphic encryption with homomorphic MAC. In particular, we use symmetric key homomorphic encryption, based on one-time pad, to protect the confidentiality of data. The security of l -HEMAC relies only on the assumption of pseudo-random function.

Let $f : \mathbb{F}^n \rightarrow \mathbb{F}$ be an arithmetic circuit composed by multiplication gates where one of the inputs is a constant, and addition gates. Without loss of generality, we define circuit $f(x_1, \dots, x_n) = \sum_{i=1}^n \alpha_i x_i + \alpha$ for some constants $\alpha_i, \alpha \in \mathbb{F}$ and x_1, \dots, x_n can take arbitrary values in \mathbb{F} .

4.1 Construction

KeyGen(1^λ). Let p be a prime number of roughly λ bits. Choose a random seed $K \xleftarrow{\$} \mathbb{Z}_p$ for pseudo-random function $F_K : \{0, 1\}^* \rightarrow \mathbb{Z}_p$, and a random number $s \xleftarrow{\$} \mathbb{Z}_p$. Output the secret key $\text{SK} = (K, s)$ and public parameter $\text{PP} = p$. Let the data space $\mathcal{M} = \mathbb{Z}_p$ and $\mathbb{F} = \mathbb{Z}_p$.

Enc(SK, τ, m). Given secret key $\text{SK} = (K, s)$, proceed as follows to encrypt a datum $m \in \mathbb{Z}_p$ with label $\tau \in \{0, 1\}^*$. First, compute pseudo-random value $r_\tau = F_K(\tau\|1)$ and one-time encryption key $k_\tau = F_K(\tau\|2)$, where $\|$ denotes concatenation. Then, encrypt $c = m - k_\tau \bmod p$ and output a ciphertext $C = (y_0, y_1)$ where

$$y_0 = c, \quad y_1 = \frac{r_\tau - c}{s} \bmod p.$$

(y_0, y_1) are coefficients of a degree-1 polynomial $y(x) = y_0 + y_1x \in \mathbb{Z}_p[x]$ with variable x . This polynomial has the special property that it evaluates to r_τ on secret key s , i.e., $y(s) = r_\tau$, and it evaluates to ciphertext c on public value 0, i.e., $y(0) = c$.

Eval($\text{PP}, \mathcal{P}, \vec{C}$). Given public parameter $\text{PP} = p$, a program $\mathcal{P} = (f, \tau_1, \dots, \tau_n)$ and a vector of ciphertexts $\vec{C} = (C_1, \dots, C_n)$, the procedure goes through a series of gates in a new circuit $\tilde{f} : (\mathbb{Z}_p^2)^n \rightarrow \mathbb{Z}_p^2$ which has the same structure as f , except that each input is a set of ciphertexts $C = (y_0, y_1)$. At each internal gate **GATE** (i.e., addition gate '+' or multiplication by a constant gate ' \times_c '), given public parameter $\text{PP} = p$ and ciphertexts (C_1, C_2) as the two input gates of **GATE**, a **GateEval** procedure (to be defined shortly) outputs a new ciphertext C which serves as input to the **GateEval** of the next internal gate. The output of the **Eval** procedure is the output of **GateEval** at the output gate of the circuit. We now describe **GateEval**.

GateEval($\text{PP}, \text{GATE}, C_1, C_2$). Let $C_i = (y_0^{(i)}, y_1^{(i)}) \in \mathbb{Z}_p^2$ for $i = 1, 2$.

- **GATE₊**: Compute $y_0 = y_0^{(1)} + y_0^{(2)}$ and $y_1 = y_1^{(1)} + y_1^{(2)}$. For correctness, we observe that (y_0, y_1) are the coefficients of a polynomial representing the sum of the two polynomials defined by $(y_0^{(1)}, y_1^{(1)})$ and $(y_0^{(2)}, y_1^{(2)})$.
- **GATE _{\times_c}** : Here, one of the two inputs is a constant $\alpha \in \mathbb{Z}_p$. Assuming that $C_2 = \alpha$, compute $y_0 = \alpha y_0^{(1)}$ and $y_1 = \alpha y_1^{(1)}$. The correctness of the computation stems from the fact that (y_0, y_1) are the coefficients of a polynomial representing the product of a polynomial defined by $(y_0^{(1)}, y_1^{(1)})$ and constant α .

Finally, return $C = (y_0, y_1)$.

Dec($\text{SK}, \mathcal{P}, C$). Given secret key $\text{SK} = (K, s)$, proceed as follows to decrypt ciphertext $C = (y_0, y_1)$ with labeled program $\mathcal{P} = (f, \tau_1, \dots, \tau_n)$.

- For every input label $\tau_i \in \{0, 1\}^*$, compute pseudo-random value $r_{\tau_i} = F_K(\tau_i\|1)$ and one-time encryption key $k_{\tau_i} = F_K(\tau_i\|2)$.
- To decrypt $m = f(m_1, \dots, m_n)$, compute the decryption key $\kappa = f(k_{\tau_1}, \dots, k_{\tau_n})$ and recover m from y_0 :

$$m = f(m_1, \dots, m_n) = y_0 + \kappa \bmod p.$$

- To verify the correctness of the computation, evaluate the circuit f on $r_{\tau_1}, \dots, r_{\tau_n}$ by computing $R = f(r_{\tau_1}, \dots, r_{\tau_n})$, then use secret key s to check whether the following condition holds:

$$R = y_0 + y_1 s \bmod p. \quad (1)$$

If equation (1) is true, output m ; otherwise, output \perp .

4.2 Correctness

We show that l -HEMAC scheme satisfies the correctness criteria.

Encryption correctness. Let $m \in \mathbb{Z}_p$ be a datum, $\tau \in \{0, 1\}^*$ be the label of m , and $\text{SK} = (K, s)$ be the secret key. Suppose that $C = (y_0, y_1) \in \mathbb{Z}_p^2$ is a ciphertext obtained from running **Enc**(SK, τ, m). We show that $m \leftarrow \text{Dec}(\text{SK}, \mathcal{I}_\tau, C)$ for identity program \mathcal{I}_τ .

Following the **Enc** procedure, we know that:

$$y_0 = c, y_1 = \frac{r_\tau - c}{s} \bmod p, r_\tau = F_K(\tau\|1), k_\tau = F_K(\tau\|2)$$

We verify equation (1) as follows:

$$y_0 + y_1 s = c + \frac{r_\tau - c}{s} \cdot s \bmod p = r_\tau.$$

From ciphertext c and decryption key k_τ , we decrypt m as $m = c + k_\tau \bmod p$. It follows that $m \leftarrow \text{Dec}(\text{SK}, \mathcal{I}_\tau, C)$.

Evaluation correctness. Given a pair of secret key and public parameter (SK, PP) , a circuit $f : \mathbb{Z}_p^t \rightarrow \mathbb{Z}_p$, and a set of data/program/ciphertext triples $\{m_i, \mathcal{P}_i, C_i\}_{i=1}^t$ such that m_i is the output of $\text{Dec}(\text{SK}, \mathcal{P}_i, C_i)$. We assume that each program \mathcal{P}_i for $i = 1, \dots, t$ is defined by a circuit f_i and a set of input labels $(\tau_{i,1}, \dots, \tau_{i,n_i})$, and κ_i is the encryption key used to compute ciphertext C_i . Let R^* be the output circuit $f(R_1, \dots, R_t)$ where $R_i = f_i(r_{\tau_{i,1}}, \dots, r_{\tau_{i,n_i}})$. Let m^* be the output of circuit $f(m_1, \dots, m_t)$, \mathcal{P}^* be the composed labeled program $f(\mathcal{P}_1, \dots, \mathcal{P}_t)$, and $C^* = (y_0^*, y_1^*)$ be the output of **Eval**($\text{PP}, \mathcal{P}^*, (C_1, \dots, C_t)$). We show that m^* is the output of $\text{Dec}(\text{SK}, \mathcal{P}^*, C^*)$.

Claim 1. Let $\mathcal{Q}_{\text{poly}} = \mathbb{Z}_p[x]$ be the ring of polynomials over \mathbb{Z}_p with variable x . Given an arithmetic circuit $f : \mathbb{Z}_p^t \rightarrow \mathbb{Z}_p$, there exists another circuit $\hat{f} : \mathcal{Q}_{\text{poly}}^t \rightarrow \mathcal{Q}_{\text{poly}}$ and function $\phi_x : \mathcal{Q}_{\text{poly}} \rightarrow \mathbb{Z}_p$ such that

$$\phi_a(\hat{f}(y^{(1)}, \dots, y^{(t)})) = f(\phi_a(y^{(1)}), \dots, \phi_a(y^{(t)}))$$

where $a \in \mathbb{Z}_p$ and $y^{(1)}, \dots, y^{(t)} \in \mathcal{Q}_{\text{poly}}$.

PROOF. We define the function $\phi_a : \mathcal{Q}_{\text{poly}} \rightarrow \mathbb{Z}_p$ as $\phi_a(y) = y(a)$ for any polynomial $y(x) \in \mathcal{Q}_{\text{poly}}$ and $a \in \mathbb{Z}_p$ is the value of variable x . Observe that ϕ_a is a homomorphism from $\mathcal{Q}_{\text{poly}} = \mathbb{Z}_p[x]$ to \mathbb{Z}_p , i.e., $\forall y^{(1)}, y^{(2)} \in \mathcal{Q}_{\text{poly}}$, we have $\phi_a(y^{(1)} + y^{(2)}) = \phi_a(y^{(1)}) + \phi_a(y^{(2)})$ and $\phi_a(y^{(1)} \cdot y^{(2)}) = \phi_a(y^{(1)}) \cdot \phi_a(y^{(2)})$.

Let $\hat{f} : \mathcal{Q}_{\text{poly}}^n \rightarrow \mathcal{Q}_{\text{poly}}$ be an arithmetic circuit similar to f except that the operation in \mathbb{Z}_p at each gate is replaced by the corresponding operation over polynomials in $\mathbb{Z}_p[x]$. For any $a \in \mathbb{Z}_p$, a homomorphism ϕ_a defined as above, any circuit f and $y^{(1)}, \dots, y^{(t)} \in \mathcal{Q}_{\text{poly}}$, the following property holds: $\phi_a(\hat{f}(y^{(1)}, \dots, y^{(t)})) = f(\phi_a(y^{(1)}), \dots, \phi_a(y^{(t)}))$. Therefore, Claim 1 is proved. \square

Let $C_i = (y_0^{(i)}, y_1^{(i)})$ be the ciphertext obtained by running the **Eval** procedure with program $\mathcal{P}_i = (f_i, \tau_{i,1}, \dots, \tau_{i,n_i})$. As

m_i is the output of $\text{Dec}(\text{SK}, \mathcal{P}_i, C_i)$, we have the following conditions: $c_i = y_0^{(i)}, R_i = y_0^{(i)} + y_1^{(i)}s \bmod p$.

Let $(y_0^{(i)}, y_1^{(i)})$ be the coefficients of degree-1 polynomial $y^{(i)}(x) \in \mathcal{Q}_{\text{poly}}$ for all $i = 1, \dots, t$. Consider the output $C^* = (y_0^*, y_1^*)$ of $\text{Eval}(\text{PK}, \mathcal{P}^*, (C_1, \dots, C_t))$. Following Claim 1, there exists another circuit $\hat{f} : \mathcal{Q}_{\text{poly}}^t \rightarrow \mathcal{Q}_{\text{poly}}$ with the same structure as f , and (y_0^*, y_1^*) are the coefficients of degree-1 polynomial y^* such that $y^* = \hat{f}(y^{(1)}, \dots, y^{(t)})$.

As in Claim 1, we show that (1) is satisfied with $a = s$:

$$\begin{aligned} y^*(s) &= \phi_s(\hat{f}(y^{(1)}, \dots, y^{(t)})) = f(\phi_s(y^{(1)}), \dots, \phi_s(y^{(t)})) \\ &= f(R_1, \dots, R_t) = R^* \end{aligned}$$

Similarly, with $a = 0$, we have:

$$\begin{aligned} y^*(0) &= \phi_0(\hat{f}(y^{(1)}, \dots, y^{(t)})) = f(\phi_0(y^{(1)}), \dots, \phi_0(y^{(t)})) \\ &= f(m_1 - \kappa_1, \dots, m_t - \kappa_t) \\ &= m^* - \kappa^* \end{aligned}$$

In addition, we have $y_0^* = y^*(0)$, so $m^* = y_0^* + \kappa^* \bmod p$. It follows that $m^* \leftarrow \text{Dec}(\text{SK}, \mathcal{P}^*, C^*)$.

4.3 Security

Let $\mathcal{H}_l = (\text{KeyGen}, \text{Enc}, \text{Eval}, \text{Dec})$ denote the linear verifiable homomorphic encryption l -HEMAC scheme as defined in Section 4.1. We formally state and prove the semantic security and unforgeability of l -HEMAC.

THEOREM 1. *Suppose that F is a secure pseudo-random function (PRF). The proposed l -HEMAC scheme is semantically secure according to Definition 1.*

PROOF. We prove the theorem using two games Game 0 and Game 1. Let W_0 and W_1 be the events that adversary \mathcal{A} wins the semantic security game in Game 0 and Game 1, respectively.

Game 0 is identical to Attack 1 applied to the \mathcal{H}_l scheme. We have:

$$\Pr[W_0] = \Pr[b = b'] \quad (2)$$

Game 1 is the same as Game 0, except that the PRF is replaced by a truly random function. That is, the challenger generates $r_\tau, k_\tau \xleftarrow{\$} \mathbb{Z}_p$ instead of $r_\tau = F_K(\tau||1)$ and $k_\tau = F_K(\tau||2)$ in the Enc procedure. Then there is a PRF adversary \mathcal{B} such that:

$$|\Pr[W_0] - \Pr[W_1]| \leq \text{prf-adv}[\mathcal{B}, F] \quad (3)$$

In the adversary's view, ciphertext $C^* = (y_0^*, y_1^*)$ where $y_0^* = m_b - k_{\tau^*} \bmod p$ and $y_1^* = \frac{r_{\tau^*} - (m_b - k_{\tau^*})}{s} \bmod p$. Since r_{τ^*}, k_{τ^*} are random values in \mathbb{Z}_p , y_0^*, y_1^* are uniformly distributed over \mathbb{Z}_p . Hence, the probability that \mathcal{A} guesses $b = b'$ correctly is exactly $1/2$ and:

$$\Pr[W_1] = 1/2 \quad (4)$$

Putting together equations (2), (3), and (4),

$$\Pr[b = b'] \leq \text{prf-adv}[\mathcal{B}, F] + \frac{1}{2}$$

This means advantage $\text{ss-adv}[\mathcal{A}, \mathcal{H}_l]$ of \mathcal{A} in Attack 1 is negligible, thus completing the proof of Theorem 1. \square

THEOREM 2. *Suppose that F is a PRF. The proposed l -HEMAC scheme is unforgeable. Specifically, for all \mathcal{H}_l adversaries \mathcal{A} , there exists a PRF adversary \mathcal{B} such that:*

$$\text{uf-cmva}[\mathcal{A}, \mathcal{H}_l] \leq \text{prf-adv}[\mathcal{B}, F] + \frac{Q+1}{p-Q}$$

where $\text{prf-adv}[\mathcal{B}, F]$ is the advantage of adversary \mathcal{B} winning the PRF security game with respect to function F , and Q is an upper bound on the number of verification queries made by \mathcal{A} during Attack 2.

PROOF. The proof is fashioned after the one by Agrawal and Boneh in [1]. We prove the theorem using a sequence of two games Game 0 and Game 1. Let W_0, W_1 be the events that \mathcal{A} wins Attack 2 in Game 0 and Game 1, respectively.

Game 0 is identical to Attack 2 applied to the \mathcal{H}_l scheme. Therefore,

$$\Pr[W_0] = \text{uf-cmva}[\mathcal{A}, \mathcal{H}_l] \quad (5)$$

Game 1 is the same as Game 0, but the PRF is replaced by a truly random function. That is, the challenger generates $r_\tau, k_\tau \xleftarrow{\$} \mathbb{Z}_p$ instead of $r_\tau = F_K(\tau||1)$ and $k_\tau = F_K(\tau||2)$ in the Enc procedure for ciphertext queries. Then there is a PRF adversary \mathcal{B} such that:

$$|\Pr[W_1] - \Pr[W_0]| \leq \text{prf-adv}[\mathcal{B}, F] \quad (6)$$

The challenger in Game 1 works as follows.

Ciphertext Queries. The adversary submits queries (τ_i, m_i) where τ_i is the label of datum m_i . The challenger creates a new list T for tracking the queries from \mathcal{A} in the game. For the i -th query, if T does not contain a tuple (τ_i, \cdot, \cdot) , i.e., the label τ_i was never queried, the challenger responds as follows:

- Generate $r_{\tau_i}, k_{\tau_i} \xleftarrow{\$} \mathbb{Z}_p$
- Compute $y_{i,0} = m_i - k_{\tau_i} \bmod p, y_{i,1} = \frac{r_{\tau_i} - y_{i,0}}{s} \bmod p$.
- Send $(y_{i,0}, y_{i,1})$ to \mathcal{A} and update $T = T \cup \{(\tau_i, r_{\tau_i}, k_{\tau_i})\}$.

If $(\tau_i, \cdot, \cdot) \in T$, i.e., label τ_i was previous queried, the challenger rejects the query.

Verification Queries. The adversary submits queries (\mathcal{P}_i, C_i) where program $\mathcal{P}_i = (f_i, (\tau_{i,1}, \dots, \tau_{i,n_i}))$ and $C_i = (y_{i,0}, y_{i,1})$. The challenger responds to the i -th verification query as follows:

- If $(\tau_{i,j}, \cdot, \cdot) \in T$ for all $j = 1, \dots, n_i$, the challenger retrieves the random values $r_{\tau_{i,j}}, k_{\tau_{i,j}}$ corresponding to $\tau_{i,j}$ from T , then returns the output of $\text{Dec}(\text{SK}, \mathcal{P}_i, C_i)$.
- If there exists some $j \in \{1, \dots, n_i\}$ such that $(\tau_{i,j}, \cdot, \cdot) \notin T$, the challenger generates random values $r_{\tau_{i,j}}, k_{\tau_{i,j}} \xleftarrow{\$} \mathbb{Z}_p$ and returns the output of $\text{Dec}(\text{SK}, \mathcal{P}_i, C_i)$.

Eventually the adversary outputs (\mathcal{P}, C^*) where $\mathcal{P} = (f^*, \tau_1^*, \dots, \tau_{n^*}^*)$ and $C^* = (y_0^*, y_1^*)$. The adversary wins the game if any of the two following types of forgery occurs.

- Type 1 forgery: If there exists $j \in \{1, \dots, n^*\}$ such that $(\tau_j^*, \cdot, \cdot) \notin T$, the challenger generates $r_{\tau_j^*}, k_{\tau_j^*} \xleftarrow{\$} \mathbb{Z}_p$. The random values corresponding to the remaining labels are retrieved from list T . Let $R^* = f^*(r_{\tau_1^*}, \dots, r_{\tau_{n^*}^*})$. The adversary wins the game if:

$$R^* = y_0^* + y_1^* \cdot s \bmod p \quad (7)$$

- Type 2 forgery: If $(\tau_j^*, \cdot, \cdot) \in T$ for all $j = 1, \dots, n^*$, the challenger retrieves $(r_{\tau_j^*}, k_{\tau_j^*})$ from list T . Let $\{(r'_{\tau_j^*}, k'_{\tau_j^*}, m'_j)\}_{j=1}^{n^*}$ be the random values and data corresponding to the labels $\tau_1^*, \dots, \tau_{n^*}^*$. Let $\kappa^* = f^*(k'_{\tau_1^*}, \dots, k'_{\tau_{n^*}^*})$ and $R^* = f^*(r'_{\tau_1^*}, \dots, r'_{\tau_{n^*}^*})$. Let $m^* = y_0^* + \kappa^* \bmod p$. The adversary wins the game if:

$$R^* = y_0^* + y_1^* \cdot s \bmod p \quad (8)$$

and $m^* \neq m' = f^*(m'_1, \dots, m'_{n^*})$.

We now compute the probability of adversary \mathcal{A} winning Game 1. Let B_i be the event that the adversary wins the game after i verification queries. Let Q be the upper bound on the number of verification queries requested by the adversary. We have:

$$\Pr[W_1] = \Pr\left[\bigvee_{i=0}^Q B_i\right] \leq \sum_{i=0}^Q \Pr[B_i].$$

Let $V, \neg V$ be the events that the adversary outputs a type 1 forgery and type 2 forgery, respectively.

- Event V happens (type 1 forgery): The left hand side of (7) is a random value in \mathbb{Z}_p that is independent of the adversary's view. In addition, since s is a secret key, the probability that equation (7) holds is exactly $1/p$. Hence,

$$\Pr[B_i \wedge V] = \frac{1}{p} \cdot \Pr[V] \quad (9)$$

- Event $\neg V$ happens (type 2 forgery): In this case, adversary \mathcal{A} uses program $\mathcal{P} = (f^*, \tau_1^*, \dots, \tau_{n^*}^*)$ in which all the labels are posed in previous ciphertext queries. Event B_i happens if $m^* \neq f^*(m'_1, \dots, m'_{n^*})$ and equation (8) holds.

Let C'_j be the ciphertext corresponding to label τ_j^* in a previous ciphertext query, for all $j = 1, \dots, n^*$. Define $C' = (y'_0, y'_1) \leftarrow \text{Eval}(\text{PP}, \mathcal{P}, (C'_1, \dots, C'_{n^*}))$. Since C' is a valid ciphertext for $m' = f^*(m'_1, \dots, m'_{n^*})$, the following relation holds:

$$R^* = f^*(r'_{\tau_1^*}, \dots, r'_{\tau_{n^*}^*}) = y'_0 + y'_1 \cdot s \bmod p \quad (10)$$

Subtracting (10) from (8), we obtain:

$$(y_0^* - y'_0) + (y_1^* - y'_1) \cdot s \bmod p = 0 \quad (11)$$

Since $m^* \neq f^*(m'_1, \dots, m'_{n^*})$, we know that $C^* \neq C'$, implying that $y_0^* \neq y'_0$ and $y_1^* \neq y'_1$. Hence, in producing a valid forgery, \mathcal{A} must guess secret key s .

As s is uniformly distributed over \mathbb{Z}_p , we have $\Pr[B_0 \wedge \neg V] = 1/p \cdot \Pr[\neg V]$. After the first verification query, since there is only one value of s that satisfies equation (11), the number of possible values for s becomes $p - 1$. Therefore, after i queries, the adversary can exclude i possible values of s , meaning that the number of possible values for s is $(p - i)$. Thus,

$$\Pr[(B_i \wedge \neg V)] \leq \frac{1}{p - i} \cdot \Pr[\neg V] \quad (12)$$

From equations (9) and (12), we obtain:

$$\Pr[B_i] \leq \frac{1}{p - i} \cdot (\Pr[V] + \Pr[\neg V]) \leq \frac{1}{p - i}$$

Finally, we have:

$$\Pr[W_1] \leq \sum_{i=0}^Q \Pr[B_i] \leq \frac{Q + 1}{p - Q} \quad (13)$$

Putting together equations (5), (6) and (13),

$$\text{uf-cmva}[\mathcal{A}, \mathcal{H}_l] \leq \text{prf-adv}[\mathcal{B}, F] + \frac{Q + 1}{p - Q}$$

Since $p \approx 2^\lambda$, $\frac{Q+1}{p-Q} = \text{negl}(\lambda)$ thus completing the proof of Theorem 2. \square

5. VERIFIABLE HOMOMORPHIC ENCRYPTION FOR QUADRATIC FUNCTIONS

In this section, we present a verifiable homomorphic encryption scheme for multivariate quadratic functions, denoted as q -HEMAC. The construction combines the Paillier cryptosystem [30] with a homomorphic MAC scheme. For q -HEMAC, we formulate a new homomorphic MAC scheme in which the authenticator does not leak any information about the message, without using bilinear maps. The correctness proof for q -HEMAC is similar to the one in Section 4.2, and is omitted.

Let $f : \mathbb{F}^n \rightarrow \mathbb{F}$ be an arithmetic circuit with addition gates and multiplication gates such that any multiplication gate, for which none of its two inputs are constants, can only be followed by addition gates or multiplication gates with at least one constant as input. Without loss of generality, we define circuit f as a quadratic multivariate polynomial:

$$f(x_1, \dots, x_n) = \sum_{i,j \in [1,n]} \alpha_{i,j} x_i x_j + \sum_{l \in [1,n]} \alpha_l x_l + \alpha \in \mathbb{F}$$

for some constants $\alpha_{i,j}, \alpha_l, \alpha \in \mathbb{F}$ and x_i, x_j, x_l taking arbitrary values in \mathbb{F} .

5.1 Construction

KeyGen(1^λ). Let p_1, p_2 be prime numbers with roughly $\lambda/2$ bits, where λ is a security parameter. Run the Paillier key generation algorithm as defined in Section 3.3 to generate public parameters $N = p_1 p_2$ and g' . Choose a random seed $K \xleftarrow{\$} \mathbb{Z}_N$ for the pseudo-random function $F'_K : \{0, 1\}^* \rightarrow \mathbb{Z}_N$. Next, let q be a large safe prime number with roughly λ bits such that $q < N$. Let \mathbb{Z}_q^* be a multiplicative cyclic group of order $q - 1$ on which the discrete logarithm problem is hard, and let g be a generator of \mathbb{Z}_q^* . Choose a random seed $R \xleftarrow{\$} \mathbb{Z}_{q-1}$ for the pseudo-random function $F_R : \{0, 1\}^* \rightarrow \mathbb{Z}_{q-1}$, and a random number $s \xleftarrow{\$} \mathbb{Z}_{q-1}^*$. Publish the public key $\text{PK} = (N, g', g, q)$, and retain the secret key $\text{SK} = (p_1, p_2, K, R, s)$. The data space is $\mathcal{M} = \mathbb{Z}_{q-1}$ and $\mathbb{F} = \mathbb{Z}_{q-1}$.

Enc(SK, τ, m). Given secret key $\text{SK} = (p_1, p_2, K, R, s)$, proceed as follows to encrypt a datum $m \in \mathbb{Z}_{q-1}$ with label $\tau \in \{0, 1\}^*$. First, compute an encryption key $k_\tau = F'_K(\tau)$ and a pseudo-random value $r_\tau = F_R(\tau)$. Then, choose a random number $u \xleftarrow{\$} \mathbb{Z}_N^*$, and output a level-1 ciphertext $C = (c_0, c_1, y_0, Y_1) \in \mathbb{Z}_{N^2} \times \mathbb{Z}_N \times \mathbb{Z}_{q-1} \times \mathbb{Z}_q^*$ where:

$$\begin{aligned} c_0 &= \text{E}(m) = g'^m \cdot u^N \bmod N^2, & c_1 &= m - k_\tau \bmod N, \\ y_0 &= \frac{m - r_\tau}{s} \bmod (q - 1), & Y_1 &= g^{r_\tau} \bmod q \end{aligned}$$

Eval(PK, \mathcal{P} , \vec{C}). Given public key PK = (N, g', g, q) , a labeled program $\mathcal{P} = (f, \tau_1, \dots, \tau_n)$, and a vector of ciphertexts $\vec{C} = (C_1, \dots, C_n)$ where $C_i = (c_0^{(i)}, c_1^{(i)}, y_0^{(i)}, Y_1^{(i)}) \in \mathbb{Z}_{N^2}^* \times \mathbb{Z}_N \times \mathbb{Z}_{q-1} \times \mathbb{Z}_q^*$ for all $i = 1, \dots, n$. The procedure goes through a series of gates in a new circuit \tilde{f} with the same structure as f , except that the input is the vector of ciphertexts \vec{C} . The output of the Eval procedure is the output of GateEval at the output gate of the new circuit \tilde{f} . GateEval is defined below.

GateEval(PK, GATE, C_1, C_2). Let $C_i = (c_0^{(i)}, c_1^{(i)}, y_0^{(i)}, Y_1^{(i)})$ for $i = 1, 2$ be the ciphertext of $m_i \in \mathbb{Z}_{q-1}$. Proceed gate-by-gate as follows.

- **GATE₊** : On input of two level-1 ciphertexts: The output of the gate is a level-1 ciphertext $C = (c_0, c_1, y_0, Y_1) \in \mathbb{Z}_{N^2}^* \times \mathbb{Z}_N \times \mathbb{Z}_{q-1} \times \mathbb{Z}_q^*$ where

$$c_0 = c_0^{(1)} \cdot c_0^{(2)} \bmod N^2, \quad c_1 = c_1^{(1)} + c_1^{(2)} \bmod N, \\ y_0 = y_0^{(1)} + y_0^{(2)} \bmod (q-1), \quad Y_1 = Y_1^{(1)} \cdot Y_1^{(2)} \bmod q.$$

On input of two level-2 ciphertexts: The output of the gate is a level-2 ciphertext $C = (c_0, y_0, Y_1) \in \mathbb{Z}_{N^2}^* \times \mathbb{Z}_{q-1} \times \mathbb{Z}_q^*$ where:

$$c_0 = c_0^{(1)} \cdot c_0^{(2)} \bmod N^2, \\ y_0 = y_0^{(1)} + y_0^{(2)} \bmod (q-1), \quad Y_1 = Y_1^{(1)} \cdot Y_1^{(2)} \bmod q.$$

On input of a level-1 and a level-2 ciphertexts: We assume that the level of ciphertext C_1 is '1' and the level of ciphertext C_2 is '2'. The output of the gate is a level-2 ciphertext $C = (c_0, y_0, Y_1) \in \mathbb{Z}_{N^2}^* \times \mathbb{Z}_{q-1} \times \mathbb{Z}_q^*$ where:

$$c_0 = g'^{c_1^{(1)}} \cdot c_0^{(2)} \bmod N^2, \quad y_0 = y_0^{(2)}, \\ Y_1 = g^{y_0^{(1)}} \cdot Y_1^{(2)} \bmod q.$$

- **GATE_x** : On input of two level-1 ciphertexts: The output of the gate is a level-2 ciphertext $C = (c_0, y_0, Y_1) \in \mathbb{Z}_{N^2}^* \times \mathbb{Z}_{q-1} \times \mathbb{Z}_q^*$ where:

$$c_0 = c_0^{(1)c_1^{(2)}} \cdot c_0^{(2)c_1^{(1)}} / g'^{c_1^{(1)} \cdot c_1^{(2)}} \bmod N^2, \\ y_0 = y_0^{(1)} y_0^{(2)} \bmod (q-1), \quad Y_1 = Y_1^{(1)y_0^{(2)}} Y_1^{(2)y_0^{(1)}} \bmod q.$$

- **GATE_{x,c}** : On input of a constant α and a level-1 ciphertext: We assume that $C_1 = \alpha$. The output of the gate is a level-1 ciphertext $C = (c_0, c_1, y_0, Y_1) \in \mathbb{Z}_{N^2}^* \times \mathbb{Z}_N \times \mathbb{Z}_{q-1} \times \mathbb{Z}_q^*$ where:

$$c_0 = c_0^{(2)\alpha} \bmod N^2, \quad c_1 = \alpha \cdot c_1^{(2)} \bmod N, \\ y_0 = \alpha \cdot y_0^{(2)} \bmod (q-1), \quad Y_1 = Y_1^{(2)\alpha} \bmod q.$$

On input of a constant α and a level-2 ciphertext: We assume that $C_1 = \alpha$. The output of the gate is a level-2 ciphertext $C = (c_0, y_0, Y_1) \in \mathbb{Z}_{N^2}^* \times \mathbb{Z}_{q-1} \times \mathbb{Z}_q^*$ where:

$$c_0 = c_0^{(2)\alpha} \bmod N^2, \\ y_0 = \alpha \cdot y_0^{(2)} \bmod (q-1), \quad Y_1 = Y_1^{(2)\alpha} \bmod q.$$

Finally, return $C = (c_0, c_1, y_0, Y_1) \in \mathbb{Z}_{N^2}^* \times \mathbb{Z}_N \times \mathbb{Z}_{q-1} \times \mathbb{Z}_q^*$ if the level of circuit f is '1', or return $C = (c_0, y_0, Y_1) \in \mathbb{Z}_{N^2}^* \times \mathbb{Z}_{q-1} \times \mathbb{Z}_q^*$ if the level of circuit f is '2'.

Dec(SK, \mathcal{P} , C). Given secret key SK = (p_1, p_2, K, R, s) , a labeled program $\mathcal{P} = (f, \tau_1, \dots, \tau_n)$, and ciphertext C . For every input label $\tau_i \in \{0, 1\}^*$ of circuit f , first compute one-time encryption key $k_{\tau_i} = F'_K(\tau_i)$ and pseudo-random value $r_{\tau_i} = F_R(\tau_i)$. Then, compute decryption key $\kappa = f'(k_{\tau_1}, \dots, k_{\tau_n})$ and $R = f(r_{\tau_1}, \dots, r_{\tau_n})$ where f' is a new circuit which has the same structure as f , except that the input and output space are \mathbb{Z}_N . Consider two different cases depending on the level of ciphertext C .

- **Level-1 ciphertext.** Let $C = (c_0, c_1, y_0, Y_1) \in \mathbb{Z}_{N^2}^* \times \mathbb{Z}_N \times \mathbb{Z}_{q-1} \times \mathbb{Z}_q^*$. Decrypt $m = f(m_1, \dots, m_n)$ with decryption key κ as:

$$m = c_1 + \kappa \bmod N \quad (14)$$

To verify the correctness of m , use secret value s to check for the following condition:

$$m = y_0 \cdot s + R \bmod (q-1) \quad (15)$$

If the condition holds, return m ; otherwise, return \perp .

- **Level-2 ciphertext.** Let $C = (c_0, y_0, Y_1) \in \mathbb{Z}_{N^2}^* \times \mathbb{Z}_{q-1} \times \mathbb{Z}_q^*$. Decrypt $m = f(m_1, \dots, m_n)$ with secret values p_1, p_2 and decryption key κ as:

$$m = D(c_0) + \kappa \bmod N \quad (16)$$

where $D(c_0) = \frac{L(c_0^{\lambda'} \bmod N^2)}{L(g^{\lambda'} \bmod N^2)} \bmod N$ as defined in Section 3.3.

To verify the correctness of m , use secret value s to check for the following condition:

$$g^{m+R} \bmod q = g^{y_0 \cdot s^2} \cdot Y_1^s \bmod q \quad (17)$$

If the condition holds, return m ; otherwise, return \perp .

5.2 Security

Let $\mathcal{H}_q = (\text{KeyGen}, \text{Enc}, \text{Eval}, \text{Dec})$ denote q -HEMAC, the verifiable homomorphic encryption scheme for multivariate quadratic functions defined in Section 5.1. We now formally state and prove the semantic security and unforgeability assurances of q -HEMAC. We omit the unforgeability proof for level-1 circuit, which is similar to the proof for Theorem 2.

THEOREM 3. Suppose that F and F' are secure PRFs, the additively homomorphic Paillier encryption is semantically secure, and the discrete logarithm problem is hard in \mathbb{Z}_q^* . Then, the proposed q -HEMAC scheme is semantically secure according to Definition 1.

PROOF. We prove the theorem using two games, Game 0 and Game 1. Let W_0 and W_1 be the events that adversary \mathcal{A} wins the semantic security game in Game 0 and Game 1, respectively.

Game 0 is identical to Attack 1 applied to the \mathcal{H}_q scheme. We have:

$$\Pr[W_0] = \Pr[b = b'] \quad (18)$$

Game 1 is the same as Game 0, except that the PRFs are replaced by truly random functions. That is, the challenger generates $k_\tau \xleftarrow{\$} \mathbb{Z}_N$, $r_\tau \xleftarrow{\$} \mathbb{Z}_{q-1}$ instead of $k_\tau = F'_K(\tau)$, $r_\tau = F_R(\tau)$ in the Enc procedure. Then there is a PRF adversary \mathcal{B} such that:

$$|\Pr[W_0] - \Pr[W_1]| \leq \epsilon_F + \epsilon_{F'} \quad (19)$$

where ϵ_F and $\epsilon_{F'}$ are the negligible advantages of \mathcal{B} in breaking the security of F and F' , respectively.

In Game 1, ciphertext $C^* = (c_0^*, c_1^*, y_0^*, Y_1^*)$ where c_0^* is the output of Paillier encryption, $c_1^* = m_b - k_{\tau^*} \bmod N$, $y_0^* = \frac{m_b - r_{\tau^*}}{s} \bmod (q-1)$, and $Y_1^* = g^{k_{\tau^*}} \bmod q$. In the adversary's view, k_{τ^*}, r_{τ^*} are random values in \mathbb{Z}_N and \mathbb{Z}_{q-1} , so c_1^*, y_0^* are uniformly distributed over $\mathbb{Z}_N, \mathbb{Z}_{q-1}$ respectively, and Y_1^* is also uniformly distributed over \mathbb{Z}_q^* . In addition, since the Paillier cryptosystem is semantically secure and discrete logarithm assumption holds in \mathbb{Z}_q^* , there are adversaries \mathcal{PA} and \mathcal{D} such that:

$$\Pr[W_1] = \frac{1}{2} + \epsilon_{pa} + \epsilon_{dlog} \quad (20)$$

where ϵ_{pa} is the negligible advantages of \mathcal{PA} in breaking the security of Paillier cryptosystem, and ϵ_{dlog} is the negligible advantages of \mathcal{D} in breaking the discrete logarithm problem.

Putting together equations (18), (19) and (20),

$$\Pr[b = b'] \leq \frac{1}{2} + \epsilon_F + \epsilon_{F'} + \epsilon_{pa} + \epsilon_{dlog}$$

This means advantage $\text{ss-adv}[\mathcal{A}, \mathcal{H}_f]$ of \mathcal{A} in Attack 1 is negligible, thus completing the proof of Theorem 3. \square

THEOREM 4. Suppose that F and F' are secure PRFs and the discrete logarithm problem is hard in \mathbb{Z}_q^* . The proposed q -HEMAC scheme is unforgeable according to Definition 2.

PROOF. The proof is fashioned after the one given by Agrawal and Boneh in [1]. We prove the theorem using a sequence of two games, Game 0 and Game 1. Let W_0 and W_1 be the events that \mathcal{A} wins Attack 2 in Game 0 and Game 1, respectively.

Game 0 is identical to Attack 2 applied to the \mathcal{H}_q scheme. Therefore,

$$\Pr[W_0] = \text{uf-cmva}[\mathcal{A}, \mathcal{H}_q] \quad (21)$$

Game 1 is the same as Game 0, but the PRFs are replaced by truly random functions. That is, the challenger generates $k_\tau \xleftarrow{\$} \mathbb{Z}_N, r_\tau \xleftarrow{\$} \mathbb{Z}_{q-1}$ instead of $k_\tau = F'_K(\tau), r_\tau = F_R(\tau)$ in the Enc procedure for ciphertext queries. Then there is a PRF adversary \mathcal{B} such that:

$$|\Pr[W_1] - \Pr[W_0]| \leq \epsilon_F + \epsilon_{F'} \quad (22)$$

where ϵ_F and $\epsilon_{F'}$ are the negligible advantages of \mathcal{B} in breaking the security of F and F' , respectively.

The complete challenger in Game 1 is described below.

Ciphertext Queries. The adversary submits queries (τ_i, m_i) where τ_i is the label of datum m_i . The challenger creates a new list T for tracking the queries from \mathcal{A} in this game. For the i -th query, if T does not contain a tuple (τ_i, \cdot, \cdot) , the challenger responds as follows:

- Generate $k_{\tau_i} \xleftarrow{\$} \mathbb{Z}_N, r_{\tau_i} \xleftarrow{\$} \mathbb{Z}_{q-1}$.
- Compute $c_{i,0} = E(m_i)$, $c_{i,1} = m_i - k_{\tau_i} \bmod N$, $y_{i,0} = \frac{m_i - r_{\tau_i}}{s} \bmod (q-1)$, and $Y_{i,1} = g^{r_{\tau_i}} \bmod q$.
- Send $(c_{i,0}, c_{i,1}, y_{i,0}, Y_{i,1})$ to \mathcal{A} , and update list $T = T \cup \{(\tau_i, r_{\tau_i}, k_{\tau_i})\}$.

If $(\tau_i, \cdot, \cdot) \in T$, i.e., label τ_i was queried previously, the challenger rejects the query.

Verification queries. The adversary submits queries (\mathcal{P}_i, C_i) where $\mathcal{P}_i = (f_i, \tau_{i,1}, \dots, \tau_{i,n_i})$ and $C_i = (c_{i,0}, y_{i,0}, Y_{i,1})$. The challenger responds to the i -th verification query as follows.

- If $(\tau_{i,j}, \cdot, \cdot) \in T$ for all $j = 1, \dots, n_i$, the challenger retrieves $k_{\tau_{i,j}}, r_{\tau_{i,j}}$ from list T , and returns the output of $\text{Dec}(\text{SK}, \mathcal{P}_i, C_i)$.

- If there exists some $j \in \{1, \dots, n_i\}$ such that $(\tau_{i,j}, \cdot, \cdot) \notin T$, the challenger generates $k_{\tau_{i,j}} \xleftarrow{\$} \mathbb{Z}_N$ and $r_{\tau_{i,j}} \xleftarrow{\$} \mathbb{Z}_{q-1}$, and returns the output of $\text{Dec}(\text{SK}, \mathcal{P}_i, C_i)$.

Eventually, the adversary outputs (\mathcal{P}, C^*) where $\mathcal{P} = (f^*, \tau_1^*, \dots, \tau_{n^*}^*)$ and $C^* = (c_0^*, y_0^*, Y_1^*)$. The adversary wins the game if any of the two types of forgery happens.

- **Type 1 forgery:** If there exists some $j \in \{1, \dots, n^*\}$ such that $(\tau_j^*, \cdot, \cdot) \notin T$, the challenger generates $k_{\tau_j^*} \xleftarrow{\$} \mathbb{Z}_N$ and $r_{\tau_j^*} \xleftarrow{\$} \mathbb{Z}_{q-1}$. The random values corresponding to the remaining labels are retrieved from list T . Let $\kappa^* = f^{*'}(k_{\tau_1^*}, \dots, k_{\tau_{n^*}^*})$ and $R^* = f^*(r_{\tau_1^*}, \dots, r_{\tau_{n^*}^*})$. Let $m^* = D(c_0^*) + \kappa^* \bmod N$. The adversary wins the game if:

$$g^{m^* + R^*} \bmod q = g^{y_0^* \cdot s^2} \cdot (Y_1^*)^s \bmod q \quad (23)$$

- **Type 2 forgery:** If $(\tau_j^*, \cdot, \cdot) \in T$ for all $j = 1, \dots, n^*$, the challenger retrieves $(k_{\tau_j^*}, r_{\tau_j^*})$ from list T . Let $\{(k'_{\tau_j^*}, r'_{\tau_j^*}, m'_j)\}_{j=1}^{n^*}$ be the random values and data corresponding to the labels $\{\tau_j^*\}_{j=1}^{n^*}$. Let $\kappa^* = f^{*'}(k'_{\tau_1^*}, \dots, k'_{\tau_{n^*}^*})$ and $R^* = f^*(r'_{\tau_1^*}, \dots, r'_{\tau_{n^*}^*})$. Let $m^* = D(c_0^*) + \kappa^* \bmod N$. The adversary wins the game if:

$$g^{m^* + R^*} \bmod q = g^{y_0^* \cdot s^2} \cdot (Y_1^*)^s \bmod q \quad (24)$$

and $m^* \neq m' = f^*(m'_1, \dots, m'_{n^*})$.

We now compute the winning probability of adversary \mathcal{A} in Game 1. Let B_i be the event that the adversary wins the game after i verification queries. Let Q be the upper bound on the number of verification queries requested by the adversary. We have:

$$\Pr[W_1] = \Pr\left[\bigvee_{i=0}^Q B_i\right] \leq \sum_{i=0}^Q \Pr[B_i]$$

Let $V, \neg V$ be the events that the adversary outputs a type 1 forgery and type 2 forgery, respectively.

- **Event V (type 1 forgery)** happens: The left hand side of equation (23) is a random element in \mathbb{G} that is independent of the adversary's view. In addition, since s is a secret key, the probability that equation (23) holds is exactly $2/(q-1)$. Hence,

$$\Pr[B_i \wedge V] = \frac{2}{q-1} \cdot \Pr[V] \quad (25)$$

- **Event $\neg V$ (type 2 forgery)** happens: In this case, adversary \mathcal{A} outputs program $\mathcal{P} = (f^*, \tau_1^*, \dots, \tau_{n^*}^*)$ in which all the labels were used in previous ciphertext queries. Event B_i happens if $m^* \neq f^*(m'_1, \dots, m'_{n^*})$ and equation (24) holds.

Let C'_j be the ciphertext corresponding to label τ_j^* in a previous ciphertext query, for all $j = 1, \dots, n^*$. Define $C' \leftarrow \text{Eval}(\text{PK}, \mathcal{P}, (C'_1, \dots, C'_{n^*}))$. Since C' is a valid

ciphertext for $m' = f^*(m'_1, \dots, m'_{n^*})$, the following relation holds:

$$g^{m' + r^*} \bmod q = g^{y'_0 \cdot s^2} \cdot (Y'_1)^s \bmod q \quad (26)$$

Dividing equation (26) by equation (24),

$$g^{m' - m^*} \bmod q = g^{(y'_0 - y_0^*) \cdot s^2} \cdot (Y'_1 / Y_1^*)^s \bmod q \quad (27)$$

Since $m^* \neq m' = f(m'_1, \dots, m'_{n^*})$, in producing a valid forgery, the adversary must guess the value of secret key s or solve the discrete logarithm problem.

Let S be a set of possible values for s . Since s is uniformly distributed over \mathbb{Z}_{q-1} and there are most two values of s satisfying equation (27), we have $\Pr[B_0 \wedge \neg V] = (2/(q-1) + \epsilon_{\text{dlog}}) \cdot \Pr[\neg V]$ where ϵ_{dlog} is the negligible advantage of an adversary \mathcal{D} in breaking the discrete logarithm problem. After the first verification query, the number of values in S becomes at least $(q-1) - 2$. Therefore, after i queries, the adversary can exclude at most $2i$ possible values of s from S . That means the number of values in S is $(q-1) - 2i$. Thus,

$$\Pr[B_i \wedge \neg V] \leq \left(\frac{2}{q-1-2i} + \epsilon_{\text{dlog}} \right) \cdot \Pr[\neg V] \quad (28)$$

From equations (25) and (28), we obtain:

$$\begin{aligned} \Pr[B_i] &\leq \left(\frac{2}{q-1-2i} + \epsilon_{\text{dlog}} \right) (\Pr[V] + \Pr[\neg V]) \\ &\leq \frac{2}{q-1-2i} + \epsilon_{\text{dlog}} \end{aligned}$$

Finally,

$$\Pr[W_1] \leq \sum_{i=0}^Q \Pr[B_i] \leq \frac{2(Q+1)}{q-1-2Q} + (Q+1) \cdot \epsilon_{\text{dlog}} \quad (29)$$

Putting together equations (21), (22) and (29),

$$\text{uf-cmva}[\mathcal{A}, \mathcal{H}_q] \leq \epsilon_F + \epsilon_{F'} + \frac{2(Q+1)}{q-1-2Q} + (Q+1) \cdot \epsilon_{\text{dlog}}$$

Since $q \approx 2^\lambda$ and Q is an upper bound on the number of verification queries, $\frac{2(Q+1)}{q-1-2Q} = \text{negl}(\lambda)$ thus completing the proof of Theorem 4. \square

6. EMPIRICAL EVALUATION

In this section, we report on a series of experiments designed to evaluate our l -HEMAC and q -HEMAC schemes.

6.1 Set-up

Schemes to be investigated. We implemented l -HEMAC and q -HEMAC in C++, using the GMP library² for large integer arithmetic, and the OpenSSL library³ for AES encryption to simulate pseudo-random functions. The parameter settings for the schemes, corresponding to various security levels, are given below:

- Linear functions: For 128-bit security, we choose $\log p = 128$.

²<https://gmplib.org>

³<https://www.openssl.org>

Table 1: Execution time (msec) of l -HEMAC on linear functions

	Enc	Eval	Dec
		GATE ₊	GATE _{×_c}
128-bit	0.0045	0.00073	0.0014
			0.0035

Table 2: Execution time (msec) of q -HEMAC on quadratic functions

	Enc	Eval		Dec
		GATE ₊	GATE _{×_c}	GATE _×
80-bit	2.89	0.006	0.04	5.67
112-bit	16.11	0.018	0.13	39.21
128-bit	47.48	0.033	0.33	116.45
				46.11

- Multivariate quadratic functions: For 80-bit (respectively 112-bit and 128-bit) security, we choose $\log p_1 = \log p_2 = 512$ ($\log p_1 = \log p_2 = 1024$, $\log p_1 = \log p_2 = 1536$), $\log q = 1024$ ($\log q = 2048$, $\log q = 3072$) and $q < N = p_1 p_2$.

As a baseline for comparison, we use the $\mathcal{VC}_{\text{quad}}$ scheme for multivariate quadratic polynomials of Fiore et al. in [21]. Since there is no available code for the scheme, we implement its ProbGen, Compute and Verify (corresponding to Enc, Eval and Dec in our schemes) procedures with the PBC library⁴ for bilinear group and pairing operations, and HELib library⁵ for the BGV [11] cryptosystem.

Experiment platform. We execute the schemes on a randomly generated table with one million records, in which each record has two attributes – a label and a datum. The experiments are carried out on a MacBook Air with a 1.33 GHz Intel Core i5 processor and 4 GB of memory, running Mac OS X version 10.10.2.

Performance metric. We evaluate the schemes on (a) the time taken by the data owner to encrypt data (with the Enc procedure), (b) the time for the server to evaluate a function (with the Eval procedure), and (c) the time for the client to decrypt a result provided by the server (with the Dec procedure). Every reported timing is averaged over 10,000 trials.

6.2 Experiments

We begin with l -HEMAC on linear functions. The execution times of the procedures in this scheme, for 128-bit security, are tabulated in Table 1. The Enc procedure requires only 4.5 μ sec for encryption, while the Dec procedure requires 3.5 μ sec to decrypt and verify a record. At only 0.73 μ sec and 1.4 μ sec, the overheads incurred by the Eval procedure to add (GATE₊) and multiply by a constant (GATE_{×_c}) are also very low. Figure 1(a) plots the execution times of the Enc, Eval,

⁴<https://crypto.stanford.edu/pbc>

⁵<https://github.com/shaiah/HELlib>

Table 3: Comparative execution time (msec) for operations in the evaluation procedure

	q -HEMAC		$\mathcal{VC}_{\text{quad}}$ [21]	
	80-bit	128-bit	80-bit	128-bit
GATE ₊	0.006	0.033	0.05	0.13
GATE _{×_c}	0.047	0.331	1.92	3.03
GATE _×	5.670	116.45	216.06	353.76

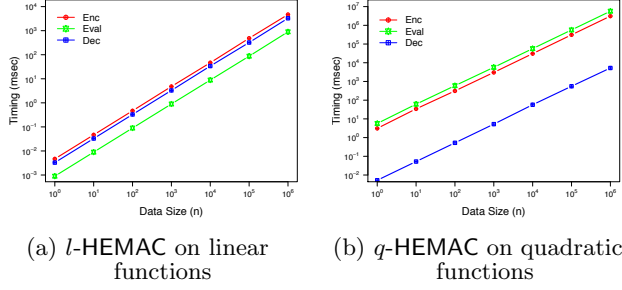


Figure 1: Execution times of our proposed schemes on linear functions and multivariate quadratic functions, with 80-bit security

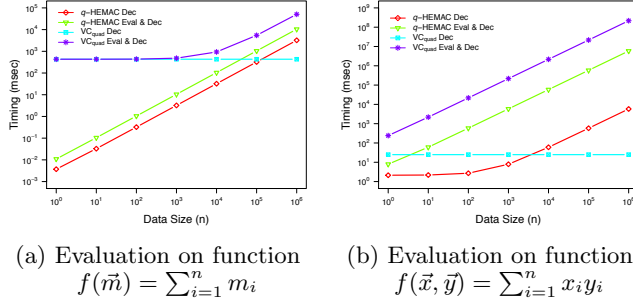


Figure 2: Comparison the decryption time and total execution time which is the sum of the evaluation and decryption time of q -HEMAC and $\mathcal{VC}_{\text{quad}}$, with 80-bit security

Dec procedures of l -HEMAC to encrypt n records, evaluate the circuit $f(m_1, \dots, m_n) = \sum_{i=1}^n \alpha_i m_i$ (where $\alpha_1, \dots, \alpha_n$ are constants) on their ciphertexts, then decrypt and verify the result. While all the three procedures have execution times that are linear to the number of input data, they are very fast. In particular, the Dec procedure incurs only 3.28 sec for $n = 10^6$ data values.

Turning to the q -HEMAC scheme for multivariate quadratic functions, Table 2 summarizes the execution times of its procedures for 80-bit, 112-bit and 128-bit security levels. Figure 1(b) plots the performance of the Enc, Eval, Dec procedures in q -HEMAC at 80-bit security, against the number of records n in evaluating the function $f(\vec{x}, \vec{y}) = \sum_{i=1}^n x_i y_i$ where $\vec{x} = (x_1, \dots, x_n)$ and $\vec{y} = (y_1, \dots, y_n)$ are data vectors. While the q -HEMAC procedures are slower than those in l -HEMAC, they are still efficient. For example, the Dec procedure completes in 5.2 sec for $n = 10^6$ data items.

6.3 Comparison

Next, we compare our proposed q -HEMAC with the existing $\mathcal{VC}_{\text{quad}}$ scheme. Both schemes require expensive computations to encrypt the data. However, q -HEMAC needs only 2,898 sec to encrypt the data set of 1 million records; that is 12 times faster than the encryption in $\mathcal{VC}_{\text{quad}}$ which ran for 36,711 sec.

Table 3 compares the execution times of the evaluation procedure in q -HEMAC and $\mathcal{VC}_{\text{quad}}$. All the operations in our scheme are faster than their counterparts in $\mathcal{VC}_{\text{quad}}$. In particular, for 80-bit security, q -HEMAC performs multiplication and multiplication by a constant at least 38 times faster than $\mathcal{VC}_{\text{quad}}$, whereas addition in q -HEMAC is 8 times faster than in $\mathcal{VC}_{\text{quad}}$.

Ultimately, the usability of the schemes hinges on their total turnaround time, the sum of server evaluation time and client decryption time. To compute a function over n records, the circuit needs to have at least $n-1$ gates. Among the three types of gates, GATE_+ , GATE_{\times_c} and GATE_{\times} , q -HEMAC has the smallest gain over $\mathcal{VC}_{\text{quad}}$ for GATE_+ . To be conservative in quantifying the performance advantage of q -HEMAC, we run the two schemes on a circuit with $n-1$ GATE_+ gates. Figure 2 plots the decryption time and total turnaround time of the competing schemes. Even though the decryption time in q -HEMAC increases linearly with the data size whereas $\mathcal{VC}_{\text{quad}}$ has constant decryption time, q -HEMAC's much faster server evaluation helps it to achieve overall turnaround times that are at least 5 times and 30 times faster than $\mathcal{VC}_{\text{quad}}$'s for linear and quadratic functions respectively.

7. CONCLUSION

Existing studies on verifiable computation on encrypted data are built on fully homomorphic encryption, which require expensive computations on the server and the client. In this paper, we present the first schemes that guarantee the confidentiality of input data and authenticity of outsourced computations, while avoiding the need for fully homomorphic encryption. The first is l -HEMAC, a simple and efficient scheme for linear functions that integrates symmetric key homomorphic encryption with homomorphic MAC. The second scheme, q -HEMAC, supports multivariate quadratic functions and combines the Paillier cryptosystem with a new homomorphic MAC scheme. We provide security analysis to prove that our schemes achieve the desired confidentiality and authenticity requirements. Through empirical evaluations and comparison with the scheme in [21], we confirm the practicality of our schemes.

8. ACKNOWLEDGMENTS

This material is based on research supported by the Singapore National Research Foundation under NCR award number NRF2014NCR-NCR001-012.

9. REFERENCES

- [1] S. Agrawal and D. Boneh. Homomorphic macs: Mac-based integrity for network coding. In *Applied Cryptography and Network Security*, pages 292–305. Springer, 2009.
- [2] P. Ananth, N. Chandran, V. Goyal, B. Kanukurthi, and R. Ostrovsky. Achieving privacy in verifiable computation with multiple servers—without fhe and without pre-processing. In *Public-Key Cryptography—PKC 2014*, pages 149–166. Springer, 2014.
- [3] N. Attrapadung and B. Libert. Homomorphic network coding signatures in the standard model. In *Public Key Cryptography—PKC 2011*, pages 17–34. Springer, 2011.
- [4] M. Backes, D. Fiore, and R. M. Reischuk. Verifiable delegation of computation on outsourced data. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pages 863–874. ACM, 2013.
- [5] M. Barbosa and P. Farshim. Delegatable homomorphic encryption with applications to secure outsourcing of computation. In *Topics in Cryptology—CT-RSA 2012*, pages 296–312. Springer, 2012.

- [6] M. Bellare and C. Namprempre. Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. In *Advances in Cryptology-ASIACRYPT 2000*, pages 531–545. Springer, 2000.
- [7] S. Benabbas, R. Gennaro, and Y. Vahlis. Verifiable delegation of computation over large datasets. In *Advances in Cryptology-CRYPTO 2011*, pages 111–131. Springer, 2011.
- [8] D. Boneh, D. Freeman, J. Katz, and B. Waters. Signing a linear subspace: Signature schemes for network coding. In *Public Key Cryptography-PKC 2009*, pages 68–87. Springer, 2009.
- [9] D. Boneh and D. M. Freeman. Homomorphic signatures for polynomial functions. In *Advances in Cryptology-EUROCRYPT 2011*, pages 149–168. Springer, 2011.
- [10] D. Boneh and D. M. Freeman. Linearly homomorphic signatures over binary fields and new tools for lattice-based signatures. In *Public Key Cryptography-PKC 2011*, pages 1–16. Springer, 2011.
- [11] Z. Brakerski, C. Gentry, and V. Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, pages 309–325. ACM, 2012.
- [12] D. Catalano and D. Fiore. Practical homomorphic macs for arithmetic circuits. In *EUROCRYPT*, volume 7881, pages 336–352. Springer, 2013.
- [13] D. Catalano and D. Fiore. Boosting linearly-homomorphic encryption to evaluate degree-2 functions on encrypted data. Cryptology ePrint Archive, Report 2014/813, 2014.
- [14] D. Catalano, D. Fiore, R. Gennaro, and L. Nizzardo. Generalizing homomorphic macs for arithmetic circuits. In *Public-Key Cryptography-PKC 2014*, pages 538–555. Springer, 2014.
- [15] D. Catalano, D. Fiore, R. Gennaro, and K. Vamvourellis. Algebraic (trapdoor) one-way functions and their applications. In *Theory of Cryptography*, pages 680–699. Springer, 2013.
- [16] D. Catalano, D. Fiore, and B. Warinschi. Adaptive pseudo-free groups and applications. In *Advances in Cryptology-EUROCRYPT 2011*, pages 207–223. Springer, 2011.
- [17] D. Catalano, D. Fiore, and B. Warinschi. Efficient network coding signatures in the standard model. In *Public Key Cryptography-PKC 2012*, pages 680–696. Springer, 2012.
- [18] D. Catalano, D. Fiore, and B. Warinschi. Homomorphic signatures with efficient verification for polynomial functions. In *Advances in Cryptology-CRYPTO 2014*, pages 371–389. Springer, 2014.
- [19] D. Catalano, A. Marcedone, and O. Puglisi. Authenticating computation on groups: New homomorphic primitives and applications. In *Advances in Cryptology-ASIACRYPT 2014*, pages 193–212. Springer, 2014.
- [20] D. Fiore and R. Gennaro. Publicly verifiable delegation of large polynomials and matrix computations, with applications. In *Proceedings of the 2012 ACM conference on Computer and communications security*, pages 501–512. ACM, 2012.
- [21] D. Fiore, R. Gennaro, and V. Pastro. Efficiently verifiable computation on encrypted data. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pages 844–855. ACM, 2014.
- [22] D. M. Freeman. Improved security for linearly homomorphic signatures: A generic framework. In *Public Key Cryptography-PKC 2012*, pages 697–714. Springer, 2012.
- [23] R. Gennaro, C. Gentry, and B. Parno. Non-interactive verifiable computing: Outsourcing computation to untrusted workers. In *Advances in Cryptology-CRYPTO 2010*, pages 465–482. Springer, 2010.
- [24] R. Gennaro and D. Wichs. Fully homomorphic message authenticators. In *Advances in Cryptology-ASIACRYPT 2013*, pages 301–320. Springer, 2013.
- [25] S. Goldwasser, Y. Kalai, R. A. Popa, V. Vaikuntanathan, and N. Zeldovich. Reusable garbled circuits and succinct functional encryption. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, pages 555–564. ACM, 2013.
- [26] C. Joo and A. Yun. Homomorphic authenticated encryption secure against chosen-ciphertext attack. In *Advances in Cryptology-ASIACRYPT 2014*, pages 173–192. Springer, 2014.
- [27] J. Lai, R. H. Deng, H. Pang, and J. Weng. Verifiable computation on outsourced encrypted data. In *Computer Security-ESORICS 2014*, pages 273–291. Springer, 2014.
- [28] S.-Y. Li, R. W. Yeung, and N. Cai. Linear network coding. *Information Theory, IEEE Transactions on*, 49(2):371–381, 2003.
- [29] B. Libert, T. Peters, M. Joye, and M. Yung. Linearly homomorphic structure-preserving signatures and their applications. In *Advances in Cryptology-CRYPTO 2013*, pages 289–307. Springer, 2013.
- [30] P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Advances in cryptology-EUROCRYPT 1999*, pages 223–238. Springer, 1999.
- [31] C. Papamanthou, E. Shi, and R. Tamassia. Signatures of correct computation. In *Theory of Cryptography*, pages 222–242. Springer, 2013.
- [32] B. Parno, M. Raykova, and V. Vaikuntanathan. How to delegate and verify in public: Verifiable computation from attribute-based encryption. In *Theory of Cryptography*, pages 422–439. Springer, 2012.
- [33] A. Shpilka and A. Yehudayoff. Arithmetic circuits: A survey of recent results and open questions. *Foundations and Trends® in Theoretical Computer Science*, 5(3–4):207–388, 2010.