Two-Server Delegation of Computation on Label-Encrypted Data

Xin Chen, Liang Feng Zhang

Abstract—Catalano and Fiore propose a scheme to transform a linearly-homomorphic encryption into a homomorphic encryption scheme capable of evaluating quadratic computations on ciphertexts. Their scheme is based on the linearly-homomorphic encryption (such as Goldwasser-Micali, Paillier and ElGamal) and need to perform large integer operation on servers. Then, their scheme have numerous computations on the servers. At the same time, their scheme cannot verify the computations and cannot evaluate more than degree-4 computations. To solve these problems, we no longer use linearly-homomorphic encryption which based on number theory assumptions. We use label and pseudorandom function to encrypt message, which significantly reduce the computations on the servers and enable us to use homomorphic MACs technology to realize verifiable computations naturally. We also extend the method to construct $d$-server schemes, which allow the client to delegate degree-$d$ computations on outsourced data.

Index Terms—efficiency, verifiable computation, label, homomorphic MACs.

1 INTRODUCTION

The prevalence of cloud computing makes it very popular for the client such as the users of resource-restricted devices to collect data, outsource the data to one or more cloud services, and later freely access the data on demand, even if the client have very limited storage or computing power. The client may not only access the outsourced data by retrieving one or more specific elements, but also request the cloud services to perform computations on the outsourced data and then return the correct results. If there is only one cloud service, then the outsourcing scenario above can be described as follows. A client collects a set of data elements $m_1, \ldots, m_n$, stores these elements on a cloud server, and later asks the server to run a program $P$ over $(m_1, \ldots, m_n)$. The server computes $m = P(m_1, \ldots, m_n)$ and returns $m$.

This simple scenario has incurred significant security concerns. The attacks [30] show that one cannot always trust the cloud services by storing sensitive information on their servers, as the cloud services may not be able to always defeat the attackers from both inside and outside. How to preserve the privacy of the outsourced data is one of the top security concerns. Encrypting the data with the traditional algorithms such as AES [29] and RSA [24] would not only allow the client to preserve the data privacy but also make the server-side computation of $P(m_1, \ldots, m_n)$ impossible. A natural way to resolve this technical difficulty is to use the homomorphic encryption schemes. Fully homomorphic encryption scheme (FHE) [14] allows the server to perform the computation of any program $P$ on the ciphertexts $\text{Enc}(m_1), \ldots, \text{Enc}(m_n)$, instead of the plaintexts, to get a ciphertext of $m = P(m_1, \ldots, m_n)$. The invention of FHE [14] has been a main breakthrough in cryptography. However, today’s FHE constructions [4], [5], [6], [15] still suffer from large parameters and are rather slow. As a result, the FHE-based outsourcing is time-consuming and far from practical.

The notion of homomorphic encryption dates back to Rivest, Adleman and Dertouzos [23]. The first homomorphic encryption schemes were constructed by [18], [21]. On one hand, these schemes allow only linear computations on the encrypted data. On the other hand, these schemes are much more efficient than FHE [4], [5], [6], [14], [15]. In the outsourcing computation scenario, if the client is only interested in a linear combination of the outsourced data, then the FHE can be replaced with a linearly homomorphic encryption scheme [18], [21] and results in strictly faster schemes.

It is possible to extend the linearly homomorphic encryption schemes to a new encryption scheme that enables the computation of nonlinear functions on the ciphertexts. Catalano and Fiore [7] proposed a transformation that can convert any public-space linearly-homomorphic encryption scheme (the message space is a publicly known ring) into a homomorphic encryption scheme supporting quadratic computations. The outcome scheme of their transformation would allow quadratic computations in the outsourcing scenario, but with a blow-up of ciphertexts. Based on the transformation, they constructed a two-server scheme for delegating quadratic computations on the outsourced data, where the blow-up in ciphertext/communication is avoided in a clever way. In their scheme, each data element $m_i$ is encrypted as a pair $(m_i - a_i, \text{Enc}(a_i))$ and given to the first server and the random number $a_i$ is given to the second server, where $\text{Enc}$ is any linearly homomorphic encryption scheme. The computation of $m_1m_2$ is done by the first server computing a ciphertext $c = \text{Enc}((m_1 - a_1)(m_2 - a_2) + a_1(m_2 - a_2) + a_2(m_1 - a_1))$ and the second server computing $a_1a_2$. The client learns $m_1m_2$ by computing $\text{Dec}(c) + a_1a_2$. The privacy of data is achieved by assuming that $\text{Enc}$ is semantically secure and the two servers do not collude with each other.

While Catalano and Fiore’s two-server scheme [7] allows...
one to delegate quadratic computations using LHE and in a succinct manner, the server-side computations however can be slow provided that the number \( n \) of data elements \( m_1, \ldots, m_n \) is large. For example, when \( n = 10^3 \) and the \( Enc \) is chosen as the fast Paillier’s encryption [20] the server-side computations for a quadratic function may require as much as 191 seconds. The waiting time could be a main measure of the clouds’ service quality and a poor quality would discourage the client from actually using the service.

Neither could the client trust the cloud services by simply storing sensitive information in clear on their servers, nor the client could trust these services by simply accepting their computation results. After all, the cloud services have the financial incentive to run an extremely fast but incorrect computation, in order to free up valuable computing time for other transactions. How to enforce the integrity of the server-side computations is also among the top security concerns in outsourcing computation. The problem of enforcing server-side computations’ integrity has been extensively studied under the name of securely outsourcing computation and realized with verifiable computation [3], [9], [10], [11], homomorphic message authenticators [8], [13], [19], and many other primitives [3], [12], [16], [22]. In Catalano and Fiore’s two-server scheme [7], each server is completely trusted to perform the specified computations correctly. However, a dishonest server may easily change the client’s output by sending back an arbitrarily chosen result (an LHE ciphertext in the first server and a ring element in the second server).

We consider the long waiting time of service and the lack of integrity of server-side computations as two main drawbacks of Catalano and Fiore [7]. It is an interesting problem to devise delegation of computation schemes with both practically fast server-side computations and integrity of server-side computations.

### 1.1 Our Contributions

In this paper, we introduce a model called **two-server delegation of computation on label-encrypted data (2S-DCLED)**, in order to provide a solution to the problem as above in the scenario of outsourcing computations. The idea of associating data elements with labels has been used in [1] to build a labeled homomorphic encryption, which supports the quadratic homomorphic computations on ciphertexts and resolves the compactness issues of [2]. Our model is obtained by integrating this idea into the two-server delegation of computation on encrypted data model of [7].

In the new 2S-DCLED model we proposed two schemes for delegating quadratic computations of the outsourced data on two non-communicating servers. Both schemes keep the client’s data private from each individual server under the mild assumption that PRFs exist. Comparing with [7], the server-side computations in our first scheme is \( \geq 2200 \) times faster, which significantly reduces the waiting time of service. Our second scheme adds integrity of server-side computations to the first scheme by using the homomorphic MAC of [9], at the price of slightly slowing the server-side computations. For every integer \( d > 2 \), we also extend the model of 2S-DCLED to the model of **\( d \)-server delegation of computation on label-encrypted data (dS-DCLED)**. We devise dS-DCLED schemes that enable the delegation of degree-\( d \) computations on the outsourced data by using \( d \) non-communicating servers.

### 1.2 Our Techniques

Our design starts from accelerating the server-side computation in the 2S-DCED scheme of [7]. Our implementation of the scheme [7] shows that the most time-consuming part of the server-side computations in [7] consists of the homomorphic computations over \( \{ (m_i - a_i, Enc(a_i)) \}_{i=1}^n \), which are done by the first server and require a large amount of public-key operations such as exponentiations modulo a large integer. Our basic idea of accelerating sever-side computations is based on removing the dependence on Enc, the linearly homomorphic encryption scheme. In [7] any quadratic computation of the form \( m_1m_2 \) was decomposed as

\[
m_1m_2 = (m_1 - a_1)(m_2 - a_2) + a_1(m_2 - a_2) + a_2(m_1 - a_1) + a_1a_2.
\]

The first server is given \( (m_1 - a_1, Enc(a_1)), (m_2 - a_2, Enc(a_2)) \) and responsible to compute a ciphertext \( c \) of \( (m_1 - a_1)(m_2 - a_2) + a_1(m_2 - a_2) + a_2(m_1 - a_1) \); the second server is given \( a_1, a_2 \) and responsible to compute \( a_1a_2 \). The reconstruction is done by computing \( Dec(c) + a_1a_2 \). The privacy of data is based on the assumption that Enc is semantically secure and the two servers do not collude with each other.

In our design, the \( Enc \) will be removed in order to accelerate the server-side computations. As a consequence, the first server is no longer able to include \( a_1(m_2 - a_2) + a_2(m_1 - a_1) \) in \( c \) and the client will not be able to recover \( m_1m_2 \) merely from \( c = Enc((m_1 - a_1)(m_2 - a_2)) \) and \( a_1a_2 \), as \( m_1, m_2 \) are both unknown to the client. To bypass this technical difficulty, we offload the linear computations such as \( a_1m_2 + a_2m_1 \) to the second server such that together the results from both servers would enable the client to remove these terms. Our key observation is a new decomposition of quadratic computations of the form \( m_1m_2 \) as below

\[
m_1m_2 = (m_1 - a_1)(m_2 - a_2) - (a_1 - b_1)(a_2 - b_2) + a_1(m_2 - b_2) + a_2(m_1 - b_1) + b_1b_2,
\]

where \( a_1, a_2, b_1, b_2 \) can be any elements from the domain of data elements. This decomposition allows us to preserve the privacy of \( m_1, m_2 \) against each individual server by sending \( (m_1 - a_1, a_1 - b_1), (m_2 - a_2, a_2 - b_2) \) to the first server and sending \( (m_1 - b_1, a_1), (m_2 - b_2, a_2) \) to the second server. If we instruct the first server to compute \( c_1 = (m_1 - a_1)(m_2 - a_2) - (a_1 - b_1)(a_2 - b_2) \) and the second server to compute \( c_2 = a_1(m_2 - b_2) + a_2(m_1 - b_1) \), then the value of \( m_1m_2 \) would be easily recovered as \( c_1 + c_2 + b_1b_2 \), where \( b_1b_2 \) can be computed on the client’s local devices. Quite different from [7], the privacy of data in our design is based on the assumption that the two servers do not collude. This is because the linearly homomorphic encryption Enc is no longer used. The server-side computation will be significantly accelerated as well since no public-key operations are involved. These improvements are not obtained at no price. In order to recover \( m_1m_2 \), the client in
our design has to compute $b_1 b_2$ locally, which will slow the client-side computation. Nevertheless we shall show with experiments that for moderately large data set size $n$, the client-side computing cost in our design is still much lower than [7]. Our client has to remember the random numbers $b_1, b_2, \ldots, b_n$. While this is not very satisfactory, especially when the client’s device has very limited storage capacity, we deal with the difficulty by associating each data element $m_i$ with a label $\tau_i$ and generate $b_i$ as a pseudorandom value $F_K(\tau_i)$, where $F$ is a PRF. In such a way we obtained a 2S-DCLED scheme where the data privacy is simply based on the mild assumption that PRFs exist and the two servers do not collude, the server-side computations are significantly faster, and the client-side computations are also faster than [7] when the size of data is moderate. Although the client-side computation is even slower than the delegated computation, the 2S-DCLED is still meaningful as long as the client is short of storage. In fact, our schemes in this paper will be specifically designed for the storage-restricted devices.

Note that in our 2S-DCLED scheme the server-side computations have very good forms. In fact, both servers only need to perform polynomial computations of degree 2 on their stored data. More precisely, when the message space is $\mathbb{Z}_p$, the finite field of $p$ elements for a prime $p$, Catalano and Fiore [8] has proposed a homomorphic MAC that enables the client to authenticate the data elements $m_1, \ldots, m_n$ with tags $t_1, \ldots, t_n$ such that any polynomial computation over $m_1, \ldots, m_n$ can be authenticated with a similar computation over the tags $t_1, \ldots, t_n$. Let $f : \mathbb{Z}_p^m \rightarrow \mathbb{Z}_p$ be any polynomial function. In the scheme of [8], both a random field element $s$ and a key $K$ for PRF $F$ are chosen as the secret key; each data element $m_i$ is authenticated with a tag $t_i(x) = m_i + F_K(\tau_i) \cdot s \cdot x$, a univariate polynomial over $\mathbb{Z}_p$ such that $t_i(0) = m_i$ and $t_i(s) = F_K(\tau_i)$. In order to learn $f(m_1, \ldots, m_n)$, the client simply gives $f$ to the server, the server sends back $y = f(m_1, \ldots, m_n)$ and $t(x) = f(t_1(x), \ldots, t_n(x))$. The client accepts $y$ if and only if $t(0) = y$ and $t(s) = f(F_K(\tau_1), \ldots, F_K(\tau_n))$. The server-side computations in our design are quadratic polynomial computations on the stored data. By applying the homomorphic MAC of [8] between the client and each individual server, we are able to add computation integrity to the 2S-DCLED scheme and obtain a scheme with with computation integrity, which is called 2S-VDCLED. While the data privacy is not changed, the computation integrity is also based on the sole assumption that PRFs exist. The additional price of adding integrity is that the client-side computation will be slightly slowed. However, we stress that, for moderately large data set size, our client-side computing cost is still much faster than [7], a scheme without integrity.

1.3 Evaluations and Comparisons

Both the 2S-DCED scheme of [7] and our 2S-DCLED/2S-VDCLED schemes allow the delegation of quadratic computations over outsourced data. In [7] the privacy of data is based on the assumption that the underlying LHE is semantically secure and the two servers do not collude. Our data privacy is based on the weaker assumption that PRFs exist and the two servers do not collude. While the 2S-DCED and 2S-DCLED schemes provide no computation integrity, our 2S-VDCLED scheme can prevent the client from accepting a wrong result. While the servers of [7] have to do a large number of exponentiations modulo large integers, our server-side computations only involve multiplications over much smaller fields and are much faster. In terms of client-side computations, although our schemes are not asymptotically better, they are still much faster when the size of data is moderate.

It is possible to extend [7] to support the computation of degree-3 polynomials. However, the server-side computational cost will increase sharply. One cannot use 2S-DCED to evaluate functions of degree $\geq 4$. In contrast, by using more servers we can outsource the computation of functions of arbitrary degrees.

1.4 Related Works

Barbosa et al. [1] constructed a labeled homomorphic encryption scheme which allows a client to delegate quadratic computations on the outsourced encrypted data to a single server. Their schemes neither support verification of servers’ computations nor allow degree-3 computations on the outsourced data. Catalano and Fiore [8] proposed a homomorphic MAC scheme that enables the client to authenticate the data elements $m_1, \ldots, m_n$ with tags $t_1, \ldots, t_n$ such that any polynomial computation over $m_1, \ldots, m_n$ can be authenticated with a similar computation over the tags $t_1, \ldots, t_n$. Their schemes cannot keep the outsourced data private. Zhang et al. [23] proposed a verifiable local computation model where the client can privately outsource data elements to cloud servers and later verify computations on any portion of the outsourced data. Their schemes satisfy our security and efficiency requirements, but require at least $d + 1$ non-communicating servers in order to compute degree-$d$ functions. Tran et al. [26] proposed two single-server schemes based on homomorphic MACs and do not rely on FHE. One of their schemes supports the computation of quadratic functions on outsourced data. Unfortunately, it has been broken [27].

1.5 Application

Disease Diagnosis. In this application, a hospital records the patient’s physical examination data for the doctor to diagnose. However, storing data will consume a lot of resources. It is difficult for a hospital to have a machine that can process function on numerous data. Then, it is feasible for hospital to store the data on cloud servers and perform desired computations on the outsourced data. Our scheme can solve these problems, and the Fig. 2 shows the process.

Data Analysis. National Climatic Data Center(NCDC) has many sensors for temperature and pressure, which are distributed throughout the world. The sensor transmits the data to the NCDC at various times. The NCDC stores the data for sale to other users who are individuals or institutions that need to use the data for research. In this scenario, there are big flaws in data collection and sales. NCDC stored at least 13.4 PB of data whose maintenance will consume a lot of resources. The users need to download all the data used in the research, and the communication cost is high. In order to solve these problems, we propose a new scheme.
whose core idea is to encrypt the data and outsource it to the server provider. The Fig. 2 presents problems solving process.

Neural Networks. Applying neural networks to a problem which involves sensitive data requires accurate predictions and maintaining data privacy and security. [17] have solve this problem, they approximate these non-linear functions using low-degree polynomials, such that the modified neural network can be evaluated using a FHE [14]. Compared with FHE, our scheme evaluate neural networks over private and maintaining data privacy and security. [17] have solve this problem, they approximate these non-linear functions using low-degree polynomials, such that the modified neural network can be evaluated using a FHE [14]. Compared with FHE, our scheme evaluate neural networks over private and maintaining data privacy and security.

Moments. In mechanics and statistics, a moment is a specific quantitative measure of the shape of a function. Since the \(d\)-th moment is computable by a degree-\(d\) polynomials, our scheme can compute the \(d\)-th moment by \(d\) servers.

Polynomials with Hidden Coefficients. The clients using Shamir secret sharing [25] can hide the coefficients of the monomials in \(f\), and turning \(f\) into a degree-\((d + 1)\) polynomial \(f'\). Furthermore, the clients can hide the monomials in \(f\) by Shamir secret sharing the coefficients in \(f\) of all monomials of degree at most \(d\).

1.6 Organization

In Section 2 we formally define the model of two-server delegation of computation on label-encrypted data; In Section 3 we present a specific construction of 2S-DCLED scheme; Section 4 contains a 2S-DCLED scheme that also satisfies the unforgeability property; In Section 5 we extend the model of 2S-DCLED to dS-DCLED scheme for any integer \(d \geq 2\). In Section 6 we implement the 2-server schemes of Section 3 and 4, and compare them with the 2-server schemes from [7]. Finally, Section 7 contains our concluding remarks.

2 Preliminaries

Notation. We denote with \(\lambda \in \mathbb{N}\) a security parameter, and with \(\text{poly}(\lambda)\) any function bounded by a polynomial in \(\lambda\). We say that a function \(\epsilon\) is negligible if it vanishes faster than the inverse of any polynomial in \(\lambda\). We use \(\text{PPT}\) for probabilistic polynomial time. If \(S\) is a set, \(x \leftarrow S\) denotes selecting \(x\) uniformly at random from \(S\). If \(\mathcal{A}\) is a probabilistic algorithm, \(x \leftarrow \mathcal{A}(\cdot)\) denotes the process of running \(\mathcal{A}\) on some appropriate input and assigning its output to \(x\). For a positive integer \(n\), we denote by \([n]\) the set \(\{1, \ldots, n\}\). Let \(X, Y\) be two random variables over a finite set \(\mathcal{U}\). We define the statistical distance between \(X\) and \(Y\) as

\[
\text{SD}[X, Y] = \frac{1}{2} \sum_{u \in \mathcal{U}} |\Pr[X = u] - \Pr[Y = u]|.
\]

2.1 Labeled Programs

A labeled program \([1]\) \(\mathcal{P}\) is a tuple \((f, \tau_1, \ldots, \tau_n)\) such that \(f : \mathcal{M}^n \rightarrow \mathcal{M}\) is an \(n\)-ary function over the message space \(\mathcal{M}\), and each label \(\tau_i \in \{0, 1\}^*\) uniquely identifies the \(i\)-th input of \(f\). Composition of labeled programs works as follows. Given labeled programs \(\mathcal{P}_1, \ldots, \mathcal{P}_t\) and a function \(g : \mathcal{M}^t \rightarrow \mathcal{M}\), the composed program \(\mathcal{P}^*\) is obtained by evaluating \(g\) on the outputs of \(\mathcal{P}_1, \ldots, \mathcal{P}_t\). Such a program is denoted as \(\mathcal{P}^* = g(\mathcal{P}_1, \ldots, \mathcal{P}_t)\). The (labeled) inputs of \(\mathcal{P}^*\) are all of the distinct labeled inputs of \(\mathcal{P}_1, \ldots, \mathcal{P}_t\) (all inputs sharing the same label are considered as a single input to the new program). Let \(f_{id} : \mathcal{M} \rightarrow \mathcal{M}\) be the canonical identity function and let \(\tau \in \{0, 1\}^*\) be a label. We denote by \(\mathcal{L}_\tau = (f_{id}, \tau)\) the identity program for input label \(\tau\). With this notation, any labeled program \(\mathcal{P} = (f, \tau_1, \ldots, \tau_n)\) can be expressed as the composition of \(n\) identity programs, i.e., \(\mathcal{P} = f(\mathcal{L}_{\tau_1}, \ldots, \mathcal{L}_{\tau_n})\).
2.2 Two-Server Delegation of Computation on Label-Encrypted Data

A two-server delegation of computation on label-encrypted data (2S-DCLED, for short) scheme is a communication protocol between a client and two non-communicating servers. It allows the client to encrypt any data item as two ciphertexts, one for each server, and then outsource the computation of the program to the servers. Each server performs a computation of the program on its ciphertexts and returns a partial result. The client can reconstruct the output of the program. The encryption should keep each individual server from learning any information about the data items. Formally, a two-server delegation of computation on label-encrypted data scheme 2S-DCLED=(KeyGen, Enc, Eval1, Eval2, Dec) consists of the following algorithms:

25.KeyGen(λ): This is a key generation algorithm. It takes the security parameter λ as input and produces a secret key sk and a public key pk.

25.Enc(sk, τ, m): This is an encryption algorithm. It takes the secret key sk, any message m ∈ M and its label τ as input, and outputs two ciphertexts C(1) and C(2).

25.Evali(pk, P, C1, ..., Cn): This is the i-th (i ∈ {1, 2}) evaluation algorithm. It takes the public key pk, a labeled program P = (f, τ1, ..., τn), and n ciphertexts C1, ..., Cn (labeled by τ1, ..., τn, respectively) as input. It outputs a ciphertext C(i).

25.Dec(sk, P, C(1), C(2)): This is a decryption algorithm. It takes the secret key sk, a labeled program P, and two ciphertexts C(1), C(2) as input, and outputs a message m ∈ M.

In our model, the client will run 25.KeyGen(λ) and generate the secret key sk and the public key pk. The client runs 25.Enc(sk, τ, m) to encrypt any message m and upload the label-encrypted data C(1), C(2) to the two servers respectively. In order to compute a function f on the data items with labels τ1, ..., τn, the client simply sends the program P = (f, τ1, ..., τn) to the servers. For every i ∈ {1, 2}, the i-th server runs 25.Evali to compute a partial result C(i) for the client. Finally, the client runs 25.Dec(sk, P, C(1), C(2)) to get the value of f(m1, ..., mn).

A 2S-DCLED scheme should satisfy the following properties: correctness, succinctness, semantic security and context hiding.

Informally, the correctness property requires that whenever the algorithms 25.KeyGen, 25.Enc, 25.Eval1, 25.Eval2 and 25.Dec are performed correctly, then the client should be able to get the correct value of f(m1, ..., mn).

Definition 1. (Correctness) The scheme 2S-DCLED is said to correctly evaluate a function family F if for all honestly generated keys (sk, pk)←25.KeyGen(λ), for all function f ∈ F, for all labels τ1, ..., τn ∈ {0, 1}*, for all messages m1, ..., mn ∈ M, for all ciphertexts C(1), C(2) ← 25.Enc(sk, m1, ..., mn) (where i = 1, 2, ..., n), we have that 25.Dec(pk, C(1), C(2), m1, ..., mn) = f(m1, ..., mn).

Informally, the succinctness property requires that the size of every ciphertext should be bounded by some fixed polynomial in the security parameter, which is independent of the size of the function.

Definition 2. (Succinctness) The 2S-DCLED is said to succinctly evaluate a function family F if there is a fixed polynomial p(·) such that every honestly generated ciphertext (output of either Enc or Evali) has size (in bits) p(λ).

The two-server delegation of computation on encrypted data scheme [7] (2S-DLED, for short) is said to compactly evaluate F if the running time of decryption is bounded by a fixed polynomial in λ, which is independent of f. Although our succinctness property is weaker than the compactness property of [7], it is especially meaningful when the client is short of communication bandwidth.

Informally, the semantic security requires that as long as the two servers do not collude with each other, each individual server cannot learn any information about the encrypted data items.

Definition 3. (Semantic Security) The semantic security of 2S-DCLED is defined with the following security game Exp2S-DCLED,A(λ) between a challenger and the PPT adversary A, where A is either the first server or the second server.

Setup. The challenger runs 25.KeyGen(λ) to obtain a pair (sk, pk) of secret key and public key. It gives the public key pk to A, and keeps the secret key sk. It also initializes a list T = ∅ for tracking the queries from A.

Queries. The adversary A adaptively issues encryption queries to the challenger, each of the form (τ, m) where τ ∈ {0, 1} and m ∈ M. The challenger then proceeds as follows: If τ /∈ T, the challenger computes (C(1), C(2)) ← 25.Enc(sk, τ, m), updates the list T = T ∪ {τ}. If A is the first server, the challenger gives C(1) to A, otherwise it gives C(2) to A. If τ ∈ T, the challenger rejects the query.

Challenge. The adversary A submits a label τ̂ ∈ {0, 1} and two data items m0, m1 ∈ M, where τ̂ is not already in the list T. The challenger selects a random bit B ∈ {0, 1}, computes (C(1), C(2)) ← 25.Enc(sk, τ̂, mB). Same as before, if A is the first server, the challenger gives C(1) to A, otherwise it gives C(2) to A.

Output. The adversary A outputs B’ representing its guess for B. If A wins the game if B’ = B.

The advantage Adv2S-DCLED,A(λ) of the adversary A in this game is defined as |Pr[B’ = B] − 1/2|, where the probability is taken over the random bits used by the challenger and the adversary A. We say that the 2S-DCLED is semantically secure if for any PPT adversary A it holds Adv2S-DCLED,A(λ) = negl(λ).

The context hiding property requires that a receiver computing m ← 25.Dec(sk, P, C(1), C(2)) should not be able to learn any additional information about the data m1, ..., mn, except what implied by m = f(m1, ..., mn). Our context hiding property will be defined in a computational setting and different from that of [7]. It is meaningful as the receiver is computationally bounded.

Definition 4. (Context Hiding) We say that a 2S-DCLED scheme satisfies context hiding for function family F if there exists a PPT simulator Sim such that the following holds. For any λ ∈ N, any keys (sk, pk)←25.KeyGen(λ), any function f ∈ F with n inputs, any messages m1, ..., mn ∈ M, any labels τ1, ..., τn ∈ {0, 1}*, if (C(1), C(2)) ← 25.Enc(sk, m1, ..., mn) for
every $i \in [n]$ and $C^{(i)} = 2S.Eval(\{pk, C^{(1)}_i, ..., C^{(n)}_i\}$ for $i = 1, 2$, then $Sim(1^\lambda, sk, P, m)$ is computationally indistinguishable from $(C^{(1)}, C^{(2)})$.

2.3 Two-Server Verifiable Delegation of Computation on Label-Encrypted Data

Our definition of 2S-DCLED has a verifiable version called two-server verifiable delegation of computation on label-encrypted data (2S-VDLED), which additionally allows the client to verify the servers’ results before actually doing the decryption. Such a scheme is defined and constructed such that no dishonest server should be able to persuade the client to accept and output a wrong value for the outsourced computation. Formally, a two-server verifiable delegation of computation on label-encrypted data scheme 2S-VDLED consists of the following algorithms:

2V.KeyGen$(1^\lambda)$: This is a key generation algorithm. It takes the security parameter $\lambda$ as input and produces a secret key $sk$ and a public key $pk$.

2V.Enc$(sk, \tau, m)$: This is an encryption algorithm. It takes the secret key $sk$, any message $m \in M$ and its label $\tau$ as input, and outputs two ciphertexts $C^{(1)}$ and $C^{(2)}$.

2V.Eval$(pk, C^{(1)}_i, ..., C^{(n)}_i)$: This is the $i$-th ($i \in \{1, 2\}$) evaluation algorithm. It takes the public key $pk$, a labeled program $P = (f, \tau_1, ..., \tau_n)$, and $n$ ciphertexts $C^{(1)}_i, ..., C^{(n)}_i$ (labeled by $\tau_1, ..., \tau_n$, respectively) as input. It outputs a ciphertext $C^{(i)}$.

2V.Dec$(sk, P, C^{(1)}_i, C^{(2)}_i)$: This is a decryption algorithm. It takes the secret key $sk$, a labeled program $P$, and two ciphertexts $C^{(1)}_i, C^{(2)}_i$ as input, and verifies the correctness of $C^{(1)}_i, C^{(2)}_i$. If both ciphertexts are correct, it decrypts and outputs a value $m \in M$. Otherwise, it outputs $\perp$ to show decryption failure.

We require 2S-VDLED to satisfy the properties of correctness, succinctness, semantic security and context hiding. The definitions of these properties for 2S-VDLED are similar to those for 2S-DCLED and omitted from here. An additional property that should be satisfied by 2S-VDLED is unforgeability, which informally requires that no malicious server should be able to provide wrong responses and persuade the client to output a wrong value.

Definition 5. (Unforgeability) The unforgeability of the 2S-VDLED scheme is defined with the following security game $\text{Exp}_{\text{2S-VDLED,}_A(\lambda)}$ between a challenger and a PPT adversary $A$, which either plays the role of a malicious first server or a malicious second server:

Setup. The challenger runs 2V.KeyGen$(1^\lambda)$ to obtain a pair $(sk, pk)$ of secret key and public key. It gives the public key $pk$ to $A$, and keeps the secret key $sk$. It also initializes a list $T = \emptyset$ for tracking the queries from $A$.

Ciphertext Queries. The adversary $A$ adaptively queries for the ciphertexts on the pairs of label and message of its choice. Given a query $(\tau, m)$ where $\tau \in \{0, 1\}^*$ and $m \in M$, the challenger performs the following: If $\tau \notin T$, the challenger computes $(C^{(1)}, C^{(2)}) \leftarrow 2V.Enc(sk, \tau, m)$, updates the list $T = T \cup \{\tau\}$. If $A$ plays the role of a malicious first server, the challenger gives $C^{(1)}$ to $A$, otherwise it gives $C^{(2)}$ to $A$.

Verification queries. The adversary $A$ adaptively issues verification queries. Let $(P, C^{(1)})$ or $(P, C^{(2)})$ be a query from $A$, where $P = (f, \tau_1, ..., \tau_n)$. Based on the types of the query, the challenger proceeds as follows.

Type 1: There exists an index $i \in [n]$ such that $\tau_i \notin T$, i.e., at least one label has not been queried. If $A$ plays the role of a malicious first server and queries with $(P, C^{(1)})$, the challenger sets $C^{(2)} = 0$ and responds with the output of 2V.Dec$(sk, P, C^{(1)}, C^{(2)})$. If $A$ plays the role of a malicious second server and queries with $(P, C^{(2)})$, the challenger sets $C^{(1)} = 0$ and responds with the output of 2V.Dec$(sk, P, C^{(1)}, C^{(2)})$.

Type 2: $T$ contains all the labels $\tau_1, ..., \tau_n$. If $A$ plays the role of a malicious first server and queries with $(P, C^{(1)})$, the challenger executes 2V.Enc and 2V.Eval2 to compute the $C^{(2)}$ and responds with the output of 2V.Dec$(sk, P, C^{(1)}, C^{(2)})$. If $A$ plays the role of a malicious second server, the challenger executes 2V.Enc and 2V.Eval1 to compute the $C^{(1)}$ and responds with the output of 2V.Dec$(sk, P, C^{(1)}, C^{(2)})$.

Output. $A$ outputs a forgery ciphertext $\tilde{C}^{(1)}$ or $\tilde{C}^{(2)}$ and a labeled program $P = (f, \tilde{\tau}_1, ..., \tilde{\tau}_n)$. The challenger runs the algorithm 2V.Dec$(sk, P, C^{(1)}, C^{(2)})$ to produce an output $\tilde{m}$. $A$ wins the game if $\tilde{m} \neq \perp$ and any of the following holds:

Type 1 forgery: There exists an index $i \in [n]$ such that $\tilde{\tau}_i \notin T$, i.e., at least one label $\tilde{\tau}_i$ has not been queried in the game.

Type 2 forgery: $T$ contains all of the labels $\tilde{\tau}_1, ..., \tilde{\tau}_n$ for the data items $m_1, ..., m_n$, and $\tilde{m} \neq \tilde{f}(m_1, ..., m_n)$, i.e., $\tilde{m}$ is not the correct output of program $P$ when executed on $(m_1, ..., m_n)$.

The advantage $\text{Adv}_{\text{2S-VDLED,}_A(\lambda)}$ of $A$ in this game is defined as the probability that $A$ wins. The scheme is said to be existentially unforgeable under adaptive chosen message and query verification attack, if for all PPT adversaries $A$, $\text{Adv}_{\text{2S-VDLED,}_A(\lambda)} = \text{negl}(\lambda)$.

Remark 1. In security game of Definition 5 the adversary $A$ can pose a verification query of the form $(P = (f, \tilde{\tau}_1, ..., \tilde{\tau}_n), C^{(1)})$ or $(P = (f, \tilde{\tau}_1, ..., \tilde{\tau}_n), C^{(2)})$. $A$ can also terminate the verification queries phase if the response by the challenger is not $\perp$ and any of the two types of forgeries happens.

Remark 2. In our treatment of type $i (i = 1, 2)$ queries, if the adversary $A$ plays the role of server $i$, then the ciphertext $C^{(3-i)}$ will be set to 0 and used for executing 2V.Dec$(sk, P, C^{(1)}, C^{(2)})$. In fact, it is not always possible to extract the message $m$ encrypted in $C^{(i)}$, as the $C^{(i)}$ is chosen by $A$ in a malicious way and possibly not well-formed. Our 2S-VDLED schemes will verify each server’s response separately. As a result, it does not matter which $C^{(3-i)}$ will be used in decryption. By default, we set $C^{(3-i)} = 0$.

3 A Construction of 2S-DCLED

In this section we present a construction of two server delegation of computation on label-encrypted data scheme that supports the evaluation of quadratic polynomials on outsourced data. In this scheme, the message space $M = \mathbb{Z}_p$, where $p$ is a $\lambda$-bit prime. Without loss of generality, we suppose that

$$\hat{f}(x_1, ..., x_n) = \sum_{i,j \in [n]} \alpha_{i,j} x_i x_j + \sum_{k \in [n]} \beta_k x_k + \gamma$$

is the quadratic polynomial that will be computed in our scheme, where $\{\alpha_{i,j}\}_{i,j \in [n]}, \{\beta_k\}_{k \in [n]}, \gamma$ are all coefficients
of $f$ and belong to $\mathbb{Z}_p$. We encrypt a message $m_1$ via a PRF $F : \mathcal{K} \times \{0,1\}^* \rightarrow \mathbb{Z}_p$ and a label $\tau$, as $m_1 - a_1$ or $m_1 - b_1$, where $a_1 = F_K(\tau||0)$, $b_1 = F_K(\tau||1)$. Since we use the label of each message to encrypt the message, we call the resulting ciphertext label-encrypted data.

The computation of any quadratic term $m_1m_2$ will be based on the following mathematical formula:

$$m_1m_2 = (m_1 - a_1)(m_2 - a_2) - (a_1 - b_1)(a_2 - b_2) + a_1(m_2 - b_2) + a_2(m_1 - b_1) + b_1b_2.$$

In our scheme, the client will send $(m_1 - a_1, a_1 - b_1, (m_2 - a_2, a_2 - b_2)$ to the first server and ask the first server to compute $C^{(1)} = (m_1 - a_1)(m_2 - a_2) - (a_1 - b_1)(a_2 - b_2)$. The client will send $(m_1 - b_1, a_1)$, $(m_2 - b_2, a_2)$ to the second server and ask the second server to compute $C^{(2)} = a_1(m_2 - b_2) + a_2(m_1 - b_1)$. Finally, the client can simply compute $C^{(1)} + C^{(2)} + b_1b_2$ to learn $m_1m_2$. In our scheme, the numbers $a_1, a_2, b_1, b_2$ will be pseudorandom values generated with a PRF. As a result, the data on each server will be pseudorandom and our scheme will be semantically secure under Definition 4. In order to compute $m_1 + m_2$, the client simply asks the second server to return $C^{(2)} = m_1 - b_1 + m_2 - b_2$ and outputs $C^{(1)} + C^{(2)} + (b_1 + b_2)$, where $C^{(1)} = 0$ as the first server is idle.

As demonstrated above, in our 2S-DCLED scheme the first server will be responsible to compute the quadratic terms of $f(m_1, m_2, \ldots, m_n)$, the second server will be responsible to compute the linear terms of $f(m_1, m_2, \ldots, m_n)$, and finally the client will be able to extract the value of $f(m_1, m_2, \ldots, m_n)$ by computing $C^{(1)} + C^{(2)} + f(b_1, \ldots, b_n)$. Throughout the process, the client only learns some random values $\{b_i\}_{i=1}^n$ and the output $f(m_1, \ldots, m_n)$, but no information about $\{m_i\}_{i=1}^n$. Our 2S-DCLED scheme can be detailed as follows:

2S.KeyGen(1$^\lambda$): Let $p$ be a $\lambda$-bit prime number. Choose a random seed $K \leftarrow \mathcal{K}$ for the PRF $F : \mathcal{K} \times \{0,1\}^* \rightarrow \mathcal{M}$. Output the secret key $sk=K$ and the public key $pk=p$. The pk implicitly defines the message space $\mathcal{M} = \mathbb{Z}_p$.

2S.Enc(sk, $\tau, m$): Given the secret key sk $= K$, the message $m \in \mathcal{M}$, and the label $\tau \in \{0,1\}^*$, compute $a = F_K(\tau||0)$ and $b = F_K(\tau||1)$, and output $C^{(1)} = (m-a,a-b)$ and $C^{(2)} = (m-b,a)$.

2S.Eval1(pk, $P, C^{(1)}_1, \ldots, C^{(1)}_n$): Given the public key pk, a labeled program $P = (f, \tau_1, \ldots, \tau_n)$ and the ciphertexts $C^{(1)}_1, \ldots, C^{(1)}_n$ with labels $\tau_1, \ldots, \tau_n$ output

$$C^{(1)} = \sum_{i,j \in [n]} \alpha_{i,j}[(m_i - a_i)(m_j - a_j) - (a_i - b_i)(a_j - b_j)].$$

2S.Eval2(pk, $P, C^{(2)}_1, \ldots, C^{(2)}_n$): Given the public key pk, a labeled program and the ciphertexts $C^{(1)}_1, \ldots, C^{(1)}_n$ with labels $\tau_1, \ldots, \tau_n$, output

$$C^{(2)} = \sum_{i,j \in [n]} \alpha_{i,j}[a_j(m_i - b_i) + a_i(m_j - b_j)] + \sum_{k \in [n]} \beta_k(m_k - b_k).$$

2S.Dec(sk, $P, C^{(1)}_1, C^{(2)}_1$): Given the secret key sk, a labeled program $P = (f, \tau_1, \ldots, \tau_n)$, and the ciphertexts $C^{(1)}_1$ and $C^{(2)}_1$, compute $b_i \leftarrow F_K(\tau_i||1)$ for each $i \in [n]$, compute $b = f(b_1, \ldots, b_n)$, and output $m = C^{(1)} + C^{(2)} + b$.

**Correctness.** The correctness of 2S-DCLED requires that the algorithm 2S.Dec always outputs the correct value of the delegated computation, if the scheme is faithfully executed.

**Theorem 1.** The proposed 2S-DCLED scheme is correct.

**Proof.** Let $m = C^{(1)} + C^{(2)} + b$ be the output of 2S.Dec. If the scheme was faithfully executed, then we have

$$m = \sum_{i,j \in [n]} \alpha_{i,j}[(m_i - a_i)(m_j - a_j) - (a_i - b_i)(a_j - b_j)] + \sum_{i,j \in [n]} \alpha_{i,j}[a_j(m_i - b_i) + a_i(m_j - b_j)] + \sum_{k \in [n]} \beta_k(m_k - b_k) + f(b_1, \ldots, b_n) = \sum_{i,j \in [n]} \alpha_{i,j}[(m_i - a_i)(m_j - a_j) - (a_i - b_i)(a_j - b_j)] + \sum_{i,j \in [n]} \alpha_{i,j}[a_j(m_i - b_i) + a_i(m_j - b_j)] + \sum_{k \in [n]} \beta_k(m_k - b_k) + f(b_1, \ldots, b_n) = \sum_{i,j \in [n]} \alpha_{i,j}m_i m_j + \sum_{k \in [n]} \beta_k m_k + \gamma - \sum_{i,j \in [n]} \alpha_{i,j} b_i b_j - \sum_{k \in [n]} \beta_k b_k - \gamma + f(b_1, \ldots, b_n) = f(m_1, \ldots, m_n).$$

By Definition 1, our scheme is correct.

**Semantic Security.** The semantic security requires that each server learns no information about the encrypted messages, as long as the two servers do not collude with each other.

**Theorem 2.** If $F$ is a secure PRF, then the proposed 2S-DCLED scheme is semantically secure.

**Proof.** We prove the theorem with two games $\text{Game } 0$ and $\text{Game } 1$. Let $W_0$ and $W_1$ be the events that a PPT adversary $A$ wins the semantic security game in $\text{Game } 0$ and $\text{Game } 1$, respectively.

**Game 0:** This is the security game $\text{Exp}_{\text{2S-DCLED}, A}(\lambda)$, respectively defined in Definition 3.

**Game 1:** This is the same as $\text{Game } 0$, except that the PRF $F$ is replaced by a truly random function. That is, the challenger chooses $a, b \leftarrow \mathcal{M}$ instead of computing $a = F_K(\tau||0)$ and $b = F_K(\tau||1)$ in the 2S.Enc procedure. It’s easy to see that there is a PRF adversary $B$ such that:

$$|\Pr[W_0] - \Pr[W_1]| \leq \text{Adv}_{F, B}^{\text{PRF}}(\lambda),$$

where $\text{Adv}_{F, B}^{\text{PRF}}(\lambda)$ is the advantage of $B$ winning the PRF security game.

If $A$ is the first server, $\hat{C}^{(1)} = (m_B - \hat{a},\hat{a} - \hat{b})$. Since $\hat{a}$ and $\hat{b}$ are random values in $\mathcal{M}$, $m_B - \hat{a}$ and $\hat{a} - \hat{b}$ are independently and uniformly distributed over $\mathcal{M}$. If $A$ is the second server, $\hat{C}^{(2)} = (m_B - \hat{b}, \hat{a})$. Since $\hat{a}$ and $\hat{b}$ are random values in $\mathcal{M}$, $m_B - \hat{b}$ and $\hat{a}$ are also independently
and uniformly distributed over $\mathcal{M}$. Hence, in both cases we have that
\[
\Pr[W_1] = \frac{1}{2}
\]  
Putting together equations (2), (3) and (4), we will have that
\[
\text{Adv}^{SS}_{2S-DCLED, A}(\lambda) = \Pr[W_0] - \frac{1}{2} \leq \text{Adv}^{PREF}_{E_S}(\lambda),
\]
which completes the proof.

**Context Hiding.** The context hiding property requires that the receiver running $2S.\text{Dec}(sk, \mathcal{P}, C(1), C(2))$ should learn no additional information about the data $m_1, \ldots, m_n$, except what implied by $f(m_1, \ldots, m_n)$.

**Theorem 3.** The proposed $2S$-DCLED scheme satisfies the context hiding property.

**Proof.** By Definition 5, we need to construct a simulator $\text{Sim}(1^{|\lambda|}, sk, \mathcal{P}, m)$ that takes the secret key sk, the program $\mathcal{P}$ and the scheme’s output $m$ as input such that its output is a pair $(C(1), C(2))$ of ciphertexts that is computationally indistinguishable from the servers’ responses $(C(1), C(2))$ in a real execution of the proposed scheme.

When $\deg(f) = 1$, we have that $\alpha_{i,j} = 0$ for all $i, j \in [n]$. It’s easy to see that $C(1) = 0$ and $C(2) = \sum_{k \in [n]} b_k (m_k - b_k)$ is pseudorandom over $\mathcal{M} = \mathbb{Z}_p$, where $b_k = F_K(\tau_k[1])$ for every $k \in [n]$. As $m = C(2) + f(b_1, b_2, \ldots, b_n)$, our simulator $\text{Sim}$ will output $C(1) = 0$ and $C(2) = m - f(b_1, b_2, \ldots, b_n)$. It’s easy to see that $\text{Sim}(1^{|\lambda|}, sk, \mathcal{P}, m)$ and $(C(1), C(2))$ are identically distributed, which implies that both distributions are computationally indistinguishable. When $\deg(f) = 2$, it is not hard to see that $C(1)$ is pseudorandom over $\mathcal{M}$ for every $i \in \{1, 2\}$ and $C(1) + C(2) = m - f(b_1, b_2, \ldots, b_n)$. Our simulator $\text{Sim}$ will choose $C(2) \leftarrow \mathcal{M}$ uniformly at random, compute $C(1) = m - f(b_1, b_2, \ldots, b_n) - C(2)$, and output $(C(1), C(2))$. It’s easy to see that $(C(1), C(2))$ and $(C(1), C(2))$ are computationally indistinguishable.

## 4 A Construction of 2S-VDCLED

In this section, we present a construction of two server verifiable delegation of computation on label-encrypted data scheme (2S-VDCLED) that supports the verifiable evaluation of quadratic polynomials. In this construction the message space $\mathcal{M}$ is $\mathbb{Z}_p$, where $p$ is a $\lambda$-bit prime. We define $f : \mathcal{M}^n \rightarrow \mathcal{M}$ as equation (1).

We use the homomorphic MACs [8] to achieve verification. In the homomorphic MACs of [8], the authentication tag of a message $m \in \mathcal{M}$ with label $\tau \in \{0, 1\}^*$ is a linear polynomial $y(x) \in \mathbb{Z}_p[x]$ such that $y(0) = m$ and $y(s) = r_{\tau}$, where $r_{\tau} = F_K(\tau)$. These operations are naturally homomorphic with respect to the evaluation of the polynomial at every point. In particular, if we have two tags $y^{(1)}$ and $y^{(2)}$ such that $y^{(1)}(0) = m_1$ and $y^{(2)}(0) = m_2$, then for $y = y^{(1)} + y^{(2)}$ (resp. $y = y^{(1)}y^{(2)}$) we clearly have $y(0) = m_1 + m_2$ (resp. $y(0) = m_1m_2$). The same homomorphic property holds for its evaluation at the random point $s$, i.e., $y(s) = r_{\tau} + r_{s\tau}$ (resp. $y(s) = r_{s\tau}$). By extending this argument to the evaluation of a function $f$, this allows to verify a tag $y$ for a labeled program $\mathcal{P} = (f, \tau_1, \ldots, \tau_n)$ and a message $m$, by simply checking that $m = y(0)$ and $f(r_{\tau_1}, \ldots, r_{\tau_n}) = y(s)$, where $r_{\tau_i} = F_K(\tau_i)$ for all $i \in [n]$.

We use homomorphic MACs separately for each server, to ensure that the output of each server is correct. Below is the description of our 2S-VDCLED scheme.

2V.KeyGen($1^\lambda$): Let $p$ be a $\lambda$-bit prime. Choose two random seeds $(K_1, K_2) \leftarrow \mathbb{Z}_p^*$ for a PRF $F : \mathcal{K} \times \{0, 1\}^* \rightarrow \mathcal{M}$. Choose $(s_1, s_2) \leftarrow \mathbb{Z}_p^2$. Output the secret key $sk = (K_1, K_2, s_1, s_2)$ and the public key $pk = p$. The pk implicitly defines $\mathcal{M} = \mathbb{Z}_p$.

2V.Enc($sk, \tau, m$): Given the secret key $sk = (K_1, K_2, s_1, s_2)$, proceed as follows to encrypt any message $m \in \mathbb{Z}_p$, with label $\tau \in \{0, 1\}^*$: First, compute $a = F_{K_1}(\tau[0]), b = F_{K_1}(\tau[1]), r_1 = F_{K_2}(\tau[0]), r_2 = F_{K_2}(\tau[1]), r_3 = F_{K_2}(\tau[2])$ and $r_4 = F_{K_2}(\tau[3])$. The ciphertext of $m$ consists of four polynomials $y^{(1)}(1), y^{(2)}(2), y^{(3)}(4), y^{(4)}(4)$, where $y^{(1)}(1) = (m - a + r_1 - s_1^2 - b^2), y^{(2)}(2) = (a - b) + r_2 - s_2^2, y^{(3)}(4) = (m - b) + r_3 - s_3^2, y^{(4)}(4) = a + r_4 - s_4^2$. The polynomial $y^{(1)}$ is constructed such that $y^{(1)}(0) = m - a$ and $y^{(1)}(s_1) = r_1$. The other polynomials $y^{(2)}$, $y^{(3)}$ and $y^{(4)}$ are constructed with the same idea. This algorithm outputs $C^{(1)} = (y^{(1)}, y^{(2)}), C^{(2)} = (y^{(3)}, y^{(4)})$.

2V.Eval($pk, \mathcal{P}, C^{(1)}, \ldots, C^{(n)}$): This algorithm takes the public key $pk$, a labeled program $\mathcal{P} = (f, \tau_1, \ldots, \tau_n)$ and the ciphertexts $C^{(1)}, \ldots, C^{(n)}$ (labeled by $\tau_1, \ldots, \tau_n$, respectively) as input, where $C^{(i)} = (y^{(i,1)}, y^{(i,2)})$ for every $i \in [n]$. It outputs
\[
C^{(1)} = \sum_{i,j \in [n]} \alpha_{i,j} [y^{(i,1)} y^{(j,1)} - y^{(i,2)} y^{(j,2)}],
\]
which is a quadratic polynomial in $x$ and usually represented with the field elements $y^{(1)}(1), y^{(1)}(1), y^{(2)}(1), y^{(2)}(1)$ such that $C^{(1)} = y^{(1)}(1) x + y^{(2)}(1) x^2$.

2V.Eval($pk, \mathcal{P}, C^{(1)}, \ldots, C^{(n)}$): This algorithm takes the public key $pk$, a labeled program $\mathcal{P} = (f, \tau_1, \ldots, \tau_n)$ and the ciphertexts $C^{(1)}, \ldots, C^{(n)}$ (labeled by $\tau_1, \ldots, \tau_n$, respectively) as input, where $C^{(i)} = (y^{(i,3)}, y^{(i,4)})$ for every $i \in [n]$. It outputs
\[
C^{(2)} = \sum_{i,j \in [n]} \alpha_{i,j} [y^{(i,3)} y^{(j,4)} + y^{(i,4)} y^{(j,3)}] + \sum_{k \in [n]} b_k y^{(k,3)},
\]
which is a quadratic polynomial in $x$ and usually represented with the field elements $y^{(2)}(1), y^{(2)}(1), y^{(2)}(1)$ such that $C^{(2)} = y^{(2)}(1) x + y^{(2)}(1) x^2$.
The proofs for the correctness, succinctness, semantic security and context hiding properties of the 2S-VDCL scheme are quite similar to those for our 2S-DCLED scheme and omitted here. It remains to show the unforgeability of the proposed 2S-VDCL scheme.

Unforgeability. This property requires that no adversary that plays the role of a malicious first server or the role of a malicious second server is able to persuade the client to output a wrong value for the delegated computations.

Theorem 4. Suppose that $F$ is an PRF. Then the proposed 2S-VDCL scheme is unforgeable. In particular, for any PPT adversary $A$ that makes $\leq Q$ verification queries, we have that

$$\text{Adv}^{\text{UF}}_{\text{2S-VDCL},A}(\lambda) \leq \epsilon_F + \frac{2(Q+1)}{p - 2Q}$$

where $\epsilon_F$ is an upper bound on the advantage of any PPT adversary winning the PRF security game with respect to $F$.

Proof. We define two games Game 0 and Game 1 and let $W_0, W_1$ be the events that $A$ wins in Game 0 and Game 1, respectively.

Game 0: This game is the standard security game $\text{Exp}^{\text{UF}}_{\text{2S-VDCL},A}(\lambda)$ of Definition 5. We have that

$$\Pr[W_0] = \text{Adv}^{\text{UF}}_{\text{2S-VDCL},A}(\lambda).$$

Game 1: This game is identical to Game 0, except that the PRF $F$ is replaced with a truly random function. That is, for every label $\tau$, the challenger generates $a, b, r_1, r_2, r_3, 1 \leftarrow \mathbb{Z}_p$ instead of computing $a = F_{K_1}(\tau || 0), b = F_{K_1}(\tau || 1), r_1 = F_{K_2}(\tau || 0), r_2 = F_{K_2}(\tau || 1), r_3 = F_{K_2}(\tau || 2)$ and $r_4 = F_{K_2}(\tau || 3)$. It’s trivial to see that

$$|\Pr[W_1] - \Pr[W_0]| \leq \epsilon_F.$$ 

Without loss of generality, we suppose that $A$ plays the role of a malicious first server. Then the challenger in Game 1 will work as follows.

Ciphertext Queries. The adversary submits queries $(\tau_i, m_i)$ where $\tau_i$ is the label of message $m_i$. The challenger creates a new list $T$ for tracking the queries from $A$ in the game. For the $i$-th query, if $T$ does not contain $\tau_i$, i.e., the label $\tau_i$ was never queried. The challenger responds as follows: choose $a_i, b_i, r_{i,1}, r_{i,2}, r_{i,3}, r_{i,4} \leftarrow \mathbb{Z}_p$; compute $C_i^{(1)} = ((m_i - a_i) + r_{i,1}(m_i - a_i), x_i, a_i - b_i + r_{i,2}(m_i - a_i))$; send $C_i^{(1)}$ to $A$ and update $T = T \cup (\tau_i, a_i, b_i, r_{i,1}, r_{i,2}, r_{i,3}, r_{i,4})$. If $(\tau_i, M_i) \in T$, i.e. label $\tau_i$ was previously queried, the challenger rejects the query.

Verification Queries. The adversary submits queries $(P_i, C_i^{(1)})$ where program $P_i = (f, (\tau_{i,1}, \ldots, \tau_{i,n_i}))$. The challenger responds to the $i$-th query as follows: If there is a $j \in [n_i]$ such that $(\tau_{i,j}, \ldots, \tau_{i,n_i}) \notin T$, the challenger chooses random values $a_{i,j}, b_{i,j}, r_{i,j,1}, r_{i,j,2}, r_{i,j,3}, r_{i,j,4} \leftarrow \mathbb{Z}_p$. The challenger performs $2V.Eval_{\tau_{i,j}}$ to compute $C_i^{(2)}$ and responds with the output of $\text{Dec}(\sk, P, C_i^{(1)}, C_i^{(2)})$. Eventually $A$ outputs $(P, C_i^{(1)})$, where $P = (f, \tilde{\tau}_1, \ldots, \tilde{\tau}_n)$, $C_i^{(1)} = (y_{i,1}^{(1)}, y_{i,2}^{(1)})$. The adversary $A$ wins the game if any of the two following types of forgeries occurs.

Type 1 forgery: If there exists an $i \in [n]$ such that $(\tilde{\tau}_i, \ldots, \tilde{\tau}_n) \notin T$, the challenger chooses random values $a_{i,j}, b_{i,j}, r_{i,j,1}, r_{i,j,2}, r_{i,j,3}, r_{i,j,4} \leftarrow \mathbb{Z}_p$ and retrieves the random values corresponding to the remaining labels from $T$. Let $\tilde{R}_1 = \sum_{i,j \in [n]} \alpha_{i,j}[r_{i,j,1} - r_{i,j,2}, r_{i,j,3} - r_{i,j,4}]$. The adversary wins the game if:

$$\tilde{R}_1 = y_{i,1}^{(1)} + y_{i,2}^{(1)} s_1 + y_{i,2}^{(1)} s_2.$$  (6)

Type 2 forgery: If for all $i \in [n]$ such that $(\tilde{\tau}_i, \ldots, \tilde{\tau}_n) \in T$, the challenger retrieves $(a_{i,j}, b_{i,j}, r_{i,j,1}, r_{i,j,2}, r_{i,j,3}, r_{i,j,4})$ from list $T$. Let $\tilde{b} = b(\tilde{b}_1, \ldots, \tilde{b}_n)$, $\tilde{R}_1 = \sum_{i,j \in [n]} \alpha_{i,j}[r_{i,j,1} - r_{i,j,2}, r_{i,j,3} - r_{i,j,4}] + \sum_{k \in [n]} \tilde{b}_k r_{k,3}$. Let $\tilde{m} = y_{i,1}^{(1)} + y_{i,2}^{(1)} + \tilde{b}$, the adversary wins the game if:

$$\tilde{R}_1 = y_{i,1}^{(1)} + y_{i,2}^{(1)} s_1 + y_{i,2}^{(1)} s_2.$$  (7)

and $\tilde{m} \neq \tilde{f}(m_1, \ldots, m_n)$.

We now compute the probability of $A$ winning Game 1. Let $B_i$ be the event that $A$ wins the game after $i$ verification queries. Let $Q$ be the upper bound on the number of verification queries made by $A$. We have:

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

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

Event $V$ happens (type 1 forgery): The left-hand side of (6) is a random value in $\mathbb{Z}_p$ that is independent of $A$’s view. In addition, since $s_1$ is a secret key, the probability that equation (7) holds is exactly $2/p$. Hence,

$$\Pr[B_i \cap V] = \frac{2}{p} \Pr[V].$$

Event $\neg V$ happens (type 2 forgery): In this case, $A$ uses a program $P = (f, \tilde{\tau}_1, \ldots, \tilde{\tau}_n)$ and all the labels have been posed in the previous ciphertext queries. Event $B_i$ happens if $\tilde{m} \neq \tilde{f}(m_1, \ldots, m_n)$ and equation (8) holds.

Let $C_i^{(1)}$ be the ciphertext corresponding to label $\tilde{\tau}_j$ in a previous ciphertext query, for all $j \in [n]$. Define $C_i^{(1)} = (y_{i,1}^{(1)}, y_{i,2}^{(1)}) = 2V.Eval_{\tilde{\tau}_j}(\sk, P, C_i^{(1)}\ldots, C_i^{(1)})$. Since $C_i^{(1)}$ is a valid ciphertext for $m = \tilde{f}(m_1, \ldots, m_n)$, the following relation holds:

$$\tilde{R}_1 = \sum_{i,j \in [n]} \alpha_{i,j}[r_{i,j,1} - r_{i,j,2}, r_{i,j,3} - r_{i,j,4}]$$

$$y_{i,1}^{(1)} + y_{i,2}^{(1)} s_1 + y_{i,2}^{(1)} s_2.$$  (9)

Subtracting (10) from (7), we obtain:

$$(y_{i,1}^{(1)} - y_{i,1}^{(1)}) s_1 + (y_{i,2}^{(1)} - y_{i,2}^{(1)}) s_2 = 0.$$  (11)

Since $\tilde{m} \neq \tilde{f}(m_1, \ldots, m_n)$, we know that $C_i^{(1)} \neq C_i^{(1)}$ implying that the left-hand side of (11) is a nonzero polynomial in $s_1$. Hence, in producing a valid forgery, $A$ must guess secret key $s_1$. 

\[JOURNAL OF INFORMATION SECURITY, VOL. 14, NO. 8, AUGUST 2015\]
As $s_1$ is uniformly distributed over $Z_p$, we have $\Pr[B_0 \cap \neg V] = 2/p \cdot \Pr[\neg V]$. After the first verification query, since there are at most 2 values of $s_1$ that satisfy equation (11), the number of possible values for $s_1$ becomes $\geq p - 2$. Therefore, after $i$ queries, $A$ can exclude at most $2i$ possible values of $s_1$, meaning that the number of possible values for $s_1$ is at least $p - 2i$. Thus,

$$\Pr[B_i \cap \neg V] \leq \frac{2}{p - 2i} \cdot \Pr[\neg V].$$  \hspace{1cm} (12)

From equations (9) and (13), we obtain:

$$\Pr[B_i] \leq \frac{2}{p - 2i} (\Pr[V] + \Pr[\neg V]) \leq \frac{2}{p - 2i}.$$

Finally, we have

$$\Pr[W_i] \leq \frac{2Q}{p - 2i}.$$

Putting together equations (4), (5) and (13),

$$\text{Adv}_{\text{2S-DCLED},A}(\lambda) \leq \epsilon_F + \frac{2Q}{p - 2i}.$$

Since $p \approx 2^\lambda$ and $Q$ is a polynomial of $\lambda$, $\frac{2Q}{p - 2i} = \text{negl}(\lambda)$ thus completing the proof of Theorem 4.

\section{Generalization to $d$ Servers ($d > 2$)}

Our definitions for 2S-DCLED and 2S-VDCLED can be generalized to the $d$-server case for any integer $d > 2$, which gives the models for dS-DCLED and dS-VDCLED. In this section, we show how to leverage any degree-$d$ computations using $d$ non-communicating servers.

\subsection{Basic Ideas for Constructing dS-DCLED}

It suffices to demonstrate the possibility of computing any degree-$d$ monomial $f(m_1, m_2, \ldots, m_d) = \prod_{i=1}^d m_i$ with $d$ non-communicating servers. In our construction, the client will use a PRF $F$ to generate a pseudorandom number $a_{i,j} \in M$ for every $i, j \in [d]$. For every $j \in [d]$, it stores the following data on the $j$-th server.

| Server $j$ |
|-----------------|
| $a_{1,1}, \ldots, a_{1,j-1}, m_1 - a_{1,j}, a_{1,j+1}, \ldots, a_{1,d}$ |
| $a_{2,1}, \ldots, a_{2,j-1}, m_2 - a_{2,j}, a_{2,j+1}, \ldots, a_{2,d}$ |
| $\ldots$ |
| $a_{d,1}, \ldots, a_{d,j-1}, m_d - a_{d,j}, a_{d,j+1}, \ldots, a_{d,d}$ |

In our construction, the first server will be responsible to compute $S_1 = \prod_{i=1}^d (m_i - a_{i,1})$ and set $c_1 = 1$. The second server will be responsible to eliminate the degree-$(d - 1)$ terms in $S_1$. More precisely, the second server will compute $S_2 = \sum_{i=1}^d \prod_{j=2}^d (m_j - a_{i,j}) a_{i,1}$ and set $c_2 = a_{1,1}$. The third server will be responsible to eliminate the degree-$(d - 2)$ terms in both $S_1$ and $S_2$. More precisely, it will compute $S_3 = \prod_{i=1}^d \sum_{j=2}^d (m_j - a_{i,j}) [c_2 (a_{i,1}) + c_2 a_{i,2}]$ and set $c_3 = c_2 (a_{1,1}) + c_2 a_{1,2}$. In general, for every $j > 1$, the $j$-th server will be responsible to eliminate all degree-$(d - j + 1)$ terms that arise from the computations of $S_1, \ldots, S_{j-1}$. More precisely, it will compute $S_j = \prod_{i=1}^d \sum_{j=2}^d (m_j - a_{i,j}) [c_2 \prod_{k=1}^{j-1} (a_{k,1}) + c_2 a_{i,2} \prod_{k=1}^{j-2} (a_{k,2}) + \ldots + c_2 a_{i,j-1} a_{i,j}]$. The following theorem shows that based on the servers' responses a client can reconstruct $f(m_1, m_2, \ldots, m_d)$ with limited local computations.

\textbf{Theorem 5}. Let $S_j$ be defined as above for every $j \in [d]$. Let $\mathbb{P}_d = \{i = (i_1, \ldots, i_d) : \{i_1, \ldots, i_d\} = [d]\}$ be the set of all permutations of $[d]$. Then

$$\sum_{j=1}^d S_j = \prod_{i=1}^d m_i + \sum_{i=1}^d (-a_{i,1}) + \sum_{j=2}^d \sum_{i=1}^d c_j \prod_{k=j}^d (-a_{i,k}).$$  \hspace{1cm} (14)

\textbf{Proof}. We show that $S_j$ can eliminate all degree-$(d - j + 1)$ terms in $S_1, \ldots, S_{j-1}$ for every $j \in \{2, \ldots, d\}$. In $S_1$, the degree-$(d - j + 1)$ terms coefficients $c_1' = \prod_{i=1}^d (-a_{i,1}) = -a_{1,1} \prod_{i=2}^d (-a_{i,1}) = -c_2 \prod_{i=2}^d (-a_{i,1})$. In $S_2$ for $\ell = 2, \ldots, j - 1$, the degree-$(d - j + 1)$ terms coefficients $c_2' = \prod_{i=1}^d (-a_{i,1}) = -c_2 \prod_{i=k+1}^d (-a_{i,k})$.

We have known the degree-$(d - j + 1)$ terms coefficients of $S_j$ is $c_j = c_2 \prod_{i=1}^{j-1} (-a_{i,1}) + c_2 a_{i,2} \prod_{i=1}^{j-2} (-a_{i,2}) + c_2 a_{i,3} \prod_{i=1}^{j-3} (-a_{i,3}) + \cdots + c_{j-1} a_{i,j-1} a_{i,j}$. To prove $S_j$ can eliminate all degree-$(d - j + 1)$ terms in $S_1, \ldots, S_{j-1}$ for every $j \geq 2$, we just prove $\sum_{i=1}^d c_i' + c_j = 0$. In fact,

$$\sum_{i=1}^d c_i' + c_j = \sum_{k=2}^{j-1} c_k \prod_{i=1}^d (-a_{i,1}) + \sum_{k=2}^{j-1} c_k (-a_{i,k}) \prod_{i=1}^d (-a_{i,k}) - c_j a_{i,j} \prod_{k=j}^d (-a_{i,k}).$$  \hspace{1cm} (15)

Thus, $\sum_{j=1}^d S_j$ only contains the degree-$d$ term $\prod_{i=1}^d m_i$ and the constant terms. Next, we give a concrete expression of the constant terms. In $S_1$, the constant term is
In the second experiment, we choose \( n \leq 5.2 \) Basic Ideas for Constructing

\[ \prod_{i=1}^{d} \left( -a_{i,1} \right) \]

The client’s local computation incurred by \( \prod \) may be large. To speed-up the client-side computation, we can distribute the computations of most monomials in \( \prod \) to the servers. We observe that any term \( a_{i_1,1} \prod_{i_2} a_{i_2,2} \cdots \prod_{i_d} a_{i_d,d} \) with \( \{i_1, i_2, \ldots, i_d\} \subset [d] \) will be computable by at least one of the \( d \) servers. In our construction, we will distribute any such term to one of the servers that can compute it. On the other hand, the term \( a_{i_1,1} \prod_{i_2} a_{i_2,2} \cdots \prod_{i_d} a_{i_d,d} \) is not computable by any of the \( d \) servers if and only if \( \{i_1, i_2, \ldots, i_d\} \) is a permutation of the set \([d]\). The client will be responsible to compute such terms.

5.2 Basic Ideas for Constructing \( d \)-S-VDCLED

In our \( d \)-S-DLED scheme each server performs a computation of degree \( \leq d \) over its data. By using the homomorphic MACs of \( \Phi \) one can make such computations can be made verifiable and therefore obtain a \( d \)-S-VDCLED scheme.

6 Performance Analysis

In this section, we shall implement the proposed schemes and compare with \([\mathbb{7}]\). As we are mostly interested in the practicality of all schemes, the comparisons between all schemes will be done in terms of the running time of the server-side computations, and the running time of the client-side computations. The comparison will be done with three experiments. The first experiment will compare the 2S-DLED/2S-VDCLED from Section 4 with the 2S-DCED from \([\mathbb{7}]\). The second experiment will do the same comparisons but in a scenario where a large number of computation requests occur at the same time.

6.1 Experiments Designs

We implement all of the schemes with a security parameter \( \lambda = 128 \) and in a Ubuntu 16.04 LTS 64-bit operating system with 4GB RAM and Intel \(^\circ\) Core \(^\circ\) i7-6700 3.40GHz processor. We choose the PRF \( F \) in all schemes as the standard AES with 128-bit secret key from the library OpenSSL 1.0.2g. We choose the efficient Paillier cryptosystem \([\mathbb{20}]\), whose ciphertext size is half of \([\mathbb{21}]\) and has a fast decryption algorithm, as the linearly homomorphic encryption for the 2S-DCED scheme from \([\mathbb{7}]\). We realize all large integer related mathematical computations based on the C libraries GMP and FLINT.

In the first experiment, we consider the computation of a quadratic function \( f(m_1, \ldots, m_n) = \sum_{i,j \in [n]} a_{i,j} m_i m_j + \sum_{k \in [n]} \beta_i m_k + \gamma \) on outsourced data, where the number \( n \) of data items is chosen from \( \{10, 50, 100, 500, 1000\} \). In the second experiment, we choose \( n = 500 \) and consider \( t \) simultaneous computation requests for \( t \in \{10, 100, 1000, 10000\} \). We compare between 2S-DCED, 2S-DLED and 2S-VDCLED with the average waiting time.

6.2 Experimental Results

Table 1 shows the evaluation algorithm and decryption algorithm execution times of 2S-DCED, 2S-DLED and 2S-VDCLED, where evaluation algorithm execution times is the sum of the 2S-Eval\(_1\) and 2S-Eval\(_2\). Fig. 3 shows the average waiting time of the client for different number of requests.

6.3 Comparisons

We mainly compare the running time of the 2S-DCED and our schemes 2S-DLED and 2S-VDCLED on the server-side and the client-side.

Server-Side. The servers perform the evaluation algorithms. For 2S-DCED, the server-side need to run Paillier cryptosystem and large integer multiplication and exponentiations. For our schemes, the server-side need do multiplications and additions modulo \( p \). Theoretically, in the server-side, our schemes are faster than 2S-DCED. And our first experiment also confirmed it. More precisely, the server-side running time of 2S-DCED and 2S-VDCLED can be 2200 and 600 times faster than 2S-DCED respectively.

Client-Side. The client perform the decryption algorithm. The client-side running time of 2S-DCED is dominated by the decryption of one Paillier ciphertext. For our schemes, the running time is dominated by the computation of \( f(b_1, \ldots, b_n) \). Then, the client-side running time of 2S-DCED is fixed, the client-side running time of our schemes become longer as the amount of data increases. But this is not a disadvantage. When the amount of data is bounded, the running time of our schemes will be shorter than 2S-DCED. Even if the client-side running time of our schemes are larger than 2S-DCED, their difference is small. When \( n = 1000 \), 2S-DCED is only 0.04 second faster than 2S-VDCLED, which does not have obvious advantages in practical applications. Our second experiment shows that this advantage is not significant. In the second experiment, if the client receives multiple requests at the same time, the time that 2S-DCED responds to each request will be significantly higher than our schemes. This means that the long waiting time would discourage the client from actually using the service and the numerous computations would require more charge by the client.
7 Concluding Remarks

In this paper, we proposed a multi-server model for delegating computations on label-encrypted data. We constructed both a 2S-DCLED scheme and a 2S-VDCLED scheme. The server-side computations in both schemes are much faster than the 2S-DCED scheme from [7]. The client-side computations in both schemes are faster than [7] when the size of the data is moderate. The semantic security of both schemes only depends on the mild assumption that PRFs exist. The 2S-VDCLED scheme also achieves verifiability, which was not provided in [7]. We also extend the study to d-server schemes, which can delegate degree-d computations, a functionality not provided in [7]. The complexity of our decryption algorithm depends on the size of the outsourced data. Removing or weakening this dependency is an interesting open problem for future work.

8 Acknowledgments

This work was supported by NSFC (No. 61602304) and Pujiang Talent Program (No. 16PJ1406500).

References

[1] Manuel Barbosa, Dario Catalano, Dario Fiore: Labeled Homomorphic Encryption - Scalable and Privacy-Preserving Processing of Outsourced Data. ESORICS (1) 2017: 146-166
[2] Sivasory Benabbas, Rosario Gennaro, Yeugeniy Vahliis: Verifiable Delegation of Computation over Large Datasets. CRYPTO 2011: 111-131
[3] Nir Bitansky, Ran Canetti, Alessandro Chiesa, and Eran Tromer. From extractable collision resistance to succinct non-interactive arguments of knowledge, and back again. In ITCS, 2012.
[4] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (Leveled) fully homomorphic encryption without bootstrapping. In Shafi Goldwasser, editor, ITCS, pages 309-325. ACM, 2012.
[5] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In Rafail Ostrovsky, editor, FOCS, pages 97–106. IEEE, 2011.
[6] Zvika Brakerski and Vinod Vaikuntanathan. Lattice-based FHE as secure as PKE. In Moni Naor, editor, Innovations in Theoretical Computer Science, ITCS’14, Princeton, NJ, USA, January 12-14, 2014, pages 1–12. ACM, 2014.
[7] Dario Catalano, Dario Fiore: Boosting Linearly-Homomorphic Encryption to Evaluate Quadratic Functions on Encrypted Data. IACR Cryptology ePrint Archive 2014: 813 (2014)
[8] Dario Catalano, Dario Fiore: Practical Homomorphic Message Authenticators for Arithmetic Circuits. J. Cryptology 31(1): 23-59 (2018)
[9] Kai-Min Chung, Yael Tauman Kalai, Salil P. Vadhan: Improved Delegation of Computation Using Fully Homomorphic Encryption. CRYPTO 2010: 483-501
[10] Dario Fiore, Rosario Gennaro: Publicly verifiable delegation of large polynomials and matrix computations, with applications. ACM Conference on Computer and Communications Security 2012: 501-512
[11] Rosario Gennaro, Craig Gentry, Bryan Parno: Non-interactive Verifiable Computing: Outsourcing Computation to Untrusted Workers. CRYPTO 2010: 465-482
[12] Rosario Gennaro, Craig Gentry, Bryan Parno, Mariana Raykova: Quadratic Span Programs and Succinct NIZKs without PCPs. EUROCRYPT 2013: 626-645
[13] Rosario Gennaro, Daniel Wichs: Fully Homomorphic Message Authenticators. ASIACRYPT (2) 2013: 301-320
[14] Craig Gentry: Fully homomorphic encryption using ideal lattices. STOC 2009: 169-178
[15] Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In Ran Canetti and Juan A. Garay, editors, Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part I, volume 8042 of Lecture Notes in Computer Science, pages 75–92. Springer, 2013.
[16] C. Gentry and D. Wichs. Separating succinct non-interactive arguments from all falsifiable assumptions. In STOC, 2011.
[17] Gilad-Bachrach Ran, Dowlin Nathan, Laine Nathan, Lauter Kristin, Naehrig Michael and Wernsing John (2016). CryptoNets: applying neural networks to encrypted data with high throughput and accuracy. International conference on machine learning.
[18] S. Goldwasser, S. Micali: Probabilistic encryption. J. Comput. Syst. Sci. 28(2): 270-299 (1984)
[19] Sergey Gorbunov, Vinod Vaikuntanathan, Daniel Wichs: Leveled Fully Homomorphic Signatures from Standard Lattices. STOC 2015: 469-477
[20] Joyce M, Libert B. Efficient cryptosystems from 2 k-th power residue symbols[C]//Annual International Conference on the Theory and Applications of Cryptographic Techniques. Springer, Berlin, Heidelberg, 2013: 76-92.
[21] Paillier Pascal. Public-key cryptosystems based on composite degree residuosity classes[C]//International Conference on the Theory and Applications of Cryptographic Techniques. Springer, Berlin, Heidelberg, 1999: 223-238.
[22] Bryan Parno, Jen Howell, Craig Gentry, Mariana Raykova: Pinocio: nearly practical verifiable computation. Commun. ACM 59(2): 103-112 (2016)
[23] Rivest Ronald L, Adleman Len, Dartouzos Michael L. On data banks and privacy homomorphisms. Foundations of Secure Computation, Academia Press, pages 169–179, 1978.
[24] Ronald L. Rivest, Adi Shamir, Leonard M. Adleman: A method for obtaining digital signatures and public-key cryptosystems. Commun. ACM 21(2): 120-126 (1978)
[25] Shamir Adi: How to share a secret[j]. Communications of the ACM, 1979, 22(11): 612-613.
[26] Ngoc Hieu Tran, HueHwa Pang, Robert H. Deng: Efficient Verifiable Computation of Linear and Quadratic Functions over Encrypted Data. AsiaCCS 2016: 605-616
[27] Shuaijianni Yu, Yan He, Liang Feng Zhang: Cryptanalysis of Tran-Pang-Deng Verifiable Homomorphic Encryption. ICISC 2017: 79-89
[28] Liang Feng Zhang, Reihan Safavi-Naini, Xiao Wei Liu: Verifiable local computation on distributed data. SCC@ASIACCS 2014: 3-10
[29] Federal Information Processing Standards Publications. (2001 November 26). Announcing the Advanced Encryption Standard (AES). Computer security resource center. Retrieved February 21, 2019, from: https://csrc.nist.gov/csrc/media/publications/standard (AES).
[30] iCloud leaks of celebrity photos. (2019 February 14). Retrieved February 21, 2019, from: https://en.wikipedia.org/wiki/ICloud_leaks_of_celebrity_photos