Efficient Computation Outsourcing for Inverting a Class of Homomorphic Functions

Fangguo Zhang[1], Xu Ma[1] and Shengli Liu[2]

[1]School of Information Science and Technology,
Sun Yat-Sen University, Guangzhou 510006, China
isszhfg@mail.sysu.edu.cn, maxusysu85@gmail.com
[2]Dept. of Computer Science and Engineering
Shanghai Jiao tong University, Shanghai 200240, China
slliu@sjtu.edu.cn

Abstract. The rise of cloud computing and the proliferation of mobile devices make computation outsourcing popular. However, the servers are not fully trusted, and a critical problem is the verifiability and privacy of such computations. Although some computation outsoucing schemes provided a general method, the complicated cryptographic tools involved result in great inefficiency. The existing efficient computation outsourcing schemes however aim only at a specific computation task, lacking in generality. In this paper, we show how to construct a generic outsourcing computation scheme for inverting a class of homomorphic functions with computation disequilibrium. Extensive analysis shows that many cryptographic computations fall into this category. The formal security analysis proves that our scheme satisfies verifiability, input and output privacy in information-theoretic sense. Since the construction of our scheme tactfully takes advantage of the intrinsic property of the computation task being outsourced, no public key operations are used in the scheme, thus our solution clearly outperforms the existing schemes in terms of efficiency. In addition, we instantiate our generic construction with concrete examples, and the experimental result testifies the efficiency of our construction.

Keywords: Computation outsourcing, homomorphic function, security, privacy.

1 Introduction

Computation outsourcing considers a scenario where a computationally weak client delegates computationally expensive tasks to one or more servers who have abundant computing resources, massive storage and bandwidth. Outsourcing computation to untrusted servers has received widespread attention due to the rise of cloud computing [4, 31]. Companies or individuals may buy computing power from a service, rather than purchasing and maintaining their own computing facilities. Another motivation of computation outsourcing is the proliferation of mobile devices, such as notebooks and smart phones. Due to the computation and storage limitations, sometimes it is desirable to off-load heavy computations, such as cryptographic operations, or photo manipulation, to the cloud server.

However, the servers are not fully trustworthy and sometimes the computation tasks outsourced to the cloud are so critical that it is imperative to rule out accidental errors during the computation outsourcing process. Consequently, a fundamental security requirement of computation outsourcing is that the client should be able to "efficiently" verify the correctness of computation result returned by the server. By "efficiently" it means that the verification process takes the client substantially less computation efforts than the original computation task. Another security requirement is to protect the secrecy of the input and

output of the computation task, since they may contain sensitive information of the client, such as the client's private key or medical records.

Previous research on computation outsourcing are moving toward two directions. One is the generic outsourcing model proposed in [3, 13, 15] that can evaluate any computational function $f : D \to R$ at any $x \in D$. The other is a specific outsourcing model designed for only one specific computation task, such as polynomial evaluation [27, 10], matrix multiplication [27], modular exponentiation [21, 24] or linear algebra [1]. Obviously, both of the generic model and concrete model are not fully satisfying. The generic computation outsourcing is searching for cryptographic tools and hoping that they can solve all the computation outsourcing problems once for all, but suffers from great inefficiency. On the other hand, the concrete computation outsourcing merely concentrates on one particular computational function, and its application is quite limited.

Related work. In [15], Gennaro et al. showed how to delegate arbitrary computations by increasing the client's offline complexity and public-key size based on a fully homomorphic encryption [17] and Yao's garbled circuit. In [13], Chung et al. proposed an improved generic computation outsourcing scheme without any garbled circuit, whereas the client needs pre-computation to verify the result. In [9], Barbosa and Farshim gave a modular construction of delegatable homomorphic encryption from fully homomorphic encryption, functional encryption and MAC, and showed how to build a secure computation outsouring scheme from delegatable homomorphic encryption. However, fully homomorphic encryptions are now far from being practical, hence outsouring schemes built from them have also limited applications.

In theoretical research community, lots of efforts have been devoted to the computation outsourcing of arbitrary functions. Researchers proposed to use various types of proof systems to convince the client the correctness of the computation result by the server. These works contain interactive proofs in [8, 18], efficient arguments based on probabilistically checkable proofs [22, 23], CS proofs [25] and the muggles proofs [19]. However, utilizing proofs systems in computation outsourcing schemes can only achieve the security requirement of verifiability. Input and output privacy cannot be guaranteed merely by the proof systems.

To outsource the computation of specific functions, plenty of research works have been proposed. In [21], Jakobsson proposed an outsourcing scheme for modular exponentiation, the work is further studied in [12]. Benjamin and Atallah [7] addressed the problem of secure outsourcing for widely applicable linear algebra computations. However, the proposed schemes required expensive homomorphic encryptions. Atallah and Frikken [3] further studied this problem and gave improved schemes based on the so-called weak secret hiding assumption. Benabbas et al. [10] presented the first practical computation outsourcing scheme for high degree polynomial functions based on the approach of [15]. Fiore and Gennaro further studied the problem and proposed a publicly verifiable computation outsourcing scheme. In 2011, Green et al. [16] proposed new methods for efficiently and securely outsourcing decryption of attribute-based encryption (ABE) ciphertexts. Based on this work, Parno et al. [26] showed a construction of a multi-function computation delegation scheme. In 2012, Waters [30] proposed a computation outsourcing for attribute-based encryption where the ciphertext update can be efficiently and securely outsourced to the server.

Theoretically it's possible to solve any computation outsourcing tasks using the generic computation outsourcing method. However, by focusing on specific computation the concrete schemes are always much more efficient. The question is: what property of the computation tasks implies the existence of an efficient outsourcing scheme for these computations? In this paper, we answer this question by identifying some computation task and design highly efficient outsourcing scheme for them.

Our contributions. In this paper, we present a class of homomorphic functions that are especially suitable for outsourcing its inverse evaluation. We designed a secure computation outsourcing scheme and gave an analysis of its efficiency and security. We also showed our proposal has wide applications in cryptography.

- First, we identify a class of homomorphic function $\phi : (D, \oplus) \to (R, \otimes)$ with computation disequilibrium: the evaluation of $\phi(y)$ for any $y \in D$ needs much less computational effort than the computation task of finding a pre-image of $x \in \phi(D)$. There are lots of examples in cryptography, like the encryption and decryption algorithm in Rabin's cryptosystem, the modular exponentiation and discrete logarithm, etc.
- We present a generic outsourcing scheme for the computation task of finding a pre-image of $x \in \phi(D)$ for the aforementioned class of homomorphic function. We proved the verifiability and input/output privacy of our proposal in information-theoretical sense. No public key operations are needed in our scheme, and there is only one round interaction between the client and server.
- We showed the practicality of our proposal by illustrating cryptographic applications.

Organization. The paper is organized as follows. In Section 2, we present the definition and security model of computation outsourcing scheme. In Section 3, we give precise definition of a class of homomorphic function with computation disequilibrium, and present the construction of computation outsourcing scheme for the homomorphic function inverse evaluation. The security proof and efficiency analysis of our proposal are presented in Section 4. In Section 5, we give concrete examples of our generic construction in cryptography and show the efficiency analysis through emulation. Finally, we conclude this paper in Section 6.

2 Computation Outsourcing and Security Model

2.1 Definition of Computation Outsourcing

A computation outsourcing scheme is a two-party protocol between a client C and server S. The client chooses a computation task f and an input x. His/her aim is to compute $f(x)$ with help of a server. First the client blinds x into σ_x for the sake of input privacy. The description of the computation task f and the blinded σ_x are provided to the server. The server computes σ_y according to σ_x and f. The client then extracts the expected result $f(x)$ from σ_y, which is provided by the server.

The definition of secure computation outsourcing scheme was given in [10, 20, 28]. Here we give a refined definition which is tailored for our construction in this paper. Here we describe the notation again.

- f: the computation task. Note that f is not necessarily a function.

– x: the input of the computation task, with domain $\mathcal{X}$;
– $f(x)$: the result of the computation task, with range $\mathcal{Y}$.
– $x \leftarrow_R \mathcal{X}$: x is chosen from $\mathcal{X}$ uniformly at random.

A secure (f, x)-computation outsourcing scheme consists of three probabilistic polynomial-time (PPT) algorithms (ProbGen,ProbGen, Verify), which are described as follows.

- $(\sigma_x, \tau) \leftarrow$ **ProbGen**(f, x)**.** On input the description of the computation task f and its input x, the problem generation algorithm outputs a state information τ and σ_x. Here σ_x is an encoded version of x.
- $\sigma_y \leftarrow$ **Compute**(f, σ_x)**.** On input the description of the computation task f and the encoded σ_x, the computation of the server results in an encoded output σ_y.
- $\{y, \perp\} \leftarrow$ **Verify**(f, x, σ_y, τ)**.** On input the description of the computation task f, its input x, the encoded σ_y and the state τ, the verification algorithm outputs y indicating that y is just the expected computation result $f(x)$, or $\perp$ indicating that σ_y is not valid.

Correctness of a (f, x)-computation outsourcing scheme requires that $f(x) =$Verify(f, x, σ_y, τ) for all $\sigma_y \leftarrow$ Compute(f, σ_x) and $(\sigma_x, \tau) \leftarrow$ ProbGen(f, x).

Unlike the previous computation outsourcing schemes [15] [13], since no public key operations are needed in our scheme, the KenGen$(\cdot)$ algorithm is excluded in our scheme, and the random numbers used in our scheme are generated on the fly.

2.2 Security Model

There are two security requirements for a (f, x)-computation outsourcing scheme: verifiability and privacy. Verifiability means that the server can not cheat by providing the client an corrupted $\tilde{\sigma}_y$ without detection, i.e., the probability that Verify$(\sigma_y, \tau) \notin \{f(x), \perp\}$ is negligible. Privacy includes input privacy and output privacy, which ensure that the server knows nothing about input x and output $f(x)$.

The security notions are formalized with the following experiments.
$\mathbf{Exp}_A^{verif}[f, \kappa]$
Query and response:
$x_0 = \sigma_{x_0} = \beta_0 = \perp$;
For $i = 1, \cdots, l$

– $x_i \leftarrow A(x_0, \sigma_{x_0}, \beta_0, \cdots, x_{i-1}, \sigma_{x_{i-1}}, \beta_{i-1})$.
– $(\sigma_{x_i}, \tau_i) \leftarrow$ ProbGen(f, x_i).
– $\sigma_{y_i} \leftarrow A(x_0, \sigma_{x_0}, \beta_0, \cdots, x_{i-1}, \sigma_{x_{i-1}}, \beta_{x_{i-1}}, \sigma_{x_i})$.
– $\beta_i =$ Verify$(f, x_i, \sigma_{y_i}, \tau_i)$.

Challenge:

– $x \leftarrow A(x_0, \sigma_{x_0}, \beta_0, \cdots, x_l, \sigma_{x_l}, \beta_l)$.
– $(\sigma_x, \tau) \leftarrow$ ProbGen(f, x).
– $\sigma_y \leftarrow A(x_0, \sigma_{x_0}, \beta_0, \cdots, x_l, \sigma_{x_l}, \beta_l, \sigma_x)$.
– $\hat{y} \leftarrow$ Verify(f, x, σ_y, τ).
– if $\hat{y} \neq f(x)$ and $\hat{y} \neq \perp$, output 1, else 0.

In the query phase, the malicious servers are given oracle access to both ProbGen and Verify. The adversary succeeds if he can convince the client to accept an output y, which is not the expected value $f(x)$. The security goal is to make sure that any adversary succeeds in the experiment only with negligible probability.

Definition 1. (Verifiability). *A computation outsourcing scheme is verifiable if any probabilistic polynomial-time (ppt) adversary $\mathcal{A}$ outputs 1 in Experiment $\mathbf{Exp}_A^{verif}[f,\kappa]$ with negligible probability, i.e.,* $\Pr\left[\mathbf{Exp}_A^{verif}(f,\kappa)=1\right]$ *is a negligible in κ.*

We define the input/output privacy based on a typical indistinguisability argument that guarantees no information about the input/output is leaked. Intuitively, a computation outsourcing scheme is input-private when the outputs of the problem generation algorithm ProbGen over two different inputs are indistinguishable. Formally, we define the input privacy with the following experiment.

$\mathbf{Exp}_A^{Ipriv}[f,\kappa]$
Query and response:
$x_0=\sigma_{x_0}=\beta_0=\perp$;
For $i=1,\cdots,l$

- $x_i \leftarrow A(x_0,\sigma_{x_0},\cdots,x_{i-1},\sigma_{x_{i-1}})$.
- $(\sigma_{x_i},\tau_i)\leftarrow \mathsf{ProbGen}(f,x_i)$.

Challenge:

- $(x^{(0)},x^{(1)})\leftarrow A(x_0,\sigma_{x_0},\cdots,x_l,\sigma_{x_l})$.
- $b\leftarrow_R\{0,1\}$, $(\sigma_{x^{(b)}},\tau^{(b)})\leftarrow \mathsf{ProbGen}(f,x^{(b)})$.
- $\hat{b}\leftarrow A(x_0,\sigma_{x_0},\cdots,x_l,\sigma_{x_l},\sigma_{x^{(b)}})$
- If $\hat{b}=b$, output 1, else 0.

In the above experiment, the adversary is given the oracle access to the problem generation algorithm in the query phase, the adversary succeeds if he can distinguish the output of the problem generation algorithm in the challenge phase, i.e, $\hat{b}=b$.

Definition 2. (Input privacy.) *A computation outsourcing scheme is input-private if*

$$\Pr\left[\mathbf{Exp}_A^{Ipriv}[f,\kappa]=1\right]-\frac{1}{2}$$

is negligible for any ppt adversary A running in $\mathbf{Exp}_A^{Ipriv}[f,\kappa]$.

The security requirement of output privacy can be defined similarly. We formalize the definition of output privacy with the following experiment.

$\mathbf{Exp}_A^{Opriv}[f,\kappa]$ *Query and response:*
$x_0=\sigma_{x_0}=\beta_0=\perp$;
For $i=1,\cdots,l$

- $x_i\leftarrow A(x_0,\sigma_{x_0},\beta_0,\cdots,x_{i-1},\sigma_{x_{i-1}},\beta_{i-1})$.
- $(\sigma_{x_i},\tau_i)\leftarrow \mathsf{ProbGen}(f,x_i)$.

– $\sigma_{y_i} \leftarrow A(x_0, \sigma_{x_0}, \beta_0, \cdots, x_{i-1}, \sigma_{x_{i-1}}, \beta_{x_{i-1}}, \sigma_{x_i})$.
– $\beta_i = \mathsf{Verify}(f, x_i, \sigma_{y_i}, \tau_i)$.

Challenge:

– Choose $x^{(0)}, x^{(1)} \in \mathcal{X}$ and $b \leftarrow_R \{0,1\}$.
– $(\sigma_{x^{(b)}}, \tau^{(b)}) \leftarrow \mathsf{ProbGen}(f, x^{(b)})$;
  $\sigma_{y^{(b)}} \leftarrow \mathsf{Compute}(f, \sigma_{x^{(b)}})$.
– $\hat{b} \leftarrow A(f(x^{(0)}), f(x^{(1)}), \sigma_{x^{(b)}}, \sigma_{y^{(b)}}, (x_j, \sigma_{x_j}, \beta_j)_{j=0,1,\cdots,l})$.
– If $\hat{b} = b$, output 1, else 0.

In the query phase, the adversary is given the oracle access to algorithms ProbGen and Verify. The adversary succeeds if he can distinguish the output of the Verify in the challenge phase.

Definition 3. (Output privacy.) *A computation outsourcing scheme is output-private if*

$$Prob[\mathbf{Exp}_A^{Opriv}[f, \kappa] = 1] - \frac{1}{2}$$

is negligible for any adversary A running in probabilistic polynomial time.

Previous security definitions limit the number l of queries to polynomial in the security parameter κ and the adversaries to be PPT algorithms. Here we remove all the limitations since our proof in this paper will be information-theoretic.

3 Computation Task and Construction of Secure OutSourcing Scheme

3.1 A Class of Computation Task

We consider a class of computation task which computes a preimage of an element under a homomorphic function.

Definition 4. *A homomorphism is a structure-preserving map between two algebraic structures (such as groups, rings, or vector spaces, etc). Without loss of generality, we consider group homomorphism. Suppose that there are two groups $(D, \oplus)$ and $(R, \otimes)$, where $\oplus$ and $\otimes$ are two binary operations with respect to group D and R. We say a function $\phi : (D, \oplus) \to (R, \otimes)$ is a group homomorphism if $\phi[g_1 \oplus g_2] = \phi[g_1] \otimes \phi[g_2]$, for all $g_1, g_2 \in D$.*

Some notations about group homomorphism are defined below.

– $\phi(D) = \{h \mid \exists g, g \in D, h = \phi[g]\}$ is the image of D under the homomorphism ϕ. It is easy to see that $\phi(D)$ is a subgroup of R.
– $\mathrm{Ker}(\phi) = \{g \mid \phi[g] = 0, g \in D\}$ is the kernel of the homomorphism ϕ.
– $\phi^{-1}[h] = \{g \mid \phi[g] = h, g \in D\}$ is the pre-image of h under the homomorphism ϕ.

According to the first isomorphism theorem of group homomorphism, $\phi(D) \cong D/Ker(\phi)$, where $D/Ker(\phi)$ is the quotient group. The pre-image of h is given by a coset of $\mathrm{Ker}(\phi)$ generated by an element g, where $\phi[g] = h$. In formula, $\phi^{-1}[h] = g \oplus \mathrm{Ker}(\phi)$ if $\phi[g] = h$.

Definition 5. *A group homomorphism $\phi : (D, \oplus) \to (R, \otimes)$ has* **computational disequilibrium** *if*

- *there exist PPT algorithms sampling an element y uniformly at random from D and sample an element x uniformly at random from R;*
- *there exist PPT algorithms implementing the binary operation $\oplus$ over D and $\otimes$ over R.*
- *there exist a PPT algorithm evaluating ϕ over D and let $O(\lambda(n))$ denote the computation complexity;*
- *there exist an algorithm computing a pre-image y of x such that $x = \phi[y], x \in \phi(D)$, and let $O(\mu_(n))$ denote the computation complexity;*
- *$\mu_(n) \gg \lambda(n)$.*

Computation Task: Given a group homomorphic function $\phi : (D, \oplus) \to (R, \otimes)$ with computational disequilibrium, and an element $x \in \phi(D)$, define the computation task ϕ^{-1} as

$$\phi^{-1}(x) := \text{find an element } y \text{ such that } \phi(y) = x,$$

i.e., find $y \in \phi^{-1}[x]$. Note that $\phi^{-1}(\cdot)$ is not necessarily a function unless ϕ is injective.

We emphasize the difference between $\phi^{-1}(x)$ and $\phi^{-1}[x]$: here $\phi^{-1}(x)$ denotes the computation task of finding an element $y \in D$ to meet $x = \phi(y)$. We also use $\phi^{-1}(x)$ to denote the output of the computation task, and generally it is an element. On the other hand, $\phi^{-1}[x]$ denote the pre-image set of x. Obviously, $\phi^{-1}(x) \in \phi^{-1}[x]$.

Remember that evaluation of function $\phi(\cdot)$ at $y \in D$ has much less computation complexity than computing $y \in \phi^{-1}[x]$ with $x = \phi(y)$. That is the exact reason of computation outsourcing.

Example 1. Let p be a prime number and $(\mathbb{Z}_p^*, \cdot)$ be a multiplicative group associated with modulo p multiplication. Then the automorphism $\phi : (\mathbb{Z}_p^*, \cdot) \to (\mathbb{Z}_p^*, \cdot)$, defined as

$$\phi[y] \equiv y^2 \mod p,$$

has computation disequilibrium. The Kernel of ϕ is $\{1, -1\}$. Let $\mathbb{QR}_p$ be the subgroup of quadratic residues in group $\mathbb{Z}_p^*$. Then $\phi(\mathbb{Z}_p^*) = \mathbb{QR}_p$.

- The computation of $\phi[y] \equiv y^2 \mod p$ is a modular multiplication, which needs $O((\log p)^2)$ bit-XORs.
- The computation of $\phi^{-1}(x) \equiv x^{1/2} \mod p$ for $x \in \mathbb{QR}_p$ needs $O((\log p)^3)$ bit-XORs if $p \equiv 3 \mod 4$, and $O((\log p)^4)$ bit-XORs if $p \equiv 1 \mod 8$ [11].

Example 2. Let p, q be prime numbers with $q|p-1$. Let $(\mathbb{Z}_p^*, \cdot)$ be a multiplicative group associated with modulo p multiplication. Let g be a generator. Let $(\mathbb{Z}_q, +)$ be an additive group with modulo q addition. Then $\phi : (\mathbb{Z}_q, +) \to (\mathbb{Z}_p^*, \cdot)$, defined as

$$\phi[y] \equiv g^{\frac{p-1}{q}y} \mod p,$$

is an injective group homomorphism with computation disequilibrium. Let $\mathbb{G}$ be the subgroup of order q in $\mathbb{Z}_p^*$. Then $g' = g^{\frac{p-1}{q}}$ is a generator of $\mathbb{G}$ and $\phi(\mathbb{Z}_q) = \mathbb{G} = \langle g' \rangle$.

- The computation of $\phi[y] \equiv g^{\frac{p-1}{q}y} \mod p$ is a modular exponentiation, which needs $O((\log p)^2 \log q)$ bit-XORs.
- The computation of $\phi^{-1}(x) \equiv \log_{g'} x \mod p$ is just the discrete logarithm of x to the base g', where $g' \equiv g^{\frac{p-1}{q}} \mod p$ and $x \in \mathbb{G} = \langle g' \rangle$. The best algorithms for this computation is still subexponential complexity for $\log p$ now.

Example 3. Let $(\mathbb{Z}_q, +)$ be an additive group with modulo q addition. Let $\mathbb{G}$ be an additive group of order q in some elliptic curve. Let P be a generator of $\mathbb{G}$. Then $\phi : (\mathbb{Z}_q, +) \to (\mathbb{G}, +)$, defined as

$$\phi[y] \equiv yP,$$

is an isomrophism with computation disequilibrium.

- The computation of $\phi[y] = yP$ is a scalar multiplication, which needs $O(\log q)$ group additions over $\mathbb{G}$.
- The computation of $\phi^{-1}(Q) = y$, such that $Q = yP$, is just the elliptic curve discrete logarithm of Q to the base P, which needs $O(\sqrt{q})$ group additions over $\mathbb{G}$.

There are many other examples: Given an invertible matrix A, to solve the inverse of A, or given an invertible matrix A and b, to solve x, such that $Ax = b$; Given an element a in a multiplicative group associated with modulo p multiplication, to find the inverse of a modulo p, etc. Any one-way functions with homomorphic property have computational disequilibrium.

3.2 Construction of Secure Outsourcing

Given a group homomorphism $\phi : (D, \oplus) \to (R, \otimes)$, and computation task $\phi^{-1}(x)$ for $x \in \phi(D)$, our computation oursourcing scheme is constructed as follows.

- $(\sigma_x, \tau) \leftarrow \mathsf{ProbGen}(\phi^{-1}, x)$. Choose $\tau \in D$ uniformly at random and encode x into $\sigma_x = \phi[\tau] \otimes x$. Output (σ_x, τ).
- $\sigma_y \leftarrow \mathsf{Compute}(\phi^{-1}, \sigma_x)$: Given σ_x and the task ϕ^{-1}, compute $\sigma_y := \phi^{-1}(\sigma_x)$. Output σ_y.
- $\{y, \perp\} \leftarrow \mathsf{Verify}(\phi^{-1}, x, \sigma_y, \tau)$: Check whether $\phi[\sigma_y] = \sigma_x$ holds. If not, output $\perp$. Otherwise compute $y := (-\tau) \oplus \sigma_y$. Output y.

The correctness of the scheme is given by the following facts: (1) $\phi[\sigma_y] = \phi[\phi^{-1}(\sigma_x)] = \sigma_x$; (2) $\phi[(-\tau) \oplus \sigma_y] = \phi[-\tau] \otimes \phi[\sigma_y] = \phi[-\tau] \otimes \sigma_x = \phi[-\tau] \otimes \phi[\tau] \otimes x = x$.

The above construction is quite concise and applies to any homomorphic function with computation disequilibrium. We tactfully employs homomorphic property of the function ϕ itself, rather than utilizing the complicated homomorphic encryption as the previous works did. Consequently, no public key operations are involved in our scheme, which greatly improves the efficiency. Our construction can be applied to many widely used homomorphic functions in cryptography, and some of the previous works on computation outsourcing of concrete functions can be reduced to our generic construction proposed above. In Section 5, we will present concrete examples for the generic construction.

4 Security proofs and efficiency analysis

4.1 Security proofs

Theorem 1. *Our scheme is verifiable according to Definition 2.*

Proof. As defined in Definition 2, we will prove that a malicious server can not persuade the client to accept an incorrect output. Our proof strictly follows the steps of the experiment $\mathbf{Exp}_A^{verif}[\phi^{-1}, \kappa]$ defined in Section 2.

$\mathbf{Exp}_A^{verif}[\phi^{-1}, \kappa]$
Query and response:
For $i = 1, \cdots, l$

- $x_i \leftarrow A(x_0, \sigma_{x_0}, \beta_0, \cdots, x_{i-1}, \sigma_{x_{i-1}}, \beta_{i-1})$.
- $(\sigma_{x_i}, \tau_i) \leftarrow \mathsf{ProbGen}(\phi^{-1}, x_i)$, where $\sigma_{x_i} = \phi[\tau_i] \otimes x_i$, and $\tau_i \leftarrow_R D$.
- $\sigma_{y_i} \leftarrow A(x_0, \sigma_{x_0}, \beta_0, \cdots, x_{i-1}, \sigma_{x_{i-1}}, \beta_{x_{i-1}}, \sigma_{x_i})$.
- $\beta_i = \mathsf{Verify}(\phi^{-1}, x_i, \sigma_{y_i}, \tau_i)$.

Challenge:

- $x \leftarrow A(x_0, \sigma_{x_0}, \beta_0, \cdots, x_l, \sigma_{x_l}, \beta_l)$.
- $(\sigma_x, \tau) \leftarrow \mathsf{ProbGen}(\phi^{-1}, x)$, where $\sigma_x = \phi[\tau] \otimes x$ and and $\tau \leftarrow_R D$.
- $\hat{\sigma}_y \leftarrow A(x_0, \sigma_{x_0}, \beta_0, \cdots, x_l, \sigma_{x_l}, \beta_l, \sigma_x)$.
- $\hat{y} \leftarrow \mathsf{Verify}(\phi^{-1}, x, \hat{\sigma}_y, \tau)$.

Let $\sigma_y = \phi^{-1}(\sigma_x)$. Since the quotient group $D/Ker(\phi)$ is isomorphic to subgroup $\phi(D)$ under homomorphic function ϕ, the pre-image set of σ_x under ϕ is given by the coset $\sigma_y \oplus \mathrm{Ker}(\phi)$.

- If $\hat{\sigma}_y \notin \sigma_y \oplus \mathrm{Ker}(\phi)$, then $\mathsf{Verify}(\tau, \hat{\sigma}_y)$ will output $\perp$.
- If $\hat{\sigma}_y \in \sigma_y \oplus \mathrm{Ker}(\phi)$, then $\mathsf{Verify}(\tau, \hat{\sigma}_y)$ will output an element in $\phi^{-1}[x] = \phi^{-1}(x) \oplus \mathrm{Ker}(\phi)$, which is just the expected computation result. The reason is as follows. Let $\hat{\sigma}_y = \sigma_y \oplus k$ for some $k \in \mathrm{Ker}(\phi)$. Then $\phi[\hat{\sigma}_y] = \phi[\sigma_y \oplus k] = \phi[\sigma_y] \otimes \phi[k] = \phi[\sigma_y] \otimes 1 = \phi[\sigma_y]$. We have
$$\phi[(-\tau) \oplus \hat{\sigma}_y] = \phi[(-\tau)] \otimes \phi[\hat{\sigma}_y] = \phi[(-\tau)] \otimes \sigma_x = x.$$
Consequently, $(-\tau) \oplus \hat{\sigma}_y \in \phi^{-1}[x]$.

Consequently, $\Pr\left[\mathbf{Exp}_A^{verif}(\phi^{-1}, \kappa) = 1\right] = 0$ regardless of the number of queries by A. □

Theorem 2. *Our scheme is input-private and output-private according to Definition 3 and Definition 4.*

Proof. As defined in Section 2, we will prove that the advantage of adversary A in experiment $\mathbf{Exp}_A^{Ipriv}[f^{-1}, \kappa]$ is negligible. As for our scheme, the experiment is implemented as follows:
Query and response:
For $i = 1, \cdots, l$

- $x_i \leftarrow A(x_0, \sigma_{x_0}, \cdots, x_{i-1}, \sigma_{x_{i-1}})$.

– $(\sigma_{x_i}, \tau_i) \leftarrow \mathsf{ProbGen}(\phi^{-1}, x_i)$, where $\sigma_{x_i} = \phi[\tau_i] \otimes x_i$ and $\tau_i \leftarrow_R D$.

Since τ_i is randomly chosen in D, we know that $\phi[\tau_i]$ is uniformly randomly distributed in $\phi(D)$. Then the encoding $\sigma_{x_i} = \phi[\tau_i] \otimes x_i$ can be seen as a one-time pad encryption with a random key $\phi(\tau_i)$. According to Shannon's information theory, one-time pad encryption has perfect secrecy. Therefore, in the *Query and response* phase of the experiment $\mathbf{Exp}_A^{Ipriv}[f, \kappa]$ leaks no information to the adversary except for what the adversary has already obtained.
Challenge:

– $(x^{(0)}, x^{(1)}) \leftarrow A(x_0, \sigma_{x_0}, \cdots, x_l, \sigma_{x_l})$.
– $b \leftarrow_R \{0, 1\}$, and $(\sigma_{x^{(b)}}, \tau^{(b)}) \leftarrow \mathsf{ProbGen}(x^{(b)})$, where $\sigma_{x^{(b)}} = \phi[\tau] \otimes x^{(b)}$ and $\tau \leftarrow_R D$.
– $\hat{b} \leftarrow A(x_0, \sigma_{x_0}, \cdots, x_l, \sigma_{x_l}, x^{(0)}, x^{(1)}, \sigma_{x^{(b)}})$.

In the *Challenge* phase, τ is independently and uniformly chosen from D. Hence $\sigma_{x^{(b)}} = \phi[\tau] \otimes x^{(b)}$ is again a one-time pad encryption of $x^{(b)}$ with a random key τ. The input $x^{(b)}$ is perfectly hidden in the ciphertext $\sigma_{x^{(b)}}$. Thus, adversary A can do nothing but randomly guess the bit $\hat{b}$. Therefore,

$$\Pr\left[\mathbf{Exp}_A^{Ipriv}[f, \kappa] = 1\right] = \frac{1}{2}.$$

As for the output privacy, the queries leaks no information to the adversary due to the perfect privacy of one-time pad. Adversary A is given the challenge $(\phi^{-1}(x^{(0)}), \phi^{-1}(x^{(1)}), \sigma_{x^{(b)}}, \sigma_{y^{(b)}})$, and he is going to guess the value of b. We know that $\sigma_{x^{(b)}} = \phi[\tau] \otimes x^{(b)}$, and $\phi[\tau]$ is uniformly distributed over $\phi(D)$, hence $x^{(b)}$ is perfectly hidden from A. On the other hand, $\sigma_{y^{(b)}} \in \tau^{(b)} \oplus \phi^{-1}(x^{(b)}) \oplus \mathrm{Ker}(\phi)$, then $\phi^{-1}(x^{(b)}) \in \sigma_{y^{(b)}} \oplus \tau^{(b)} \oplus \mathrm{Ker}(\phi)$. We know that $\tau^{(b)}$ is randomly chosen from D, hence the coset $\tau^{(b)} \oplus \mathrm{Ker}(\phi)$ is randomly distributed over all the $|\phi(D)|$ cosets. Conditioned on the information of $\phi^{-1}(x^{(0)}), \phi^{-1}(x^{(1)})$, we know that $\Pr\left[\phi^{-1}(x^{(0)}) \in \sigma_{y^{(b)}} \oplus \tau^{(b)} \oplus \mathrm{Ker}(\phi)\right] = 1/2$. Hence A correctly guesses b with probability exactly $1/2$. □

It should be noted that our scheme is information-theoretic secure. All the above proofs does not rely on the security parameter κ, and there is no computational assumptions at all.

4.2 Efficiency analysis

In Section 3, we assumed that the binary operations $\oplus$ and $\otimes$ can be efficiently implemented in D and R and the computation complexity of ϕ^{-1} is far more greater than that of ϕ. In other words, let $O(\lambda(n))$ and $O(\mu_(n))$ denote the computation complexity of ϕ and ϕ^{-1} respectively, then $\mu_(n) \gg \lambda(n)$. For the client, the computation complexity consists of four parts: one evaluation of ϕ on input τ, one $\otimes$ operation to generate the blinded input $\sigma_x = \phi[\tau] \otimes x$, one inverse operation to compute $-\tau$, and one $\oplus$ operation to recover the result y. We denote the total complexity of these computations as $O(\theta(n))$. Therefore, our scheme is outsourceable as long as $O(\theta(n) < O(\lambda(n))$, where n is the length of the input. In the following section, we will show that many widely used cryptography functions satisfy this condition, and thus can be outsourced with our construction.

5 Instantiations and performance evaluation

In this section, we first present an instantiation of our construction for Example 1, and analyze its efficiency through emulation. Then we show how instantiations of our construction for Example 2 and 3 are employed in cryptanalysis. At last, we give other positive application of computation outsourcing of discrete logarithm (for Example 2 and 3) in cryptography.

5.1 Instantiation and Performance evaluation

Recall that the homomorphism $\phi : (\mathbb{Z}_p^*, \cdot) \to (\mathbb{Z}_p^*, \cdot)$ in Example 1 is defined by

$$\phi[y] \equiv y^2 \mod p.$$

Consider the computation task

$$\phi^{-1}(x) \equiv x^{1/2} \mod p$$

for $x \in \mathbb{QR}_p$. The aim is finding a square root of a quadratic residue x modulo a large prime p. If $y \equiv x^{1/2} \mod p$, then $-y \equiv x^{1/2} \mod p$ as well, since $\mathrm{Ker}(\phi) = \{\pm 1\}$.

The computation task is widely used in encryptions and signature schemes. The average computation complexity of $x^{\frac{1}{2}}$ is $O((\log p)^3)$ in terms of bit-XORs, when $p \equiv 3 \mod 4$. And the computation complexity gets even worse, which is $O((\log p)^4)$, when $p \equiv 1 \mod 8$. In the descriptions below, we show how to outsource the computation of $x^{\frac{1}{2}} \mod p$ according to the generic construction proposed in Section 3.

- $(\sigma_x, \tau) \leftarrow \mathsf{ProbGen}(\phi^{-1}, x)$. The client randomly selects an element $\tau \in Z_p$ and encodes x to $\sigma_x \equiv \tau^2 x \mod p$. Then he/she sends σ_x to the server S.
- $\sigma_y \leftarrow \mathsf{Compute}(\phi^{-1}, \sigma_x)$. Given σ_x and the description of computation task ϕ^{-1}, the server simply computes $\sigma_y = \phi^{-1}(\sigma_x) \equiv (\tau^2 x)^{\frac{1}{2}} \mod p$ and returns σ_y to the client C.
- $\{y, \perp\} \leftarrow \mathsf{Verify}(\phi^{-1}, x, \sigma_y, \tau)$. Given σ_y from S, C first checks whether ${\sigma_y}^2 \equiv \sigma_x \mod p$ holds. If not, output $\perp$. Otherwise compute the result $y \equiv \tau^{-1}\sigma_y \mod p$.

Note that if $\sigma_y \equiv (\tau^2 x)^{\frac{1}{2}} \equiv \pm\tau x^{\frac{1}{2}} \mod p$ then $y \equiv \tau^{-1}\sigma_y \equiv \pm x^{\frac{1}{2}} \mod p$.

According to the above computation outsourcing scheme for square root of a quadratic residue, the client's computation consists of one modular square operation, two modular multiplications, and one inverse operation. Generally, the average computation complexity for both modular square and modular multiplication is $O((\log p)^2)$ bit-XORs, and the average computation complexity of the best inverse algorithm is $O((\log p)^2)$ bit-XORs [11] as well. Therefore, the total computation complexity of the client is again $O((\log p)^2)$. Review that the average computation complexity of the square root computation is $O((\log p)^4)$ when $p \equiv 1 \mod 8$. Therefore, the theoretical analysis shows that the scheme is quite efficient for the client.

To verify the theoretical efficiency analysis, we emulate the implementation of the scheme on a computer and test the time consumptions of each operations in the protocol. In the emulation experiment, $|p|, |\tau|, |x|$ are set from 128 bits to 1024 bits with $p \equiv 1 \mod 8$. The configuration of our platform is as follows: Inter(R) Core(TM)2 Duo CPU @2.4GHz and the

size of the RAM is 1.95G. In Table 1, we show the time costs comparison between the client's calculation of square root by himself/herself and his/her computation in the proposed outsourced scheme. The experimental results shows that our scheme greatly outperforms the direct calculation method. Precisely, the time costs of client's computing square root directly is about $(\log p)^2$ times greater than the client's time costs in our outsourcing scheme. It is consistent with the theoretical analysis.

Table 1. Client's Time costs comparison

Bit length	Direct calculation (s)	Our scheme(s)
128	0.032	10.8×10^{-6}
256	0.25	17.1×10^{-6}
384	1.219	27.7×10^{-6}
512	7.954	41.9×10^{-6}
640	14.484	58.8×10^{-6}
768	46.172	81.2×10^{-6}
896	98.469	97.3×10^{-6}
1024	277.734	121.8×10^{-6}

5.2 Other Applications

Modern cryptography assumes the existence of one-way functions. The well-known candidates are $f(x, y) = x \cdot y$ and $f(g, x) \equiv g^x \mod p$, where integer factoring and discrete logarithm problems are believed to be hard. Most of encryption schemes today are based on the factoring-related assumptions and discrete-logarithm related assumptions. Our proposal makes potential cryptanalysis of the encryption schemes more powerful with help of computation outsourcing.

For example, the Certicom ECC Challenge [14] is to determine the private key y, given yP and P, which is a generator of an additive group $\mathbb{G}$ on some elliptic curve. This is exactly Example 3 in Section 3. Using our proposal, the computation of elliptic curve discrete logarithm can be outsourced to servers with greater computation power.

Another application is the implementation of Quadratic Sieve Algorithm to factor integer N, which is a product of two prime numbers. The time-consuming part of the algorithm is the computation of square root $x^{1/2} \mod N$. This part of computation can be outsourced to servers exactly like the instantiation in subsection 5.1, and the only difference is that $\mathbb{Z}_q^*$ is replaced by $\mathbb{Z}_N^*$.

Our proposal can also find positive applications in cryptography. Although discrete logarithms are believed to be hard problems. When the exponent is limited in a small interval (set), discrete logarithm is not difficult any more. Computations of such discrete logarithms with small intervals are used as subroutines in several cryptographic protocols in the literature. For example, the BGN degree-2-homomorphic public-key encryption [6] system uses generic square-root discrete-logarithm methods for decryption. In [5], Bernstein shows that computing a discrete logarithm in an interval of order l takes only $1.93l^{\frac{1}{3}}$ multiplications on average using a table of size $l^{\frac{1}{3}}$ built from $1.21l^{\frac{2}{3}}$ multiplications, and computing a discrete logarithm in a group of order l takes only $1.77l^{\frac{1}{3}}$ multiplications on average using a table

of size $l^{\frac{1}{3}}$ built from $1.24l^{\frac{2}{3}}$ multiplications. Although this is an improvement on the computation of discrete logarithm, the complexity is still much greater than that of computing multiplicative inverse, which is $O(|l|)$ in terms of modular multiplication. Remember that $|l| \approx \log_2 l$ denotes the bit length of l.

We know that the discrete logarithm function is just the computation task $\phi^{-1}(x) \equiv \log_{g'} x \mod p$ in Example 2 in Section 3. Therefore, it can also be outsourced following our proposal (for Example 3, it can be similarly instantiated in the additive group).

- $(\sigma_x, r) \leftarrow \mathsf{ProbGen}(\phi^{-1}, x)$. The client randomly selects an element $r \in Z_p^*$ and generates the blinded input as $\sigma_x = xg'^r$. Then he sends σ_x to the server S.
- $\sigma_y \leftarrow \mathsf{Compute}(\phi^{-1}, \sigma_x)$. Given σ_x and the function ϕ^{-1}, the server simply computes $\sigma_y = \log(g'^r y)$ and returns σ_y to the client C.
- $\{y, \perp\} \leftarrow \mathsf{Verify}(\phi^{-1}, x, \sigma_y, r)$. On receiving the output σ_y from S, C first checks whether $g'^{\sigma_y} = \sigma_x$ holds. If not, output $\perp$. Otherwise compute $\phi^{-1}(x) = \sigma_y - r$.

Note that not only the square root and discrete logarithm problem can be outsourced using our scheme, the matrix inversion and decoding computations for error-correct coding can also be outsourced using our construction. Those computations are all homomorphic with respect to their own algebra structure $(R, \otimes)$ and $(D, \oplus)$. Moreover, the corresponding inverse functions of these functions can be evaluated much more efficient with outsourcing. Retrospect on the works of oursourcing computation of matrix inversion or linear equation in [2, 1, 28], the construction of the schemes all take advantage of the fact that the computation complexity of matrix inversion is much more greater than that of matrix multiplication, and the matrix operations used in the schemes are homomorphism. In nature, these schemes can be reduced to our construction in Section 3.

6 Conclusion

In this paper, we propose a generic computation outsourcing scheme for inverting homomorphic functions with computation disequilibrium. We utilize the intrinsic property of such functions to construct a concise and secure generic computation outsourcing scheme. No public key operations are used in our scheme, and the scheme is round efficient. The client and the server just have to exchange one message with each other. We give formal security proofs of our scheme according to the security definitions of verifiability, input and output privacy. The security proof is information-theoretic and does not rely on any computational assumption. We instantiate the generic construction with a few concrete examples. The experimental result justifies the efficiency of our construction.

References

1. M. J. Atallah and K. B. Frikken. Securely outsourcing linear algebra computations. AISACCS 2010, pp. 48-59. 2010.
2. M. J. Atallah, K. N. Pantazopoulos, J. R. Rice, and E. H. Spafford, Secure outsourcing of scientific computations, Advances in Computers, vol. 54, pp. 216-272, 2001.
3. B. Applebaum, Y. Ishai, E. Kushilevitz. From secrecy to soundness: Efficient verification via secure computation. ICALP 2010, pp. 152-163. 2010.
4. Amazon Elastic Compute Cloud. Online at http://aws.amazon.com/ec2.

5. D. J. Bernstein and T. Lange. Computing small discrete logarithms faster. http://eprint.iacr.org/2012/458
6. D. Boneh, E. J. Goh, K. Nissim, Evaluating 2-DNF formulas on ciphertexts. TCC 2005. pp. 325-341. 2005.
7. D. Benjamin and M. J. Atallah, Private and cheating-free outsourcing of algebraic computations, PST 2008, pp. 240-245, 2008.
8. L. Babai. Trading group theory for randomness. STOC 1985, pp. 421-429. ACM 1985.
9. M. Barbosa and P. Farshim. Delegatable homomorphic encryption with applicatoins to secure outsourcing of computation. CT-RSA.2012.
10. S. Benabbas, R. Gennaro and Y. Vahlis. Verifiable delegation of computation over large datasets. CRYPTO 2011, LNCS 6841, pp. 111-131. 2011.
11. H. Cohen. A course in computational algebraic number theory. Graduadte tests in mathematics 138. 1996.
12. Xiaofeng Chen, Jin Li, Jianfeng Ma, Qiang Tang and Wenjing Lou. New algorithms for secure outsourcing of modular exponentiations. ESORICS 2012, LNCS 7459, pp. 541-556. 2012.
13. K. M. Chung, Y. Kalai, S. P. Vadhan. Improved delegation of computation using fully homomorphic encryption. CRYPTO 2010, LNCS 6223, pp. 483-501. 2010.
14. Certicom ECC Challenge, http://www.certicom.com/ and http://pauillac.inria.fr/ harley/ecdl/.
15. R. Gennaro, C. Gentry, and B. Parno. Non-interactive verifiable computing: computation outsourcing to untrusted workers. CRYPTO 2010, LNCS 6223, pp. 465-482. 2010.
16. M. Green, S. Hohenberger, B. Waters, Outsourcing the decryption of ABE ciphertexts, USENIX 2011. The full version can be found at http://static.usenix.org/events/sec11/tech/fullpapers/Green.pdf
17. C. Gentry. Fully homomorphic encryption using ideal lattices. STOC 2009, pp. 169-178. 2009.
18. S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof systems. SIAM Journal on computing, vol 18, No. 1, pp. 186-208. 1989.
19. S. Goldwasser, Y. T. Kalai, and G. N. Rothblum. Delegating computation: interactive proofs for muggles. STOC 2008, pp. 113-122, 2008.
20. Seny Kamara, Payman Mohassel, and Mariana Raykova. Outsourcing multi-party computation. Cryptology ePrint Archive, Report 2011/272, 2011. http://eprint.iacr.org/.
21. M. Jakobsson and S. Wetzel. Secure server-aided signature generation. PKC 2001, LNCS 1992, pp. 383-401. 2001.
22. J. Kilian. A note on efficient zero-knowledge proofs and arguments. STOC 1992, pp. 723-732. 1992.
23. J. Kilian. Improved efficient arguments. CRYPTO 1995, LNCS 963, pp. 311-324. 1995.
24. C. H. Lim and P. J. Lee. More flexible exponentiation with precomputation. CRYPTO 1994. LNCS 839, pp. 95-107. 1994.
25. S. Micali. CS proofs (extended abstract). Proceeding of the IEEE symposium on foundations of computer science, 1994.
26. B. Parno, M. Raykova and V. Vaikuntanathan, How to delegate and verify in public: verifiable computation from attribute-based encryption, to appear in TCC 2012.
27. C. Papamanthou, Elaine Shi, R. Tamassia. Publicly verifialbe delegation of computation. http://eprint.iacr.org/2011/587.
28. P. Mohassel. Efficient and secure delegation of linear algebra. http://eprint.iacr.org/2011/605.
29. M. O. Rabin. Digital signatures and public-key functions as intractible as factorization. Technical reoprt LCS/TR-212. MIT laboratory for computer science. 1979.
30. A. Sahai, H. Seyalioglu, and B. Waters. Dynamic credentials andciphertext delegation for attribute-based encryption. CRYPTO 2012, LNCS 7417, pp. 199-217, 2012.
31. Sun Utility Computing. Online at http://www.sun.com/service/sungrid/index.jsp.