社団法人 情報処理学会 研究報告
IPSJ SIG Technical Reports

2005－ＣＳＥＣ－30 (39)
2005／7／22

ElGama暗号を用いた秘密回路計算について

山本 剛† 千田 浩司† アンダーソンナシメント† 鈴木幸太郎† 内山 成憲†

† 日本電信電話株式会社
神奈川県横須賀市光の丘 1-1

あらまし 閾値 ElGamal 暗号を用いた秘密計算方式の改良を提案する．提案方式は，知る限りの既存の方式に比べて，計算量の観点で効率に優れる．

キーワード ElGamal 暗号，マルチパーティー計算，秘密計算

On a secure circuit evaluation protocol using ElGamal encryption

Go YAMAMOTO†, Koji CHIDA†, Anderson NASCIMENT†, Kouratou SUZUKI†, and Shigenori UCHIYAMA†

† NTT Corporation
1-1 Hikarinooka, Yokosuka, Japan

Abstract We propose a protocol for implementing secure function evaluation based on the homomorphic threshold ElGamal encryption scheme. To the best of our knowledge, our solution is more efficient in terms of computational complexity than previous solutions existent in the literature.

Key words ElGamal encryption, multiparty computation, secure function evaluation

1. Introduction

1.1 Background

Two-party secure function evaluation consists of a protocol which allows n players, to compute a function $f(\cdot)$, which depends on inputs from the players, such that at the end of the protocol: the parties are sure that the result of the computation is correct; no party has learned more about each other's input than what can be computed from the output itself; dishonest players did not obtain significant knowledge about the output of the protocol while preventing the honest parties from receiving the result of the computation.

Secure function evaluation (SFE) is a central problem in the theory of cryptography and has received considerable attention since its introduction in [19]. Several different solutions, based on a wide range of models and assumptions, were proposed, e.g. [2], [11], [19]

However, it is still a big challenge to design protocols which are *secure* and *efficient*. Most of the proposed works till now aimed at proving the impossibility/possibility of SFE in principle, rather than in practice.

With the advent of ubiquitous computing and the bigger role played by low-computational powered devices in security protocols, the search for efficient protocols, in terms of computational complexity, for implementing SFE becomes a crucially important topic.

In this contribution, we give an efficient protocol for implementing secure function evaluation based on the DDH assumption which, to the best of our knowledge, possesses better computational and communication complexities than previous solutions in the literature attaining a similar level of security.

1.2 Previous Work

Many function evaluation protocols were presented in the literature in several different forms and flavors. The problem was first considered by Yao in [19], who proved that given the existence of one-way functions, any distributed computation can be securely implemented.

Subsequently, several researchers tried to obtain more efficient protocols by using different computational assumptions.

In [5], Cramer et al. proposed a generic and very efficient protocol, in terms of communication complexity, implementing secure distributed computation based on homomorphic encryption. They also proved that their protocol is secure when implemented with the Paillier encryption scheme.

It would be desirable to replace the Paillier encryption used in [5] by the elliptic ElGamal encryption since the former can be implemented with much less computational effort and its key generation process is much simpler and efficient. This goal was achieved in [17], where a SFE protocol was proposed which has its security based on the homomorphic ElGamal encryption. The proposed scheme is secure against active adversaries, and the computational and communication complexities of the protocol are linear in the number of players. However, two prices were paid to obtain this improvement: (*i*) the round complexity of the protocol depends on the number of players (what is not a big problem if the number of players is not large) and (*ii*) the scheme of [17] is optimistic, that is, its performance degrades if the parties engaged in the protocol misbehave.

1.3 Our Contribution

This paper proposes a protocol for secure function evaluation which is as secure as the protocol proposed in [17], but achieves an improved communication and computational complexities while being *non-optimistic*.

We achieve this improvement by modifying the original protocol for computing conditional gates proposed in [17] so as to remove cheating parties from the computation without any need to re-start it and by using slightly modified versions of known zero-knowledge proofs ([6], [3], and [16]).

1.4 Road Map

Our paper is organized as follows. In Section 2 we present our security model, definitions and assumptions. In Section 3 we present our protocol, its security and performance analysis. Conclusions and open problems are given in Section 4. Some proofs of knowledge used in the main protocol are stated in an Appendix.

2. Security Model

Our model is very similar to the ones presented in [5] and [17].

We assume that there are n players connected by authenticated channels and a broadcast channel. We assume synchronous communications among the players. No more than $t < n$ players are corrupted by a static malicious adversary (thus, the adversary has to choose which parties are to be corrupted before the beginning of the protocol and does not corrupt any more players once the protocol starts). All the parties, including the adversary, are polynomial time Turing machines.

The goal of the protocol is to evaluate a certain function F represented as a binary circuit composed of addition and multiplication gates.

To define security, we introduce a trusted third party, which is connected by private and synchronous channels to all other players. A protocol secure in the *ideal* world is one where all the players give their respective inputs to the trusted party which them computes the outputs of all the players and distribute them to the respective parties.

A protocol secure in the *real* world is one which efficiently emulates the ideal protocol previously described. That is, any adversary attacking the real protocol can be simulated in polynomial time, given only the view of an adversary attacking the ideal protocol. For further details we refer to [5].

We assume the hardness of the decisional Diffie-Helmann problem (DDH problem) and the existence of random oracles, as in [17].

We note here that, in the case a majority of the players is dishonest, there are certain attacks which are unavoidable. For instance, if $t > n/2$ players are dishonest, always a subset $n - t$ of players will be able to abort the protocol. Thus, in the case of dishonest majority we always consider a non-aborting adversary. Moreover, if $t > n/2$ nothing prevents cheating parties from leaving the protocol after they have obtained their desired output, even if the honest parties have not yet received theirs. Thus, *strong* fairness is never achieved in this scenario. Therefore, in our work, in case of dishonest majority, we aim at a weaker form of fairness, where cheating parties can leave the protocol with some, but not significant, advantage over honest parties.

3. Proposed Protocols

3.1 Preliminary

In our protocol we use the threshold ElGamal encryption scheme. For the sake of simplicity, we first introduce

its non-threshold version.

Consider a cyclic group $\mathcal{G}$ of order q generated by G where the DDH problem is hard. Consider the ElGamal public key $(G, H = uG)$ and its secret key u.

An encryption of $a \in \{0,1\}$ is defined as:

$$E(a,r) \stackrel{\text{def}}{=} (rG, (a+r)H), \tag{1}$$

where $r \in_R \mathbb{Z}/q\mathbb{Z}$.

a can be obtained from $E(a,r)$ by dividing $(a+r)H$ by urG and then computing the discrete logarithm of the result. This is infeasible in general. However, in our case, because a is always taken from a small (binary) domain, this task can be performed efficiently.

Note that this encryption scheme is homomorphic, that is, $E(a,r) \times E(\tilde{a},\tilde{r})^{\lambda} = E(a+\lambda\tilde{a}, r')$, for a publicly known value λ. Thus, linear operations are easily implementable on ciphertexts. Also, it is easy to see that, given $E(a,r)$, $E(\tilde{a},\tilde{r})$ and λ, a party A can prove in zero-knowledge to a verifier B that $E(a+\lambda\tilde{a},\tilde{r})$ is indeed a valid encryption of $a+\lambda\tilde{a}$. For further details, please see Appendix.

In the multi-party setting in this paper, players keep shares of the secret key u in advance. We can use key generation schemes as the ones proposed in [14] and k-out-of-n verifiable secret sharing schemes as [8] to securely distribute shares of an unknown and randomly chosen secret key u.

Our goal in the subsequent sections is to securely implement an operation $\oplus$ on $a \in \{0,1\}$ and b taken not necessarily from a binary domain, such that:

$$a \oplus b \stackrel{\text{def}}{=} \begin{cases} b & (a=0) \\ 1-b & (a=1) \end{cases}$$

It is obvious that $\oplus$ stands for the ordinary XOR if $a, b \in \{0,1\}$. A gate implementing $a \oplus b$ was called a conditional gate in [17], where they, together with addition gates, were proven to be sufficient for realizing secure function evaluation. In the next section we give a new, more efficient protocol for obtaining conditional gates.

3.2 Proposed Protocol for Implementing Conditional Gates

Here we give a new, more efficient protocol for implementing conditional gates. Compared to the protocol proposed in [17] our solution is non-optimistic while presenting a slightly better computational performance.

Hereafter, $E(\cdot)$ stands for the threshold homomorphic ElGamal encryption with its secret key shared among n players. We start with input bits $a, b \in \{0,1\} \subset \mathbb{Z}/q\mathbb{Z}$ (not necessarily known to any of the players) and random numbers $r, s \in_R \mathbb{Z}/q\mathbb{Z}$. The proposed protocol uses $E(a,r)$ and $E(b,s)$ as input, and computes $E(a \oplus b, t)$ for some $t \in_R \mathbb{Z}/q\mathbb{Z}$ while keeping a and b secret through its entire execution. Denote the n players participating in the protocol by $P_1, P_2, \cdots, P_n$.

The basic idea of the protocol is close to that of [17], but the scheme is different. The proposed protocol requires no translation from $\{0,1\}$ to $\{-1,1\}$ for input/output bits, and no Pedersen commitments in the process, what overall results in a more efficient protocol. We assume that at a setup phase, the players generated a public key π and secret shares of a private key σ of an ElGamal encryption scheme, for instance, by using the protocols proposed in [14].

Our proposed protocol is described in Table 1. It is easy to verify the correctness of our protocol. First check that

$$e_1 \oplus (e_2 \oplus b) = (e_1 \oplus e_2) \oplus b,$$

where $e_1, e_2 \in \{0,1\}$, $b \in \mathbb{Z}/q\mathbb{Z}$, and $\oplus$ stands for

$$a \oplus b \stackrel{\text{def}}{=} \begin{cases} b & (a=0) \\ 1-b & (a=1). \end{cases}$$

The output of Protocol Z is $E(a_n \oplus b_n, r'')$ for some r''. Since $a_n = e_1 \oplus e_2 \oplus \cdots \oplus e_n \oplus a$ and $b_n = e_1 \oplus e_2 \oplus \cdots \oplus e_n \oplus b$, thus

$$\begin{aligned} a_n \oplus b_n &= (e_1 \oplus e_2 \oplus \cdots \oplus e_n \oplus a) \oplus (e_1 \oplus e_2 \oplus \cdots \oplus e_n \oplus b) \\ &= (e_1 \oplus e_2 \oplus \cdots \oplus e_n \oplus a \oplus e_1 \oplus e_2 \oplus \cdots \oplus e_n) \oplus b \\ &= a \oplus b, \end{aligned}$$

hence $E(a_n \oplus b_n, r'') = E(a \oplus b, r'')$.

3.3 Security Analysis

[Theorem 1] In protocol Z, when less than $t < n$ players are corrupted, the adversary does not learn any non-negligible information about a, b under the DDH assumption and the random oracle model.

PROOF (Sketch)

Since we assume that, in the case $t > n/2$ corrupted adversaries do not abort the protocol, and the threshold of the ElGamal encryption is always set to be larger than t, we know that protocol Z does not abort and that unauthorized ciphertexts are never decrypted.

The proof will follow from two facts: the security of the proofs of knowledge presented in the appendix and the fact that during the entire protocol,

Parties: players $\{P_i\}_{i=1,2,\cdots,n}$.
Common input: ciphertexts $E(a,r)$, $E(b,s)$, where $a \in \{0,1\}$, $b \in \mathbb{Z}/q\mathbb{Z}$
Output: ciphertext $E(a \oplus b, r'')$ for some random r''.

(1) Let $E(a_0,r_0) \stackrel{\text{def}}{=} E(a,r)$, $E(b_0,s_0) \stackrel{\text{def}}{=} E(b,s)$.
(2) Repeat the following steps for $i = 1,2,...,n$.
(a) P_i generates a random bit e_i and a random number $t_i, u_i \in_R \{0,1,2,\cdots,p-1\}$.
(b) P_i takes $E(a_{i-1},r_{i-1}), E(b_{i-1},s_{i-1})$ as input, computes $E(a_i,r_i) = (A',B'), E(b_i,s_i) = (X',Y')$ according to the equations below. If $i \neq n$ then P_i sends this result to P_{i+1}, otherwise move to Step 3.
Denote $(A,B) = E(a_{i-1},r_{i-1})$, $(X,Y) = E(b_{i-1},s_{i-1})$.

$$(A',B') = \begin{cases} (A,B) + (t_iG, t_iH) & (e_i = 0) \\ (-A,-B) + (t_iG, (t_i+1)H) & (e_i = 1) \end{cases} \tag{2}$$

$$(X',Y') = \begin{cases} (X+Y) + (u_iG, u_iH) & (e_i = 0) \\ (-X,-Y) + (u_iG, (u_i+1)H) & (e_i = 1) \end{cases} \tag{3}$$

(c) P_i proves in zero knowledge (by using protocol B described below) that he has acted honestly in the previous step. Set

$$\begin{aligned} &((G,p,G,H),(G_0,H_0),(G_1,H_1),\\ &(G_0',H_0'),(G_1',H_1'),(e,t,t')) \leftarrow \\ &((G,p,G,H),(A'-A,B'-B),(A+A',B+B'-H),\\ &(X'-X,Y'-Y),(X+X',Y+Y'-H),(e_i,t_i,u_i)). \end{aligned}$$

(d) In case P_i fails to prove he acted honestly in the previous step set his output to $E(a_{i-1},r_{i-1})$, $E(b_{i-1},s_{i-1})$ and exclude him from the remaining of the protocol.
(3) The players decrypt $E(a_n,r_n)$ using verifiable ElGamal distributed decryption, and open a_n publicly.
(4) Define $(X,Y) = E(b_n,s_n)$. (C,D), the output of the protocol, is:

$$(C,D) = \begin{cases} (X,Y) & (a_n = 0) \\ (-X,-Y-H) & (a_n = 1) \end{cases} \tag{4}$$

Table 1 Protocol Z (Conditional Gate).

Common input: $(G,H,G_0,H_0,G_1,H_1,G_0',H_0',G_1',H_1') \in \mathcal{G}_q^{10}$, where $(\mathcal{G}_q,G,H) \leftarrow \mathsf{G}^{\mathsf{DL}}(1^\kappa)$.
Private input to P: $b \in \{0,1\}$, $s,t \in \mathbb{Z}/q\mathbb{Z}$ s.t. $G_b = sG$, $H_b = sH$, $G_b' = tG$, and $H_b' = tH$.
Statement to prove: $(G,H,G_0,H_0,G_1,H_1,G_0',H_0',G_1',H_1') \in$ ANDORDL.
(1) P chooses $r,v,c_{1-b} \in_R \mathbb{Z}/q\mathbb{Z}$ and computes $R_G^b = rG$, $R_H^b = rH$, $R_G^{1-b} = vG + c_{1-b}(eG_{1-b} + G'_{1-b})$, $R_H^{1-b} = vH + c_{1-b}(eH_{1-b} + H'_{1-b})$, $c_b = \mathcal{H}_1(R_G^0||R_H^0||R_G^1||R_H^1) - c_{1-b}$, $z_b = r - c_b(se+t)$, and $z_{1-b} = v$, where $e = \mathcal{H}_0(G||H||G_0||H_0||G_1||H_1||G_0'||H_0'||G_1'||H_1')$ and $\mathcal{H}_0$ and $\mathcal{H}_1$ are hash functions that map $\{0,1\}^* \to \mathbb{Z}/q\mathbb{Z}$. It then sends (z_0,z_1,c_0,c_1) to V.
(2) V verifies
$c_0 + c_1 = \mathcal{H}_1(z_0G + c_0(eG_0 + G_0')||z_0H + c_0(eH_0 + H_0')||z_1G + c_1(eG_1 + G_1')||z_1H + c_1(eH_1 + H_1'))$ and returns accept or reject.

Table 2 Protocol B: a protocol for honest verifier zero-knowledge proof of knowledge for a witness on the membership of ANDORDL(see Appendix).

only data indistinguishable from random is presented to the adversary. In detail, the input for Protocol B is computationally indistinguishable from a random input from the point of view of the adversary because $((G_0,H_0),(G_1,H_1),(G_0',H_0'),(G_1',H_1')) = ((A'-A,B'-B),(A+A',B+B'-H),(X'-X,Y'-Y),(X+X',Y+Y'-H))$, while A,A',X,X' are randomly chosen and we have that DDH assumption holds. So according to Proposition 4, if the active adversary corrupts a player, and the corrupted player generates input/output with no

knowledge of e_i, t_i, u_i, Protocol B will reject this player. Since the rejected player is immediately excluded from protocol, the only action that the active adversary can take is to control the choices of (e_i, t_i, u_i). It is obvious that Protocol Z outputs $E(a \oplus b, r'')$ even when some of the players are corrupted.

Without loss of generality we assume that a single player j is uncorrupted.

To see that Protocol Z leaks no information on a, b, we configure a simulator for the protocol that has no decryption oracle but takes $E(a, r)$, $E(b, s)$, $E(a \oplus b, r'')$ as input. Without loss of generality we may assume all players are corrupted except for a player j.

For the simulation of corrupted players, the simulator execute the protocol as described in Protocol Z, while setting e_i, t_i, u_i as the adversary chooses.

For the simulation of player j, the only uncorrupted player, the simulator choose $e_j \in_R \{0, 1\}$ and outputs $E(e_j, t_j)$, $E(e_j \oplus a \oplus b, s_j)$ in place of $E(a_j, r_j)$, $E(b_j, u_j)$ respectively, where $E(e_j \oplus a \oplus b, s_j)$ is obtained by

$$E(e_j \oplus a \oplus b, s_j) = \begin{cases} E(a \oplus b, r'') + (u_j G, u_j H) \\ \qquad (a = 0) \\ -E(a \oplus b, r'') + (u_j G, (u_j + 1)H) \\ \qquad (a = 1). \end{cases}$$

To authenticate its input/output by Protocol B, the simulator execute the simulation of Protocol B as in Proposition 1.

For simulating the decryption stage, the simulator outputs $\tilde{a_n} = \bigoplus_{i=j}^{n} e_i$, and generates the proof by executing the simulation for the verifiable decryption protocol.

Since the output of the simulated player k is $E(\bigoplus_{i=j}^{n} e_i, r_n)$ and $E(a \oplus b \oplus \bigoplus_{i=j}^{n} e_i, s_n)$, one obtains $E(a_n \oplus b_n, r'') = E(a \oplus b, r'')$ as the output of the simulation, successfully simulating Protocol Z.

To see the simulated view of the adversary is indistinguishable from that of the real protocol it suffices to see

$$\text{view}_R = (\{(E(a_i, r_i), E(b_i, s_i), \Pi_i\}, \Pi', a_n),$$

and

$$\text{view}_S = (\{(\tilde{E}(a_i, r_i), \tilde{E}(b_i, s_i), \tilde{\Pi}_i\}, \tilde{\Pi}', \tilde{a_n}),$$

are indistinguishable. Here Π_i is the proof of protocol Z in Step 2, and Π' is the proof in Step 3, $\tilde{E}(a_i, r_i)$, $\tilde{E}(b_i, s_i)$ are the outputs of simulated players, $\tilde{\Pi}, \tilde{\Pi}'$ are simulated proofs for Step 2 and Step 3 respectively.

It is obvious that $E(a_i, r_i)$ and $\tilde{E}(a_i, r_i)$ are computationally indistinguishable because of the DDH assumption and so for $E(b_i, s_i)$ and $\tilde{E}(b_i, s_i)$. Π_i and $\tilde{\Pi}_i$ for each i, Π' and $\tilde{\Pi}'$ are zero-knowledge proofs, thus they cannot help distinguish view_R and view_S. a_n is a random bit since at least one of the players in the real protocol is not corrupted. $\tilde{a_n} = \bigoplus_{i=j}^{n} e_i$ is also a random bit because player j is not corrupted in the simulation. Hence view_R and view_S are computationally indistinguishable under the DDH assumption. ■

3.4 Secure Function Evaluation

First note that in the case we restrict our inputs a and b to be binary in protocol Z, it is easy to show that conditional gates can be used to securely evaluate any logic gate, while keeping the the logic gate itself hidden.

Consider a "quadratic form"

$$f_{x,y,z,w}(a, b) = (a \oplus x) + (b \oplus y) + (a \oplus b) \oplus z + w, \quad (5)$$

where $x, y, z, w \in \mathbb{Z}/q\mathbb{Z}$.

It is easy to see that $f_{x,y,z,w}$ can be configured to be any logic gate if one choose $x, y, z, w \in \mathbb{Z}/q\mathbb{Z}$ appropriately.

By having encrypted values $E(x, r_x)$, $E(y, r_y)$, $E(z, r_z)$, and $E(w, r_w)$ the inputs $E(a, r)$, $E(b, s)$, $E(x, r_x)$, $E(y, r_y)$, $E(z, r_z)$, and $E(w, r_w)$, one can securely evaluate f_{xyzw}. Additions are performed by exploiting the homomorphism of the underlying encryption scheme, whereas XOR operations can be performed by using our protocol Z. Thus one can apply any logic gate to encrypted plaintexts while hiding the gate itself.

Informally speaking, a computation is said to be secure if it is private, correct and fair [12], informally these properties are:

Private: No party learns anything more than what can be computed from the output.

Correct: The output received by each party is guaranteed to be the output of the specified function.

Fair: Corrupted parties should receive an output iff honest parties do.

The fairness requirement is usually relaxed in the faulty majority scenario. We assume that the additional unfair information a corrupted party has about the computation's output can be made arbitrarily small in a security parameter k.

A secure computation usually has three stages:

Input Stage: Here the parties enrolled in the protocol

commit to their inputs.

Computation Stage: In the computational stage, the parties evaluate the circuit which describes the function to be evaluated gate by gate. We consider only AND and negation gates, since they are universal.

Output Stage: In this stage, the parties receive their correspondent outputs.

In our protocol we assume an extra stage which happens before the input stage, it is called Setup phase.

Setup Phase: During the setting phase, the players generate the public/private keys for the threshold ElGamal encryption scheme used subsequently in the computation stage.

A Protocol Implementing Secure Multi-Party Computation Our protocol is similar to the one presented in [5] and the security analysis there presented can be straightforwardly modified to show the security and correctness of our protocol.

(1) **Setup Phase** - In this stage, all the players generate the private/public keys of the threshold encryption schemes used in the subsequent stages.

(2) **Input Stage** Each player encrypts his own input by using the ElGamal threshold encryption scheme agreed on during the Setup phase. The players prove in zero-knowledge that they have behaved correctly.

(3) **Computation Stage** During the computation stage, the players evaluate the circuit being computed gate by gate. AND gates can be evaluated by using protocol Z. Negation gates can be easily implemented by exploiting the linearity of our encryption scheme (the players should prove in zero knowledge that they behave correctly).

(4) **Opening Stage** Here, all the players reconstruct the result of the computation. If the number of corrupted players is larger or equal to a half of the players, then fairness becomes an important issue. However, we note that a solution proposed in [17] equally applies to our setting and can be straightforwardly used here to achieve weak fairness.

3.5 Performance Analysis

In this section, we study the performance of the proposed protocol in terms of computational and communication complexities.

We compare our protocol with the other protocols possessing linear communication and computational complexities in the literature that are secure against active adversaries, namely [5] and [17]. We compare the costs of implementing a conditional gate with our protocol and the protocol proposed in [17] to the cost of implementing a multiplication with the protocol proposed in [5].

Table 1 shows a comparison of the required computational effort, and Table 2 shows a comparison of the communications complexities for each XOR gate (multiplication gate in the case of [5]). Here n is the number of participants in the protocol, MLT is the amount of computational effort required for computing XOR (multiplication gates) with honest-but-curious adversaries (without any kind of verification), PRF is the computational cost of the proofs for making the XOR (multiplication) secure against active adversaries and VRF is the cost of verifying those proofs. $M_{\mathbf{Pai}}$, $M_{\mathbf{ElG}}$ are the times required for computing the modulo exponentiation operation in the Paillier encryption and the elliptic multiplication by scalars in the elliptic ElGamal encryption, respectively. We take the protocol proposed in [10] as the distributed decryption scheme in [5].

The modulo exponentiation in Paillier encryption is executed on $\mathbb{Z}/N^2\mathbb{Z}$, the bit length of N is 1024. The elliptic ElGamal encryption is executed on an elliptic curve over $\mathbb{F}_{p^k}$. q is the order of the base point. We consider that the resulting primitives have about the same level of security. Regarding elliptic exponentiation on OEF(Optimal Extension Fields), 174 bit elliptic multiplication by scalars is computed in 0.254 ms on a 500 MHz Alpha 21264 processor if we optimize it according to [1] etc. Regarding modulo exponentiation, the library evaluation by [18] indicates that 1.6 GHz AMD Opteron processor took 28.41 ms to compute 2048 bits RSA decryption. If we take this value as a rough approximation, we may consider $M_{\mathbf{Pai}} \sim 200 M_{\mathbf{ElG}}$. Hence the proposed protocol seems to be the most efficient homomorphic encryption based scheme in terms of computational communication complexities (but it should be remarked that [5] has a better round complexity).

4. Conclusions and Future Works

In this paper, we proposed a protocol to perform secure distributed computations based on the DDH assumption. The performance of our protocol was superior when compared to a previous construction [17] while, at the same time, being *non-optimistic*. Our solution seems applicable when the number of players engaged in the computational is no so large, e.g. secure two-party computations.

The biggest open problem left by this work is to im-

Table 3 Computation time for XOR gate(worst case)

	MLT	PRF	VRF
[5]	$4nM_{\mathbf{Pai}}$	$15nM_{\mathbf{Pai}}$	$13nM_{\mathbf{Pai}}$
[17]	$6nM_{\mathbf{ElG}}$	$18nM_{\mathbf{ElG}}$	$27nM_{\mathbf{ElG}}$
ours	$5nM_{\mathbf{ElG}}$	$10nM_{\mathbf{ElG}}$	$16nM_{\mathbf{ElG}}$

Table 4 Communications traffic for XOR gate(outbound, worst case)

	MLT	PRF
[5]	$3n\|N^2\|$	$4n\|N^2\|$
[17]	$6n(\|p^k\|+1)$	$11n\|q\|$
ours	$5n(\|p^k\|+1)$	$6n\|q\|$

prove the round complexity of the protocol, while preserving its computational efficiency.

References

[1] K. Aoki, F. Hoshino, and T. Kobayashi, "A Cyclic Window Algorithm for ECC Defined over Extension Fields," S. Qing, T.Okamoto, and J. Zhou (Eds.), Proceedings of International Conference on Information and Communication Security (ICICS 2001), LNCS 2229, pp. 62–73, Springer-Verlag, 2001.

[2] D. Chaum, C. Crépeau, and I. Damgård, "Multiparty unconditionally secure protocols," STOC '88.

[3] D.L. Chaum and T.P. Pedersen, "Wallet databases with observers," Advances in Cryptology - CRYPTO '92, LNCS 740, pp. 80–105, Springer-Verlag, 1993.

[4] H. Cohen, A. Miyaji, and T. Ono, "Efficient elliptic curve exponentiation using mixed coordinates," K. Ohta and D. Pei (Eds.), Advances in Cryptology - ASIACRYPT '98, LNCS 1514, pp. 51–65, Springer-Verlag, 1998.

[5] R. Cramer, I. Damgård and J.B. Nielsen, "Multiparty computation from threshold homomorphic encryption," Basic Research in Computer Science (BRICS) RS-00-14, Jun. 2000.

[6] R. Cramer, I. Damgård and B. Schoenmakers, "Proofs of partial knowledge," Advances in Cryptology - CRYPTO '94, LNCS 839, pp. 174–187, Springer-Verlag, 1994.

[7] Y. Desmedt and Y. Frankel, "Threshold cryptosystems," G. Brassard (Ed.), Advances in Cryptology - CRYPTO '89, LNCS 435, pp. 307–315, Springer-Verlag, 1990.

[8] P. Feldman, "A practical scheme for non-interactive verifiable secret sharing," In Proc. of the 28th IEEE Symposium on the Foundations of Computer Science (FOCS), pp. 427–437, IEEE Press, Oct. 1987.

[9] A. Fiat and A. Shamir, "How to Prove Yourself: practical solutions of identification and signature problems," A. M. Odlyzko (Eds.), Advances in Cryptology - CRYPT '86, LNCS 263, pp. 186–194, Springer-Verlag, 1987.

[10] P.-A. Fouque, G. Poupard, and J. Stern, "Sharing decryption in the context of voting or lotteries," Financial Cryptography '00, LNCS 1962, pp. 90–104, Springer-Verlag, 2000.

[11] O. Goldreich, S. Micali, and A. Widgerson, "How to play any mental game," STOC '87, pp. 218–229, 1987.

[12] O. Goldreich, "Secure Multi-Party Computation," Working Draft, Version 1.1, 1998. Available at `http://www.wisdom.weizmann.ac.il/~oded/pp.html`.

[13] D. Grigoriev and I. Ponomarenko, "Homomorphic public-key cryptosystems over groups and rings," `arXiv:cs.CR/0309010 v1`, 8 Sep. 2003.

[14] T. P. Pedersen, "A threshold cryptosystem without a trusted party," Advances in Cryptology - EUROCRYPT '91, LNCS 547, pp. 522–526, Springer-Verlag, 1991.

[15] T. P. Pedersen, "Non-interactive and information-theoretic secure verifiable secret sharing," J. Feigenbaum (Ed.), Advances in Cryptology - CRYPTO '91, LNCS 576, pp. 129–140, Springer-Verlag, 1991.

[16] D. Pointcheval and J. Stern, "Security Proofs for Signature Schemes," U. Maurer (Ed.), Advances in Cryptology - EUROCRYPTO '96, LNCS 1070, pp. 387–398, Springer-Verlag, 1996.

[17] B. Schoenmakers and P. Tuyls, "Practical Two-Party Computation Based on the Conditional Gate," P.J.Lee (Ed.), ASIACRYPT 2004, LNCS 3329, pp. 119–204, Springer-Verlag, 2004.

[18] W. Dai, `http://www.eskimo.com/~weidai/benchmarks.html`, 2004.

[19] A.C. Yao, "How to generate and exchange secrets," In Proc. of the 27th IEEE Symp. on Foundations of Computer Science (FOCS '86), IEEE Press, pp. 162–167, 1986.

Appendix

In this appendix our goal is to present Protocol B, which is essential to prove the security of Protocol Z. Roughly speaking, through Protocol B the players engaged in Z can prove in zero-knowledge that they acted correctly. Our protocol is a modification of earlier results in the literature (mostly [6], [3], and [16]).

AND-OR Proof: Consider the language defined by

$$\mathsf{ANDORDL} \stackrel{\text{def}}{=} \{(G,H,G_0,H_0,G_1,H_1,G_0',H_0',G_1',H_1') \in \mathcal{G}_q^{10} \mid$$

$$(\log_G G_0 = \log_H H_0 \wedge \log_G G_0' = \log_H H_0') \vee$$

$$(\log_G G_1 = \log_H H_1 \wedge \log_G G_1' = \log_H H_1')\}.$$

This is the language that the discrete logarithms of G_b and G_b' to the base G are respectively equal to the discrete logarithms of H_b and H_b' to the base H for $b \in \{0,1\}$.

The Protocol B illustrates an honest verifier zero-knowledge proof of knowledge for a witness on the membership of ANDORDL. For the convenience of the readers here we describe Protocol B again.

[Proposition 1] **(Simulatability)** Define $\text{view}_R = (z_0, z_1, c_0, c_1)$. There exists a simulator that, on input $(G, H, G_0, H_0, G_1, H_1, G'_0, H'_0, G'_1, H'_1) \in \text{ANDORDL}$, outputs view_S which is perfectly indistinguishable from view_R in expected polynomial time in κ under the random oracle model.

PROOF

A simulator performs the following procedure for input
$(G, H, G_0, H_0, G_1, H_1, G'_0, H'_0, G'_1, H'_1) \in \text{ANDORDL}$.

(1) Choose $\tilde{r}, \tilde{v}, \tilde{c}_0, \tilde{c}_1 \in_R \mathbb{Z}/q\mathbb{Z}$.

(2) Generate
$e = \mathcal{H}_0(G||H||G_0||H_0||G_1||H_1||G'_0||H'_0||G'_1||H'_1)$.

(3) Compute $\tilde{R}^0_G = \tilde{r}G + \tilde{c}_0(eG_0 + G'_0)$, $\tilde{R}^0_H = \tilde{r}H + \tilde{c}_0(eH_0 + H'_0)$, $\tilde{R}^1_G = \tilde{v}G + \tilde{c}_1(eG_1 + G'_1)$, $\tilde{R}^1_H = \tilde{v}H + \tilde{c}_1(eH_1 + H'_1)$, $\tilde{z}_0 = \tilde{r}$, $\tilde{z}_1 = \tilde{v}$.

(4) Output $\text{view}_S \stackrel{\text{def}}{=} (\tilde{z}_0, \tilde{z}_1, \tilde{c}_0, \tilde{c}_1)$.

Here we assume $\mathcal{H}_0$ is an ideal random function that maps $\{0,1\}^*$ to $\mathbb{Z}/q\mathbb{Z}$. We also assume $\mathcal{H}_1$ is a random function that maps $\{0,1\}^*$ to $\mathbb{Z}/q\mathbb{Z}$, however, it returns $\tilde{c}_0 + \tilde{c}_1$ when the string $\tilde{R}^0_G||\tilde{R}^0_H||\tilde{R}^1_G||\tilde{R}^1_H$ is input. Then it is clear view_S is accepted by V and view_R and view_S are perfectly indistinguishable. ■

[Proposition 2] **(Soundness)** If P is successful in producing (z_0, z_1, c_0, c_1) accepted by V, P has witnesses $b \in \{0,1\}$ and $s, t \in \mathbb{Z}/q\mathbb{Z}$ s.t. $G_b = sG$, $H_b = sH$, $G'_b = tG$, and $H'_b = tH$ with overwhelming probability assuming the hardness of discrete logarithm problem under the random oracle model.

PROOF

Set $\alpha_i = \log_G G_i$, $\beta_i = \log_H H_i$, $\alpha'_i = \log_G G'_i$, and $\beta'_i = \log_H H'_i$ for $i = 0, 1$. Then, it is considered the following three cases;

Case 1: $\alpha_b = \beta_b (= s) \wedge \alpha'_b = \beta'_b (= t)$ for $b \in \{0,1\}$, that is, P is honest.

Case 2: $(\alpha_0 \neq \beta_0 \wedge \alpha_1 \neq \beta_1) \vee (\alpha'_0 \neq \beta'_0 \wedge \alpha'_1 \neq \beta'_1)$, but $\alpha_b e + \alpha'_b = \beta_b e + \beta'_b$ for $b \in \{0,1\}$.

Case 3: $(\alpha_0 \neq \beta_0 \wedge \alpha_1 \neq \beta_1) \vee (\alpha'_0 \neq \beta'_0 \wedge \alpha'_1 \neq \beta'_1)$ and $\alpha_0 e + \alpha'_0 \neq \beta_0 e + \beta'_0 \wedge \alpha_0 e + \alpha'_0 \neq \beta_0 e + \beta'_0$.

In Case 3, it is obvious from Lemma ?? that V rejects the proof generated by P. In Case 1, if the proof generated by P is acceepted, b and $se + t$ can be extracted by the knowledge extractor in Lemma 3. Thus we separate the analysis of Case 1 in two subcases. The first is the case where P has all of witnesses b, s, and t. The other is when P does not have s and t though it has $se + t$. We show that a probabilistic polynomial-time adversary $\mathcal{A}$ breaks the discrete logarithm problem using $\tilde{P}$ who outputs the correct proof for ANDORDL assuming the latter case.

Let $(\mathcal{G}_q, G, \tilde{G}) \leftarrow \mathsf{G}^{\mathsf{DL}}(1^\kappa)$ be an instance of the discrete logarithm problem. Denote $\log_G \tilde{G}$ by x. Let $\mathcal{H}_0$ be an ideal random function that maps $\{0,1\}^*$ to $\mathbb{Z}/q\mathbb{Z}$. $\mathcal{A}$ and $\tilde{P}$ are allowed to access to $\mathcal{H}_0$. Without loss of generality, we can see that $\tilde{P}$ is an oracle that inputs

$$(G, H, G_0, H_0, G_1, H_1, G'_0, H'_0, G'_1, H'_1) \in \text{ANDORDL}, \tag{6}$$

which is the input of Protocol B, and $e \in \mathbb{Z}/q\mathbb{Z}$ and outputs $w = se + t$ with non-negligible probability in κ, where $s = \log_G G_b = \log_H H_b$, $t = \log_G G'_b = \log_H H'_b$, and $b \in \{0,1\}$. $\mathcal{A}$ performs the following procedure.

(1) Inputs $(\mathcal{G}_q, G, \tilde{G})$.

(2) Chooses $\tilde{b} \in_R \{0,1\}$ and $\tilde{z}, \tilde{z}', \tilde{s}, \tilde{t} \in_R \mathbb{Z}/q\mathbb{Z}$.

(3) Computes $\tilde{H} = \tilde{z}G$, $\tilde{G}_{\tilde{b}} = \tilde{s}\tilde{G}$, $\tilde{H}_{\tilde{b}} = \tilde{z}\tilde{G}_{\tilde{b}}$, $\tilde{G}'_{\tilde{b}} = \tilde{t}\tilde{G}$, and $\tilde{H}'_{\tilde{b}} = \tilde{z}\tilde{G}'_{\tilde{b}}$.

(4) Chooses $\tilde{G}_{1-\tilde{b}}, \tilde{H}_{1-\tilde{b}}, \tilde{G}'_{1-\tilde{b}}, \tilde{H}'_{1-\tilde{b}} \in_R \mathcal{G}_q$.

(5) Sends $ins_{\mathcal{A}} \stackrel{\text{def}}{=} (G, \tilde{H}, \tilde{G}_0, \tilde{H}_0, \tilde{G}_1, \tilde{H}_1, \tilde{G}'_0, \tilde{H}'_0, \tilde{G}'_1, \tilde{H}'_1)$ to $\mathcal{H}_0$ and obtains $\tilde{e} \in \mathbb{Z}/q\mathbb{Z}$ from it.

(6) Sends $ins_{\mathcal{A}}$ and $\tilde{e}$ to $\tilde{P}$ and obtain $\tilde{w} \in \mathbb{Z}/q\mathbb{Z}$ from it.

Then, since $\tilde{w}$ is equal to $(\tilde{s}x)\tilde{e} + \tilde{t}x$ with non-negligible probability, $\mathcal{A}$ can obtain x with high probability. $\tilde{P}$ always works because $ins_{\mathcal{A}} \in \text{ANDORDL}$ and e and $\tilde{e}$ are perfectly indistinguishable.

Finally, we show the success probability of probabilistic polynomial-time adversary $\mathcal{A}'$ that aims at generating Case 2 is negligible. Let $\mathcal{O}_B$ be an oracle that executes Protocol B. We denote by q_{H} the maximum number of access that $\mathcal{A}'$ has to $\mathcal{O}_B$. Note that, q_{H} is polynomially bounded in κ. Since challenge e is randomly chosen by $\mathcal{H}_0$ after α_b, β_b, α'_b, and β'_b are publicly committed, the success probability of an adversary $\mathcal{A}'$ that is allowed to access $\mathcal{O}_B$ only once is exactly $1/q$. Namely, the success probability, $\text{Win}_{\mathcal{A}'}$, of $\mathcal{A}'$ that access to $\mathcal{O}_B$ q_{H}-times is at most $1 - (1 - \frac{1}{q})^{q_{\mathsf{H}}}$. Thus, it is obtained

$$\begin{aligned} \text{Win}_{\mathcal{A}'} &= 1 - (1 - \tfrac{1}{q})^{q_{\mathsf{H}}} \\ &= 1 - (1 - \tfrac{q_{\mathsf{H}}}{q} + \tfrac{q_{\mathsf{H}}(q_{\mathsf{H}}-1)}{2q^2} - \cdots) \\ &< \tfrac{q_{\mathsf{H}}}{q}, \end{aligned}$$

and this is negligible. ■