A Secure Scalar Product Protocol and Its Applications to Computational Geometry

Bo YANG
School of Computer Science, Shaanxi Normal University, Xi'an, 710062, PR China
Email: byang@snnu.edu.cn

Chung-Huang YANG
Graduate Institute of Information and Computer Education, National Kaohsiung Normal University, Tai-wan
Email: chyang@nknucc.nknu.edu.tw

Yong YU
School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu, 610054, PR China
Email: yuyong@uestc.edu.cn

Dan XIE
School of Computer Science, Shaanxi Normal University, Xi'an, 710062, PR China
Email: xied111@163.com

***Abstract*—A secure scalar product protocol is a type of specific SMC problem, and has found various applications in many areas such as privacy-preserving data mining, privacy-preserving cooperative statistical analysis, and privacy-preserving geometry computation.**

In this paper, we firstly extend to a solution of homomorphic-encryption based secure scalar product protocol such that it enables the scheme to be used in distributed decryption, and to deal with negative vectors. Secondly, we propose two-party secure computation of a public Boolean function on private inputs of each party. Thirdly, we describe two applications of our secure scalar product protocol to computational geometry: determining securely location of a point to a directed line segment, and conditional oblivious transfer based on the relation between a private point and a private directed line.

***Index Terms*—Secure multi-party computation, computational geometry, secure scalar product protocol, conditional oblivious transfer**

I. Introduction

In secure multiparty computation (SMC) [1-2], a set of parties want to jointly compute a function of their inputs over the Internet or any computer network without revealing to the other participants any information about their private inputs. A secure scalar product protocol [3-8] is a type of specific SMC problem, and its goal is that two parties jointly compute the scalar product of their private vectors, but no party will reveal any information about his private vector to another party. As a building block, secure scalar product protocol has found various applications in many areas such as privacy-preserving data mining [3-4], privacy-preserving cooperative statistical analysis [5], and privacy-preserving geometry computation [6].

A secure scalar product protocol deals with the following problem. Let Alice have a private vector $X=(x_1,\cdots,x_l)$ and Bob has another private vector $Y=(y_1,\cdots,y_l)$, They compute cooperatively

$$u+v=X\cdot Y=\sum_{i=1}^{l}x_iy_i,$$

where u is a uniformly distributed random number known by Alice and v is a dependent uniformly distributed random number known by Bob.

To solve this problem, many solutions have been proposed [3-8], and a nice overview of the problem and the security properties of some solutions were given in [3].

In [3], it was shown that if a vector has low support (the support of a vector is defined by the number of nonzero elements in this vector), another party will learn half elements of this vector, further learn the whole vector with a high probability. Then the authors proposed a new secure scalar product protocol, which is based on homomorphic encryption with plaintext space Z_m for some large m. The protocol is secure in the semi-honest model [2], assuming that $X,Y\in Z_\mu^l$, in which $\mu=\left\lfloor\sqrt{m/l}\right\rfloor$.

However, all of these protocols considered only positive integer vectors. How to deal with negative vectors is an important and interesting issue because of their broad applications in reality.

In this paper, we make an extension to the solution in [3], which enables the scheme to be used in distributed decryption, and to handle negative vectors. Furthermore, we describe two-party secure computation of a public Boolean function on private input of each party. Finally,

doi:10.4304/jcp.8.8.2018-2026

we propose two applications of our secure scalar product protocol to computational geometry. The former is to determine which side a private point P_0, held by a party (Alice), lies in a directed line segment $\overrightarrow{P_1P_2}$, held by another party (Bob), but neither of them wants to reveal its secret to another party. The latter is a conditional oblivious transfer protocol based on the relation between a point P_0, held by a party (Alice), and a directed line $\overrightarrow{P_1P_2}$, held by another party (Bob). Assume Alice has two secret messages s_0, s_1, and wishes (obliviously to her) to transfer one of them to Bob depending on which side P_0 lies in $\overrightarrow{P_1P_2}$, but will not learn Bob's private directed line and the location relation of one's own point with Bob's private directed line.

II. Some Fundamental Concepts and Building Blocks

A. Some Concepts in Computational Complexity

(1) Negligible Function. A function $\mu \mapsto (0,1)$ is called negligible if for every positive polynomial $p(x)$, and all sufficiently large n's, $\mu(n) < 1/p(n)$.

(2) Probability Ensembles. A probability ensemble indexed by $S \subseteq \{0,1\}^*$ is a family $\{X_w\}_{w\in S}$, so that each X_w is a random variable (or distribution) which ranges over (a subset of) $\{0,1\}^{poly(|w|)}$. Typically, we consider $S = \{0,1\}^*$ and $S = \{1^n : n \text{ is a naural number}\}$.

(3) Identically distributed. We say that two ensembles, $X \overset{def}{=} \{X_w\}_{w\in S}$ and $Y \overset{def}{=} \{Y_w\}_{w\in S}$, are identically distributed, and denoted by $X \equiv Y$, if for every $w \in S$ and every α

$$\Pr[X_w = \alpha] = \Pr[Y_w = \alpha] \qquad (1)$$

(4) Computationally Indistinguishable. Two ensembles, $X \overset{def}{=} \{X_w\}_{w\in S}$ and $Y \overset{def}{=} \{Y_w\}_{w\in S}$, are computationally indistinguishable, and denoted by $X \overset{C}{\equiv} Y$, if for every non-uniform distinguisher D there exists a negligible function $\mu(\cdot)$ such that for every $a \in \{0,1\}^*$,

$$|\Pr[D(X_w(a,n)) = 1] - \Pr[D(Y_w(a,n)) = 1]| < \mu(n) \qquad (2)$$

B. The Semi-honest Model

We assume that all parties are semi-honest. Roughly speaking, a semi-honest party is one who follows the protocol properly with the exception that it keeps a record of all its intermediate computations and might derive the other parties' inputs from the record. A protocol is private in the semi-honest model if whatever is obtained by a party participating in the protocol also can be computed from its input and output only. This assumption is formalized by simulation paradigm as follows [2].

Let $f = (f_1, f_2)$ be a probabilistic polynomial-time functionality and $\prod$ be a two-party protocol for computing f. The view of the first party during an execution of $\prod$ on the input (x, y), denoted by $view_1^{\Pi}(x, y)$, is $(x, r^1, m_1^1, \cdots, m_t^1)$, where r^1 represents the outcome of the first party's internal coin tosses, and m_i^1 denotes the ith message received during executing of $\prod$. The output of the first party during executing of $\prod$ on the input (x, y), denoted $output_1^{\Pi}(x, y)$, is implicit in the party's view of the execution. The view and output of the second party can be defined under analogous notation.

Definition:For a functionality $f = (f_1, f_2)$, $\prod$ privately computes f (or $\prod$ is secure) if there exist two probabilistic polynomial-time algorithms, denoted by S_1 and S_2 such that

$$\{S_1(x, f_1(x,y)), f_2(x,y)\}_{x,y} \overset{C}{\equiv} \{view_1^{\Pi}(x,y), output_2^{\Pi}(x,y)\}_{x,y},$$

$$\{f_1(x,y), S_2(y, f_2(x,y))\}_{x,y} \overset{C}{\equiv} \{output_1^{\Pi}(x,y), viewt_2^{\Pi}(x,y)\}_{x,y}$$

where $\overset{C}{\equiv}$ denotes the computational indistinguishability, $view_1^{\Pi}(x,y)$, $view_2^{\Pi}(x,y)$, $output_1^{\Pi}(x,y)$ and $output_2^{\Pi}(x,y)$ are random variables, defined as a function of the same random execution. In particular, $output_1^{\Pi}(x,y)$ is fully determined by $view_i^{\Pi}(x,y)$.

C. Homomorphic Encryption Schemes

An encryption scheme is homomorphic [9-10] if for some operations $\square$ and $*$, $E_k(x) \square E_k(y) = E_k(x * y)$, where x and y are two elements from the message space and k is the key. If $*$ is an additive (multiplicative) operator, the encryption scheme is said to be additive (multiplicative) homomorphic. A useful property of homomorphic encryption schemes is that an operation can be conducted based on the encrypted data without decrypting them. If an encryption scheme is additive and multiplicative homomorphic simultaneously, it is said to be fully homomorphic [17-23]. A fully homomorphic encryption scheme can be used to fulfill the secure scalar product protocol, and it can make significant improvement in communication complexity, but it would suffer huge computational overhead with current state of the art.

Therefore, we will not adopt the fully homomorphic encryption scheme, instead of adopting an only additive homomorphic encryption scheme, I. Damgärd and M. Jurik's cryptosystem in Z_{n^s}, as our building block in this paper. The encryption scheme is as follows [10]:

Key generation: Let $n = pq$ be the RSA-modulo with $p = 2p' + 1$, $q = 2q' + 1$, where p, q, p', q' are primes, $g \in Q_n$, the group of all squares of Z_n^*, $\alpha \in Z_\tau$, where

$\tau = p'q' = |Q_n|$, $h = g^{\alpha} \bmod n$. The public key is (n, g, h) and the private key is α .

Encryption: For $m \in Z^+$, choose an integer $s > 0$ such that $m \in Z_{n^s}$. The encryption is

$$E_h(m; r) = (g^r \bmod n, (h^r \bmod n)^{n^s} (n+1)^m \bmod n^{s+1}),$$

where $r \in_R Z_n$.

Decryption: For a ciphertext $c = (G, H)$, s can be deduced from the length of c (or attached to the encryption), and the message is computed from

$$m = L_s(H(G^{\alpha} \bmod n)^{-n^s})$$

$$= L_s((g^{\alpha r} \bmod n)^{n^s} (n+1)^m (g^{r\alpha} \bmod n)^{-n^s})$$

$$= L_s((n+1)^m \bmod n^{s+1}) = m \bmod n^s$$

where L_s is an algorithm for calculating m from $(n+1)^m \bmod n^{s+1}$ [11], whose computational cost is $O(s^2 |n^s|^2)$.

Scalaring: For a ciphertext $c = E_h(m, r) = (G, H)$, the scalaring is done by computing $c' = E_h(km, kr) = (G^k, H^k)$ for $k \in Z_N^*$. If $m = 0$, the scalaring operation does not change the content of the ciphertext.

The scheme is semantically secure under two assumptions, the CRA (Composite Residuosity Assumption) and the composite DDH assumption. The CRA states that it is computationally infeasible to distinguish whether an element $z \in Z_{n^2}^*$ is a residue or not, while the composite DDH assumption says that it is computationally infeasible to distinguish two tuples $(n, g, g^a \bmod n, g^b \bmod n, g^{ab} \bmod n)$ and $(n, g, g^a \bmod n, g^b \bmod n, y)$, where $y \in_R Q_n$.

The scheme is additively homomorphic since

$$E_h(m_1; r_1) \cdot E_h(m_2; r_2)$$

$$= (g^{r_1} \bmod n, (h^{r_1} \bmod n)^{n^s} (n+1)^{m_1} \bmod n^{s+1}) \cdot$$

$$(g^{r_2} \bmod n, (h^{r_2} \bmod n)^{n^s} (n+1)^{m_2} \bmod n^{s+1})$$

$$= (g^{r_1+r_2} \bmod n, (h^{r_1+r_2} \bmod n)^{n^s} (n+1)^{m_1+m_2} \bmod n^{s+1})$$

$$= E_h(m_1 + m_2; r_1 + r_2)$$

D. Re-encryption

Let (G, H) be a ciphertext, (G_0, H_0) be the ciphertext of the plaintext 0, then the plaintext of $(G', H') = (G, H) \cdot (G_0, H_0)$ be the same as that of (G, H) .

The computational cost of Re-encryption is twofold that of the original scheme.

In the following we will denote $E(m)$ as $E_h(m; r)$ when no misunderstanding is possible.

E. Distributed key Generation

Alice (Bob) picks $\alpha_1(\alpha_2)$ at random and publishes $h_1 = g^{\alpha_1}$ ($h_2 = g^{\alpha_2}$) along with a zero-knowledge proof of knowledge of h_1's (h_2's) discrete logarithm. The public key is $h = h_1 h_2$, and the private key is $\alpha = \alpha_1 + \alpha_2$.

F. Distributed Decryption

Given an encrypted message (G, H) , Alice publishes $G_1 = G^{\alpha_1} (\bmod n)$, and proves its correctness by showing the equality of logarithm of h_1 and G_1 , that is, proves (g, G, h_1, G_1) satisfying the relation

$$R_{DH} = \{((g, G, h_1, G_1), \alpha_1) \mid$$

$$h_1 = g^{\alpha_1} \bmod n \wedge G_1 = G^{\alpha_1} \bmod n\}$$

The proof of the relation R_{DH} can be done in zero-knowledge [12], whose communication cost is $2\log n + \log \tau$. Similarly,Bob publishes $G_2 = G^{\alpha_2} \bmod n$, and proves (g, G, h_2, G_2) satisfying the relation

$$R_{DH} = \{((g, G, h_2, G_2), \alpha_2) \mid$$

$$h_2 = g^{\alpha_2} \bmod n \wedge G_2 = G^{\alpha_2} \bmod n\}$$

The plaintext can be derived by computing

$$m = L_s(H(G_1 G_2 \bmod n)^{-n^s})$$

G. Mix Network (MN)

Intuitively, a mix network [13] is a multi-party protocol that takes as input a list of ciphertext items and from this produces a new, random list of ciphertext items such that there is a one-to-one correspondence between the underlying plaintexts of input and output items. In other words, the underlying output plaintexts represent a random permutation of the underlying input plaintexts. The security of a mix network is characterized by the infeasibility for an adversary of determining which output items correspond to which input items.

Our mix network is similar to the one in [13]. Let the input to the mix network be a sequence of ciphertexts $(G_1, H_1), (G_2, H_2), \cdots, (G_k, H_k)$, and the output is a random permutation and re-encryption of the inputs, namely a sequence

$$(G'_{\sigma(1)}, H'_{\sigma(1)}), (G'_{\sigma(2)}, H'_{\sigma(2)}), \cdots, (G'_{\sigma(k)}, H'_{\sigma(k)})$$

where (G'_i, H'_i) represents a random re-encryption of (G_i, H_i) , and σ is a random permutation on k elements. The computational cost of MN is k -times that of the employed encryption scheme.

H. Shuffle

A shuffle [14] of ciphertexts $C_1, C_2, \cdots, C_k$ is a new set of ciphertexts $C'_1, C'_2, \cdots, C'_k$, such that both sets of the ciphertexts have the same plaintexts. If we are working with I. Damgärd and M. Jurik's cryptosystem, $C_1, C_2, \cdots, C_k$ can be shuffled by selecting a permutation σ and setting

$$C'_1 = C_{\sigma(1)} E(0), C'_2 = C_{\sigma(2)} E(0), \cdots, C'_k = C_{\sigma(k)} E(0) .$$

It is impossible for anybody else to see which permutation was used in the shuffle because of the semantic security of the cryptosystem. On the other hand this also means that anybody else cannot check if we did make a correct shuffle directly.

The computational cost of a shuffle is k -times that of the employed encryption scheme.

I. Distributed Plaintext Equality Test (ΠET)

Our protocol is similar with the one in [13]. Let $C_1 = (G_1, H_1)$ and $C_2 = (G_2, H_2)$ be two ciphertexts with respective underlying plaintexts m_1 and m_2 encrypted under the same public key h. Two participants, Alice and Bob, jointly determine whether $m_1 = m_2$. Consider the ciphertext $c = (\frac{G_1}{G_2}, \frac{H_1}{H_2})$, Alice (Bob) selects a blinding factor $z_1(z_2)$, and computes $c^{z_1}(c^{z_2})$. Then both of them jointly compute $c' = c^{z_1}c^{z_2} = (G', H')$ as blinded c. According to distributed decryption algorithm, they publish $G_1' = (G')^{\alpha_1} \bmod n$ and $G_2' = (G')^{\alpha_2} \bmod n$, respectively, and jointly judge whether $H'(G_1'G_2' \bmod n)^{-n^s} = 1$ or not. If it does, they conclude that $m_1 = m_2$, else $m_1 \neq m_2$. We denote the protocol above by $PET(C_1, C_2)$.

The computational cost of ΠET is the sum of the one of employed encryption with $4\log n$ modular multiplications, then subtract the one of L_s.

III. A Secure Scalar Product Protocol in Field

In the following, we will make an improvement on the solution in [3], such that it can be realized based on I. Damgärd and M. Jurik's cryptosystem in Z_{n^s}, where $s > 0$ is an integer such that any element involved in the computation are in Z_{n^s}.

In our applications, we assume $N_0 = \lfloor \log n^s \rfloor + 1$, that is, n^s is of N_0 bits length. We say $d < 0$ if $n^s / 2 < d < n^s$, thus the most significant bit in elements of Z_{n^s} denotes its sign, that is, 1 denotes the negative element, 0 is positive.

Protocol 1 (Secure scalar product protocol)

Inputs: Alice has a private vector $X = (x_1, \cdots, x_l)$ and Bob has another private vector $Y = (y_1, \cdots, y_l), x_i, y_i \in Z_{n^s}$ for $i = 1, 2, \cdots, l$.

Outputs: Alice gets u and Bob gets v, satisfying $u, v \in Z_{n^s}$ and $u + v = X \cdot Y \bmod n^s$.

(1) Alice computes $c_i = E(x_i)$ and sends c_i to Bob, for $i = 1, 2, \cdots, l$.

(2) Bob computes $w = \prod_{i=1}^{l} c_i^{y_i}$, generates a random plaintext $v \in Z_{n^s}$, computes $w' = wE(v) = (G, H)$, sends w' to Alice, publishes $G_2 = G^{\alpha_2} \bmod n$ and proves (g, G, h_2, G_2) satisfying the relation

$$R_{DH} = \{((g, G, h_2, G_2), \alpha_2) \mid h_2 = g^{\alpha_2} \bmod n \wedge G_2 = G^{\alpha_2} \bmod n\}$$

(3) Alice compute $G_1 = G^{\alpha_1} \bmod n$, $u = L_s(H(G_1G_2 \bmod n)^{-n^s})$, and obtains $u = (\sum_{i=1}^{l} x_i y_i + v) \bmod n^s$.

(4) Bob computes $v = n^s - v$.

Security: The input of Bob is Y, the view of Bob during an execution of the protocol is $E(x_1), \cdots, E(x_n)$. Because of the semantic security property of employed encryption scheme, Bob will not learn any information about X, thus Alice's security is proven.

For Bob's security, we can construct a simulator S_1 to simulate the view of Alice. The input, output, and view of Alice are $X, u, view_1^{\Pi}(X, Y) = \{w', prooftext\}$, respectively, where *prooftext* is Alice's view in proof of R_{DH}. Because the proof of R_{DH} can be done in zero-knowledge, there exists a simulator S_{DH} to perfectly simulate the view of the verifier (Alice here). Let the output of S_{DH} be R'. $\{prooftext\}$ and R' are identically distributed.

S_1 takes as input (X, u), and proceeds as follows.

(1) generates randomly a vector $Y' = (y_1', \cdots, y_l')$, compute v' satisfying $u + v' = X \cdot Y' \bmod N$ and $v' < N$.

(2) computes

$$c_i' = E(x_i),\ w'' = (\prod_{i=1}^{n} (c_i')^{y_i'}) \bmod N^2$$
$$w''' = ((w''E(-v')) \bmod N^2.$$

(3) outputs $S_1(X, u) = \{w'''\} \bigcup R'$.

It can be verified that $D(w''') = u = D(w')$, that is, w''' and w' are two ciphertexts with the same plaintext. So $\{w'''\}$ and $\{w'\}$ are identically distributed. $\{S_1(X, u)\}$ and $\{view_1^{\Pi}(X, Y)\}$ are identically distributed.

Note: if $\{A\}$ and $\{B\}$ are identically distributed, and $\{C\}$ and $\{D\}$ are identically distributed, it is easy to prove that $\{A, C\}$ and $\{B, D\}$ are identically distributed.

Resource analysis: During an execution of the protocol, Alice and Bob obtain 1 ciphertext and n ciphertexts, respectively. Besides, $2\log n + \log \tau$ bits are needed in proof of R_{DH}, therefore the communication cost is $[(l+1)(s+2)+2]\log n + \log \tau$ bits.

Alice needs to perform l encryptions and 1 decryption, while Bob needs to compute l multiplications and 1 encryption.

If the elements of vectors are negative, we can transform negative elements to positive elements, then perform scalar product of positive elements. For example, we would like to compute the scalar product of two vectors $(-2, 3, -6, 7)$ and $(4, -5, 2, -6)$ in Z_{15}, we can firstly transform them to $(13, 3, 9, 7)$ and $(4, 10, 2, 9)$, then compute

$$(13, 3, 9, 7) \cdot (4, 10, 2, 9) \equiv 13 (\bmod 15)$$

It can be verified that

$(-2,3,-6,7)\cdot(4,-5,2,-6)\equiv 13(\mathrm{mod}15)$

The result remains unchanged.

IV. Two-party Computation of a Boolean Function

In [15], a computation of a symmetric Boolean function was given, a symmetric Boolean function is a Boolean function whose output depends only on the number of 1's in its input. The solution is given under the situation where any secret is shared among a set of users, it is easy to compute sharings of $[a+b]_p$ and $[a\cdot b]_p$ from $[a]_p$ and $[b]_p$, where $[a]_p$ denotes a secret sharing of $a\in F_p$ over F_p, the multiplication is done by unbounded fan-in multiplication [16].

In the case of two parties involved, the addition and multiplication of two encryptions can not be done simultaneously, unless there is a fully homomorphic encryption algorithm available. In the following, based on mix network, we give a computation of any Boolean function in the case of two parties involved, and assume that there is not a fully homomorphic encryption algorithm available.

Let $\phi(a_1,\cdots,a_l)$ be a public Boolean function where $a_i\in\{0,1\}(i=1,\cdots,l)$. Two parties, Alice and Bob, want to compute ϕ on their private inputs $\{a_{i_1},\cdots,a_{i_{l_1}}\}$ and $\{a_{j_1},\cdots,a_{j_{l_2}}\}(l_1+l_2=l)$, respectively. The truth table T, with $2^l\times(l+1)$ elements, of ϕ can be created and made public. For example, TABLE I (named T_1 below) is a truth table of bitwise sum of u,v in which $z_i=u_i+v_i$, and c_i is the carry bit.

TABLE I.
THE TRUTH TABLE OF BITWISE SUM $z_i=u_i+v_i$

u_i	v_i	c_{i-1}	c_i	z_i
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

The protocol of computing ϕ is as follows, in which E is I. Damgärd and M. Jurik's cryptosystem with public key h.

Protocol 2 (Secure computation of a public Boolean function)

Inputs: The public truth table T, Alice has private inputs $(a_{i_1},\cdots,a_{i_{l_1}})$ and Bob has private inputs $\left(a_{j_1},\cdots,a_{j_{l_2}}\right)$.

Outputs: An encrypted Boolean function value $E(\phi(a_1,\cdots,a_l))$.

(1) Alice applies MN to T (using y as the encryption key), obtains T' and publishes it.

(2) Bob shuffles T', obtains T'' and publishes it.

(3) Using y as the encryption key, Alice encrypts $(a_{i_1},\cdots,a_{i_{l_1}})$, and obtains $\{E(a_{i_1}),\cdots,E(a_{i_{l_1}})\}$, Bob encrypts $(a_{j_1},\cdots,a_{j_{l_2}})$, and obtains $\{E(a_{j_1}),\cdots,E(a_{j_{l_2}})\}$.

(4) for $i=1$ to 2^t do {
if $PET(E(a_{i_1}),T''[i,i_1])=1,\cdots,$
and $PET(E(a_{i_{l_1}}),T''[i,i_{l_1}])=1$
and $PET(E(a_{j_1}),T''[i,j_1])=1,\cdots,$
and $PET(E(a_{j_{l_2}}),T''[i,j_{l_2}])=1$,
then return $T''[i,l+1]$ }.

If the function ϕ is a bitwise sum of u,v, the step (4) in Protocol 2 is modified as that Protocol 4 shows.

Security: The security of Protocol 2 can be derived from the security of the employed building blocks such as *MN*, shuffle and *ΠET*.

Resource analysis: Because all $T,T',T'',\{E(a_{i_1}),\cdots,E(a_{i_{l_1}})\},\{E(a_{j_1}),\cdots,E(a_{j_{l_2}})\}$ are public, it is not necessary to consider the communication cost.

Because step 1~3 can be precomputed, the computational cost is dedicated by step 4, which needs $2^l\times l$ PET. Thus the computational cost of Protocol 2 is $2^l\times l$ times that of *ΠET*.

V. Two Applications in Computational Geometry

In this section, we give two applications of the secure scalar product in secure computational geometry.

A. Secure Location of a Point to a Directed Line Segment

Suppose that there is a triangle in the plane with the vertices $P_1(x_1,y_1),P_2(x_2,y_2)$ and $P_0(x_0,y_0)$. Then the signed area of the triangle is half of the determinant

$$D(P_1,P_2,P_0)=\begin{vmatrix} x_1 & y_1 & 1\\ x_2 & y_2 & 1\\ x_0 & y_0 & 1\end{vmatrix}$$

where the sign of $D(P_1,P_2,P_0)$ is positive if and only if (P_1,P_2,P_0) forms a counterclockwise cycle, i.e. P_0 lies in the left of directed line segment $\overrightarrow{P_1P_2}$, and negative if and only if (P_1,P_2,P_0) forms a clockwise cycle, i.e. P_0 lies in the right of directed line segment $\overrightarrow{P_1P_2}$ [24].

$D(P_1,P_2,P_0)$ can be substituted for the z coordinate of the cross product $\overrightarrow{P_1P_2}\times\overrightarrow{P_1P_0}$. We assume that there are two parties, Alice and Bob. Alice holds a private point $P_0(x_0,y_0)$ and Bob holds a private directed line segment with direction from $P_1(x_1,y_1)$ to $P_2(x_2,y_2)$, where $(x_i,y_i)\in(Z_{n^s})^2(i=0,1,2)$. They want to determine which side P_0 lies in the directed line segment $\overrightarrow{P_1P_2}$, but neither of them wants to reveal its private point to another party.

The protocol is as follows.

Protocol 3 (Secure location of a point to a directed line segment)

Inputs: Alice has a private point $P_0(x_0, y_0)$ and Bob has a private directed line segment with direction from $P_1(x_1, y_1)$ to $P_2(x_2, y_2)$.

Outputs: Which side of P_0 lies in the directed line segment $\overrightarrow{P_1P_2}$.

(1) Alice takes a vector $X = (x_0, y_0, 1)$. Bob takes a vector $Y = (y_1 - y_2, x_2 - x_1, x_1y_2 - x_2y_1)$, and picks randomly and uniformly a number $v \in Z_{n^s}$.

(2) Alice engages in Protocol 1 with Bob, and gets $u = (X \cdot Y + v) \bmod n^s$.

Protocol 3 is secure because of the security of Protocol 1.

In Protocol 1, we take $l = 3$, and obtain the resource needed in Protocol 3 as follows. The communication cost is $[4s+10]\log n + \log \tau$ bits. Alice needs to perform 3 encryptions and 1 decryption, and Bob needs to perform 3 multiplications and 1 encryption.

To determine the relation of P_0 with $\overrightarrow{P_1P_2}$, it suffices for both of them to determine $X \cdot Y > 0$ or $X \cdot Y < 0$. If all operations are performed in real R, Alice and Bob can determine $X \cdot Y > 0$ or < 0 by comparing u and v through a millionaire protocol [25-27]. But now, all computations are done under modulo n^s, and there exists wrap-round modulo n^s, thus they are not able to compare u and v by a millionaire protocol. To obtain the sign of $X \cdot Y$, what they need is to get the most significant bit of $u+v$ by computing $u+v$ bit-by-bit, denoted by $z_i = u_i + v_i$, for $i = 1, \cdots, N_0$. They establish the truth table of the bit-wise addition as in table 1, in which $z_i = u_i + v_i$, and c_i is the carry bit. In the truth table T_1, bits are regarded as logical values, and the bit-wise addition can be transformed to the computation of Boolean function. The protocol is as follows.

Protocol 4 (Secure determination of scalar product by the most significant bit)

Inputs: Alice has a private integer u, denoted by $u_1, \cdots, u_{N_0}$ bit-by-bit, where u_1 is the least significant bit and u_{N_0} is the most significant bit. Bob has another private integer v, denoted by $v_1, \cdots, v_{N_0}$ bit-by-bit, in which v_1 is the least significant bit and v_{N_0} is the most significant bit.

Outputs: Both of them get the encryption of $u+v$ bit-by-bit.

(1) Alice applies MN to T_1 (using h as the encryption key), obtains T_1' and publishes it.

(2) Bob shuffles T_1', obtains T_1'' and publishes it.

(3) Using h as the encryption key, Alice encrypts $\{u_1, \cdots, u_{N_0}\}$, and obtains $\{E(u_1), \cdots, E(u_{N_0})\}$, Bob encrypts $\{v_1, \cdots, v_{N_0}\}$, and obtains $\{E(v_1), \cdots, E(v_{N_0})\}$.

(4) $c_0 = 0$; $E(c_0) = E(0)$;
for $i = 1$ to N_0 do {
for $j = 1$ to 8 do {
if $PET(E(u_i), T''[j,1]) = 1$
and $PET(E(v_i), T''[j,2]) = 1$
and $PET(E(c_{i-1}), T''[j,3]) = 1$,
then return $E(c_i) = T''[j,4]), E(z_i) = T''[j,5]$;
}
}.

When the Protocol 4 is finished, both of them obtain the encryption of $E(z_{N_0})$, which is the encryption of the most significant bit of the sum $u+v$. They can jointly decrypt it and obtain z_{N_0}, which gives the result of $A \cdot B > 0$ or $A \cdot B < 0$.

The security of Protocol 4 is guaranteed by that of Protocol 2.

Similarly to Protocol 2, it is not necessary to consider the communication cost.

The computational cost is $24N_0$ times that of PET.

In computational geometry, if we measure the distance by meters, the value of 16-bit length can measure 65536 meters, so it is enough to take $N_0 \le 16$ in reality.

B. Conditional Oblivious Transfer based on the Relation between A Point and A Directed Line

A conditional oblivious transfer is a variant of oblivious transfer [27-28]. Intuitively, it considers the following problem: two participants, Alice and Bob, have private inputs x and y respectively, and share a public predicate $Q(\cdot,\cdot)$. Alice has two secret messages s_0, s_1, and wishes (obliviously to her) to transfer one of them to Bob depending on Q, but will not learn Bob's private input and the value of Q.

The conditional oblivious transfer is very useful to computational geometry. For example, there are two companies, a railway company and a construction company. The railway company wants to build a railway, while the construction company wants to erect a building. But, the construction of railway will bring inconvenience to the pass in and out of the residents in the building, and the extent of inconvenience depends on which side the building lies on the railway. Therefore, the railway company will compensate the construction company in accordance with which side building lies on. The railway company would not like to tell the construction company where the railway will be built and two compensation schemes, in case the construction company can decide where his building will be erected, while the construction company does not tell the railway company where his building will be erected, in case the railway company can decide his own scheme according to the scheme of construction company to reduce the fee to be paid.

We propose a conditional oblivious transfer scheme based on Protocol 4.

In the scheme, after finishing Protocol 4, both of participants do not decrypt $E(c_{N_0})$

Protocol 5 (Conditional oblivious transfer based on Protocol 4)

Inputs: Alice has a private point $P_0(x_0, y_0)$ and two secret message s_0, s_1, Bob has a private directed line segment with direction from $P_1(x_1, y_1)$ to $P_2(x_2, y_2)$.

Outputs: Bob gets one of s_0 and s_1.

(1) Alice and Bob perform Protocol 3, and obtain u, v, respectively, satisfying $u,v \in Z_{n^s}$ and $u + v = X \cdot Y \bmod n^s$.

(2) Alice and Bob perform Protocol 4, and obtain $E(z_{N_0})$, where E is I. Damgärd and M. Jurik's cryptosystem in Z_{n^s}, u_{N_0}, v_{N_0}, z_{N_0} are the most significant bit of u, v, and $u + v$, respectively, $c_{N_0\text{-}1}$ is the $(N_0 - 1)$th carry bit of $u + v$.

(3) Alice and Bob perform Sub-protocol 6.1. Bob obtains one of s_0 and s_1.

In step (3), a sub-protocol is needed, it is described as follows.

Sub-protocol 5.1

Inputs: The public input is $E(z_{N_0})$, and Alice has two secret message s_0, s_1.

Outputs: Bob gets one of s_0 and s_1.

(1) Alice computes $c=[E(z_{N_0})^{s_1}][E(1)E(z_{N_0})]^{-1}]^{s_0}$. Let $c = (G, H)$, Alice computes $G_1 = G^{\alpha_1}$, sends $c = (G, H)$ and $G_1 = G^{\alpha_1}$ with the proof

$$R_{DH} = \{((g,G,h_1,G_1),\alpha_1) | h_1 = g^{\alpha_1} \bmod n \wedge G_1 = G^{\alpha_1} \bmod n\}$$

to Bob.

(2) Bob computes $G_2 = G^{\alpha_2}$ and $u = L_s(H(G_1 G_2 \bmod n)^{-n^s})$, and obtains $z_{N_0} s_1 + (1 - z_{N_0}) s_0$.

If $z_{N_0} = 1$, that is, P_0 lies in the right of directed line segment $\overrightarrow{P_1P_2}$, then Bob obtains s_1, else Bob obtains s_0.

Security: The security of Bob trivially holds because he does not send anything to Alice.

The security of Alice can be proven by constructing a simulator S_2 for the view of Bob. The view of Bob is $view_2^\Pi = \{c, \mathit{prooftext}\}$, where $\mathit{prooftext}$ is Bob's view in proof of R_{DH}. Because the proof of R_{DH} can be done in zero-knowledge, there exists a simulator S_{DH} to perfectly simulate the view of the verifier (Bob here). Let the output of S_{DH} be R'. $\{\mathit{prooftext}\}$ and R' are identically distributed. The input of S_2 is s, it encrypts s and obtains $c' = E(s)$, takes as output $\{c', R'\}$. Because both of c and c' are encryptions of s, c and c' are identically distributed. So $\{c', R'\}$ is identically distributed with $\{c, \mathit{prooftext}\}$, S_2 simulates perfectly $view_2^\Pi$.

Resource analysis: During an execution of the protocol, only one encryption is exchanged between Alice and Bob, the communication cost is $(s+4)\log n + \log \tau$ bits.

Alice needs to perform 2 encryptions and 3 multiplications, Bob needs to perform 1 decryption.

The security of Protocol 5 is guaranteed by that of Protocol 3, Protocol 4, and Sub-protocol 5.1.

The resource needed in Protocol 5 is the sum of the ones of three protocols employed, that is, the communication cost is $(5s+14)\log n + \log \tau$ bits. Both of participants need to perform jointly $24N_0$ PETs. Besides, Alice needs 5 encryptions, 1 decryption and 3 modular multiplications, while Bob needs 2 decryptions and 3 modular multiplications.

VI. Conclusion

We have given 5 two-party secure protocols: a secure scalar product protocol based on I. Damgärd and M. Jurik's cryptosystem, secure computation of a public Boolean function, secure location of a point to a directed line segment, secure determination of scalar product by the most significant bit, conditional oblivious transfer based on secure determination of scalar product by the most significant bit. The security, communication cost and parties' computational cost in all protocols are analyzed.

Acknowledgement

This work is supported by the National Natural Science Foundation of China under Grants 61272436, 61272404, 61003232, and the Natural Science Foundation of Guang dong Province under Grants 10351806001000000.

References

[1] A. C. Yao, "Protocols for secure computations," Proc. 23rd Annual IEEE Symposium on Foundations of Computer Science, Chicago, pp. 160-164, 1982.

[2] O. Goldreich, *Foundations of Cryptography: Basic Applications*, Cambridge University Press, London, 2004.

[3] B. Goethals, S. Laur, H. Lipmaa, and T. Mielikainen, "On Private Scalar Product Computation for Privacy-Preserving Data Mining," Proc. 7th Annual International Conference in Information Security and Cryptology (ICISC), LNCS 3506, pp. 104- 120, Springer-Verlag, Seoul, Korea, Dec. 2-3, 2004.

[4] Y. Zhu, L. Huang, W. Yang, F. Dong, "Privacy Preserving Aggregate Query of OLAP for Accurate Answers," Journal of Computers, 5(11):1678-1685, 2010.

[5] Du W, Atallah M, "Privacy-preserving cooperative statistical analysis," Proc. 17th Annual Computer Security Applications Conference (ACSAC), ACM SIGSAC, IEEE Computer Society, New Orleans, Louisiana, pp. 102-110, 2001.

[6] C. Chao, Y. Luo, W. Cheng, "Research on the Problem of Privacy-Preserving Closest Pair," Journal of Computers, 5(7): 1120-1124, 2010.
[7] Du W, Zhan Z, "Building decision tree classifier on private data," IEEE ICDM Workshop Proceedings, Volume 14 in the Conferences in Research and Practice in Information Technology Series, Australian Computer Society, Sydney, Australia, pp. 1-8, 2002.
[8] A Artak, E C Vladimir, "A New Effcient Privacy-Preserving Scalar Product Protocol," Proc. 6th Australasian Data Mining Conference (AusDM'7), Gold Coast, Australia. pp. 209-214, 2007.
[9] P. Paillier, "Public-key cryptosystems based on composite degree residue classes," Advances in Cryptology, Eurocrypt'99, LNCS 1592, pp. 223-238, 1999.
[10] Ivan Damgärd, Mads Jurik, "A length-flexible threshold cryptosystem with applications," Proc. 8th Australasian conference on Information security and privacy (ACISP'03). LNCS 2727, pp. 350-364, 2003.
[11] Ivan Damgärd, Mads Jurik, "A Generalisation, a Simplification and Some Applications of Paillier' Probabilistic Public-Key System," Public Key Cryptography 2001, LNCS 1992, pp. 119-136, 2001.
[12] D. Chaum, and T. P. Pedersen, "Wallet Databases with Observers," CRYPTO'92, LNCS 740, pp. 89-105, 1992.
[13] M. Jakobsson and A. Juels, "Mix and Match: Secure Function Evaluation via Ciphertexts," Advances in Cryptology- ASIACRYPT'00, LNCS 1976, pp. 162-177, 2000.
[14] Jens Groth, "A Verifiable Secret Shuffle of Homomorphic Encryptions," Public Key Cryptography 2003, LNCS 2567, pp. 145-160, 2003.
[15] Ivan Damgärd, Matthias Fitzi, Eike Kiltz, Jesper Nielsen, Tomas Toft, "Unconditionally Secure Constant-Rounds Multi-party Computation for Equality, Comparison, Bits and Exponentiation," Theory of Cryptography (2006), LNCS 3876. pp. 285-304, 2006.
[16] J. Bar-Ilan, D. Beaver, "Non-cryptographic fault-tolerant computing in constant number of rounds of interaction," Proc. 8th annual ACM Symposium on Principles of distributed computing, pp. 201-209, 1989.
[17] Craig Gentry, "Fully homomorphic encryption using ideal lattices," Proc. 41st annual ACM symposium on Theory of computing (STOC), pp. 169-178, 2009.
[18] Craig Gentry, "Toward basing fully homomorphic encryption on worst-case hardness," Advances in Cryptology-CRYPTO'10, LNCS 6223, pp. 116-137, 2010.
[19] Craig Gentry and Shai Halevi, "Implementing gentry's fully homomorphic encryption scheme," Advances in Cryptology EUROCRYPT'2011, LNCS 6632, pp. 129-148, 2011.
[20] Nigel P. Smart and Frederik Vercauteren, "Fully homomorphic encryption with relatively small key and ciphertext sizes," Public Key Cryptography-PKC'10, LNCS 6056, pp. 420-443, 2010.
[21] Marten van Dijk, Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan, "Fully homomorphic encryption over the integers," Advances in Cryptology-EUROCRYPT'10, LNCS 6110, pp. 24-43, 2010.
[22] Jean-Sebastien Coron, Avradip Mandal, David Naccache, and Mehdi Tibouchi, "Fully Homomorphic Encryption over the Integers with Shorter Public-Keys," Advances in Cryptology-CRYPTO'11, LNCS 6841, pp. 487-504, 2011.
[23] Zvika Brakerski and Vinod Vaikuntanathan, "Fully Homomorphic Encryption for Ring-LWE and Security for Key Dependent Messages," Advances in Cryptology-CRYPTO'11, LNCS 6841, pp. 505-524, 2011.
[24] Jianer Chen, *Computational Geometry: Methods and Applications*, Texas Texas AδM University, 1996.
[25] Marc Fischlin, "A cost-effective pay-per-multiplication comparison method for millionaires," Proc. the 2001 Conference on Topics in Cryptology: The Cryptographer's Track at RSA, Springer-Verlag, pp. 457-472, 2001.
[26] Li S D, Wang D S, Dai Y Q, Luo P, "Symmetric cryptographic solution to Yao's millionaires'problem and an evaluation of secure multiparty computations," *Information Sciences*, 178(1): 244-255, 2008.
[27] Ian F. Blake, Vladimir Kolesnikov, "Strong Conditional Oblivious Transfer and Computing on Intervals," ASIACRYPT2004, LNCS 3329, pp. 515-529, 2004.
[28] Ian F. Blake, Vladimir Kolesnikov, "Conditional Encrypted Mapping and Comparing Encrypted Numbers," Financial Cryptography and Data Security Conference 2006, LNCS 4107, pp: 206-220, 2006.

Bo Yang received the B. S. degree from Peking University in 1986, and the M. S. and Ph. D. degrees from Xidian University in 1993 and 1999, respectively. From July 1986 to July 2005, he had been at Xidian University, from 2002, he had been a professor of National Key Lab. of ISN in Xidian University, and a supervisor of Ph.D. He has served as a Program Chair for the fourth China Conference on Information and Communications Security (CCICS'2005) in May 2005, vice-chair for ChinaCrypt'2009 in Nov. 2009, and general chair for the Fifth Joint Workshop on Information Security (JWIS 2010), in Aug. 2010. He is currently a professor and supervisor of Ph.D. at School of Computer Science, Shaanxi Normal University, a special-term professor of Shaanxi Province. His research interests include information theory and cryptography.

Chung-Huang Yang received the B.S. degree from National Cheng-Kung University in 1981 and the M. S. and Ph. D. degrees from University of Louisiana at Lafayette in 1986 and 1990, respectively. He is currently a professor at Graduate Institute of Information and Computer Education, National Kaohsiung Normal University, and a member of board of executive directors of (Taiwan) Chinese Cryptology and Information Security Association. His research interests include the implementation of cryptographic algorithms, computer forensics and mobile phone forensics.

Yong Yu received the Ph.D. in cryptography from Xidian University in 2008. He is currently an associate professor at School of Computer Science and Engineering, University of Electronic Science and Technology of China, and his research interests focus on cryptography and its applications, especially public encryption and digital signature.

Dan Xie is a graduate student at School of Computer Science, Shaanxi Normal University, Her research interests include cryptography and cloud computing security.