2009 Eighth IEEE International Symposium on Network Computing and Applications

A Collusion-Resistant Distributed Scalar Product Protocol With Application To Privacy-Preserving Computation of Trust

Carlos Aguilar Melchor *, Boussad Ait-Salem† and Philippe Gaborit‡
Xlim Laboratory UMR CNRS n°6172, University of Limoges
83 rue d'Isle 87000 Limoges, FRANCE
Telephone: +33 (05) 55 43 69 76, Fax: +33 (05) 55 43 69 77
* Email: carlos.aguilar@unilim.fr † Email: boussad.ait-salem@unilim.fr ‡ Email: philippe.gaborit@unilim.fr

***Abstract*—Private scalar product protocols have proved to be interesting in various applications such as data mining, data integration, trust computing, etc. In 2007, Yao et al. proposed a distributed scalar product protocol with application to privacy-preserving computation of trust [1]. This protocol is split in two phases: an homorphic encryption computation; and a private multi-party summation protocol. The summation protocol has two drawbacks: first, it generates a non-negligible communication overhead; and second, it introduces a security flaw.**

The contribution of this present paper is two-fold. We first prove that the protocol of [1] is not secure in the semi-honest model by showing that it is not resistant to collusion attacks and we give an example of a collusion attack, with only four participants. Second, we propose to use a superposed sending round as an alternative to the multi-party summation protocol, which results in better security properties and in a reduction of the communication costs. In particular, regarding security, we show that the previous scheme was vulnerable to collusions of three users whereas in our proposal we can fix $t \in [1..n-1]$ and define a protocol resisting to collusions of up to t users.

***Index Terms*—Privacy-preserving computation of trust; Secure multi-party computation; Secure scalar product; Superposed sending.**

I. Introduction

Secure multi-party computation (SMPC) [2], [3], [4] allows multiple parties to perform a cooperative computation, based on their private inputs, without revealing to each other anything about these inputs except the computation result. Since 1982, when A. C. Yao [2] firstly introduced SMPC, it has been a hot research topic. In recent years, it has become essential in several applications [5], including in privacy-preserving computation of trust.

Among all the basic functions that may be securely computed in a multi-party setting, scalar products are specially interesting, as many applications can essentially be reduced to the evaluation of such a product. As an example, scalar products can be used for the computation of weighted trust values. In this application, Alice, denoted A, wants to obtain the trust value of an unknown entity denoted E based on what other entities denoted $(B_1, B_2, ..., B_n)$ think about E together with A's confidence in these entities. Actually, in real life, this information is usually considered sensitive, e.g., B_i may not want to disclose that he does not trust E at all, and A hopes to conceal the fact that her confidence in B_j is low.

The problem was solved by Yao et al. in [1] by computing the scalar product of two vectors - one vector representing A's confidence values for a set of entities, the other vector representing recommendations of these entities on E. In their paper Yao et al. extend the previously existing private two-party scalar product protocols [6], [4] to a multi-party setting in which one of the vectors of the scalar product is distributed among multiple entities.

The goal of this present paper is two-fold. First we prove that Yao et al. proposition doesn't fully respect the semi-honest model, since it is not resistant to collusion attacks. Actually, we prove that Yao et al. scheme is vulnerable to collusion between four players that can be practically realized. Second, we present a new scheme which resists to collusions. The scheme that we propose doesn't depend on its practical application. It can be adapted for any application that requires a secure distributed scalar product for trust establishment, especially, for recommendation and reputation systems.

A. Contribution

In the protocol of Yao et al. there is a first phase based on homomorphic encryption, which results in the computation of the scalar product plus a random mask. In a second phase, they use a Private Multi-Party Summation (PMPS) protocol [8] to compute the random mask which is subtracted from the result of the first phase to obtain the unmasked scalar product. The main issue with this protocol is a security flaw in the Private Multi-Party Summation protocol (PMPS) which allows collusions of Alice and three honest-but-curious users to learn other participants' inputs. We present two scenarios of attack which can be done during the summation step.

In this paper, we propose a new scheme inspired by the Dining Cryptographers protocol (also known as DC-net protocol) proposed in 1988 [9]. The DC-net protocol (also known as superposed sending) allows to send a message collaboratively from a set of users without revealing which user is the sender even to a collusion of users of the collaborative set. Based on its resistance to collusions, we are able to obtain a scheme that allows to compute a distributed scalar product with resistance to collusions of up to $n-2$ participants with Alice. Besides, our solution is easily scalable and more efficient in terms of communication cost, since we reduce by 50% the communication overhead.

The basic idea in the DC-net protocol is that each message (which is composed of zeroes if the user has nothing to send) is masked by adding a random number (the mask) chosen in such a way that when all the masked messages are added the masks cancel each other leaving only the sum of the messages sent.

In our private scalar product protocol, we do the same homomorphic encryption computation than in the protocol of Yao et al. : each user computes a part of the scalar product using a ciphertext of the homomorphic encryption scheme and hides the result by adding a mask. The main difference is that in the protocol of Yao et al. this mask is chosen uniformly at random and in our protocol it is computed as on a DC-net protocol. This ensures that the masks cancel each other and that at the end of the first phase Alice obtains an unmasked private scalar product and the PMPS protocol is unnecessary.

In a first protocol we use the fully-connected DC-net protocol that is proved unconditionally secure but less practical. Then, we propose a computationally secure variant, taking into account some performance properties. This solution is based on DC-net a scheme combined with Diffie-Hellman (DH) Secret Key Exchange (SKE) protocol and a Pseudo-Random Number Generator (PRNG).

B. Organization

This paper is organized as follows. In section II we recall basic notions: cryptographic definitions and some specific building blocks. In section III, we introduce our protocol. Section IV is devoted to the performance evaluation. Finally, we conclude in Section V.

II. Basic Notions

A. Cryptographic definitions

We provide here some basic definitions of cryptographic techniques used in our protocol and their security properties.

1) Homomorphic Encryption schemes: An homomorphic encryption scheme (see for example [10], [11]) has three functions (Gen, Enc, Dec), where Gen generates a private key s_k and a public key p_k, Enc and Dec are encryption and decryption functions, respectively. The encryption function Enc is said to be homomorphic, if the following holds: $Enc_{p_k}(x;r).Enc_{p_k}(y;r^{'}) = Enc_{p_k}(x+y;r.r^{'})$, where x and y denote plaintext messages and r and $r^{'}$ denote random strings. Another property of such scheme is that $Enc_{pk}(x;r)^y = Enc_{pk}(x.y;r^y)$. This means that a party can add encrypted plaintexts by doing simple computations with ciphertexts, without having the private key. The arithmetic performed under the encryption is modular. An Homomorphic scheme is called semantically secure when a probabilistic polynomial-time adversary cannot distinguish between random encryption of two elements chosen by herself.

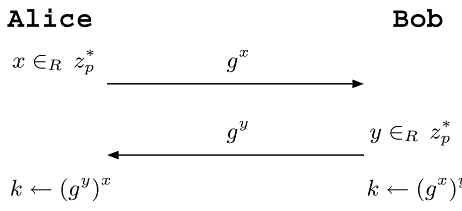

Fig. 1. The Diffie-Hellman Key Exchange

2) Diffie-Hellman Key Exchange: The DH protocol is one of the most frequently used key exchange protocol for sharing keys in cryptographic schemes. It was proposed by Whitfield Diffie and Martin Hellman in 1976 [12].

In a DH-SKE between two players $Alice$ and Bob, $Alice$ and Bob fix a finite cyclic group G and a generator g. They respectively pick random $a, b \in [1, |G|]$ and exchange g^a, g^b. The common secret key they obtain is g^{ab}. An example of DH-SKE in z_p^* cyclic group is depicted in figure 1.

In order to totally break the protocol, a passive eavesdropper, Eve, must compute the Diffie-Hellman function defined as: $DH_g(g^a, g^b) = g^{ab}$. We say that the group G relies on the *Computational Diffie-Hellman assumption* (CDH) if no efficient algorithm can compute the function $DH_g(x, y)$ in G. Some related works provide limited reductions from computing discrete log to compute the Diffie-Hellman function [13], [14]. Nevertheless, this assumption remains insufficient to prove the security of many cryptographic protocols based on Diffie-Hellman protocol. There is another stronger assumption called the *Decision Diffie Hellman (DDH)* [15]. Loosely speaking, the DDH assumption states that no efficient algorithm can distinguish between the two distributions (g^a, g^b, g^{ab}) and (g^a, g^b, g^c) where a, b, c are chosen at random in $[1, |G|]$.

3) Pseudo-Random Number Generators: PRNG schemes were introduced by Goldreich, Goldwasser and Micali [16]. A Pseudo-Random Bit Generator (PRNG) is defined in [17] as a deterministic [1] algorithm which, given a truly random binary sequence of length k, outputs a binary sequence of length $l >> k$ which "appears" being random. The input to the PRNG is called the seed, while the output of the PRNG is called a pseudorandom bit sequence.

The minimum security requirements for a PRNG is that the length l of the random seed should be sufficiently large so that a search over 2^l elements (the total number of possible seeds) is infeasible for the adversary. The truly randomness of the seed is also required. Recommendations about the seed requirements can be found in [18].

We can categorize the pseudo-random functions following their security requirements. The first category includes the

[1]Deterministic means that given the same initial seed, the generator will always produce the same output sequence

pseudo-random generators which pass all *polynomial-time statistical tests*. We say that a PRNG pass these tests if no polynomial-time algorithm can correctly distinguish between an output sequence of the generator and a truly random sequence of the same length with a probability significantly greater than $1/2$.

The second category called cryptographically secure PRNGs implies stronger security assumptions. The security of each generator relies on the presumed intractability of an underlying number-theoretic problem. Numerous works have been proposed, each one based on a specific assumption. For instance, it is shown in [19][20] that efficient pseudorandom generators could be built if one assumes that computing discrete logarithms with short exponents is a hard problem. Steinfeld et al. [21] propose an improved version of the well-known RSA generator assuming the intractability of a strong variant of the RSA problem. There are also propositions related to the intractability of the DDH problem, Naor and Reingold [22] describe a beautiful application of DDH. They show how to construct a collection of efficient pseudo random functions. These methods are stronger than the first one, however, they are relatively slower compared to the first category.

B. Specific privacy-preserving protocols

In this section we introduce the building blocks used in Yao et al. protocol as well as in our proposal.

1) Private Multi-Party Summation Protocol: A PMPS protocol is a protocol that allows to compute the sum of private values of n parties, each one with a number V_i, cooperate to simultaneously compute the sum $S = \sum_{i=1}^{n} V_i$ without revealing to each other anything other than the final sum S. In Yao et al. any PMPS protocol may be used, but for communication performance reasons they propose to use the scheme of Atallah et al. [8] which is presented below.

- Every party i gets a random number R_i
- Every party $2i$ sends to $2i+1$ his $V_{2i} + R_{2i}$
- Evey $2i+1$ sends to $2i$ his R_{2i+1}
- The odd-numbered parties compute $V + R = \sum_{i=1}^{n} V_i + R_i$ and the even-numbered parties compute $R = \sum_{i=1}^{n} R_i$ using for instance a "tree based" approach
- At the end, the odd (resp. even) party simultaneously exchange their quantities to obtain V (The authors do not explicitly define the simultaneous secret exchange used here).

2) Superposed Sending (DC-net): Chaum introduced the dining cryptographers problem in 1988: three cryptographers want to find out whether if one of them has paid for the dinner, without learning whom [9], in other words they want to be able to say "I have paid" anonymously. Computer networks that implement the resulting protocol are called dining cryptographers networks (DC-nets); the protocol itself is named the DC-net protocol (or superposed sending protocol) and allows to send messages in a per-round basis. Each round is similarly called a DC-net round (or a superposed sending round).

In a superposed sending protocol, all the participating users send scrambled messages at each round, even if they don't have anything to transmit. The protocol described below shows how a DC-net round is performed.

Superposed Sending

Let $U_1, ..., U_n$ be a set of users such that each couple of users (U_i, U_j) shares a unique common l-bits long secret, $s_{i,j} = -s_{j,i}$. Each user U_i encodes a message $Message(i)$ as a string of l zeroed bits if he has nothing to transmit, or as an l-bit message if he has something to transmit.

Round progress:

1. Every user U_i broadcasts: $Broadcast(i) = Scrambling(i) \sum Message(i)$ with $Scrambling(i) = \sum_{j=1,\, j\neq i}^{n} s_{i,j}$.
2. Every user computes the result of the round: $Result = \sum_{i=1}^{n} Broadcast(i)$.

Since every $s_{i,j}$ appears in two forms (once inserted by U_i $(s_{i,j})$ and once inserted by U_j $(-s_{i,j})$), the scramblings cancel each other and each user obtains $Result = \sum_{i=1}^{n} Message(i)$.

In [9] Chaum proves that the basic DC-net protocol achieves sender anonymity and recipient anonymity even against *computationally unrestricted attackers*:it is proved to be *unconditionnaly secure*. However, the proof for recipient anonymity implicitly assumes that the partial sums are broadcast reliably, i.e., each message of an honest participant is broadcast to all participants without being modified.

In order to maintain the unconditional sender anonymity, the $S_{i,j}$ must be used just once and renewed each round, which is a similar situation as for the *one-time pad*. They can be randomly generated and stored in large storage devices (DVDs, for instance), that are exchanged physically by the users, or be pseudo-randomly generated from a secret seed known only to users U_i and U_j.

The DC-net approach introduces a vulnerability if some users are malicious. In fact, the service of a DC-net can be easily disrupted by one or more cheating participants who send wrong messages. Such an attack results in a denial of service, but does not reveal any information about which masks are used, or which user has sent a given message. Disruptions of the DC-net has been considered in [9], [24], [25].

In order to simplify the security proofs given along this paper, we represent DC-net as a connected graph $G(V, E)$ named *key graph* where, the set of vertices V includes the all participants and the set of edges E represents the shared keys $s_{i,j}$.

Definition 1: An *anonymity set* seen by a set of keys is the set of vertices in a connected component of G formed from the original graph by removing the edges concerned.

Theorem 1: Let O be an observer and A the non-singleton anonymity set seen by O's known keys, O can learn nothing about the members of A except the overall party of their sums.

Proof: For the theorem proofs see [9] ∎

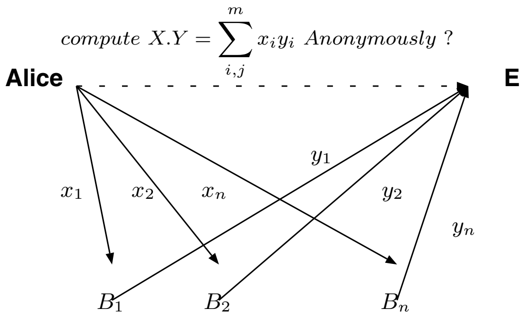

Fig. 2. The Problem Definition

III. A collusion-resistant distributed scalar product protocol

A. Goal

Our aim through this paper is to give a solution to the distributed scalar product computation problem applied to the privacy-preserving computation of trust, defined as follows.

Definition 2: Let $P = \{p_1, p_2, ..., p_N\}$ be the set of all participants in the network. Alice, an element of P denoted by A, wants to compute the trust weight of E (another element of P) by taking into account the recommendations of a set of elements B_i (for $1 \leq i \leq n$) of P. Alice's input is the private vector $X = (x_1, ..., x_n) \in z_m^n$. Each party B_i (for $1 \leq i \leq n$) has a private input value y_i. At the end of the protocol, only the scalar product $X.Y\ mod\ m$ is learned by Alice or by every participant, where $Y = (y_1, ..., y_n) \in z_m^n$ and m are public parameters (see fig. 2).

B. Adversary model

We place ourselves in the *semi-honest* model [3]. A semi-honest user follows the protocol properly but can keep a record of all its intermediate computations and analyze them in order to obtain more information.

The formal definition of the semi-honest model in the case of two-party computation has been presented by Goldreich in [3].

Suppose that f is a polynomial-time function, and Π is a secure two-party protocol for computing f. Let x and y be the parties' respective private inputs to the protocol. For each party, define its view of the protocol as $(x, r_1, m_1, ..., m_t)$ (respectively, $(y, r_2, m_1, ..., m_t)$), where r_1 and r_2 are the parties' internal coin tosses, and m_i is the i-th message received during the execution of the protocol. The view of the first (resp., second) party is denoted by $view_1^{\Pi}(x, y)$ (resp., $view_2^{\Pi}(x, y)$). The output of the protocol is denoted by $output^{\Pi}(x, y) = (output_1^{\Pi}(x, y), output_2^{\Pi}(x, y))$.

Definition 3: (privacy with respect to semi-honest behavior): Regarding f, if there is a probabilistic polynomial-time algorithm, denoted by S, such that:

$$\{(S(x, f_1(x, y)), f(x, y))\}_{x,y \in \{0,1\}^*}$$
$$\equiv^c \{(view_1^{\Pi}(x, y), output^{\Pi}(x, y))\}_{x,y \in \{0,1\}^*}$$
$$\{(S(x, f_2(x, y)), f(x, y))\}_{x,y \in \{0,1\}^*}$$
$$\equiv^c \{(view_2^{\Pi}(x, y), output^{\Pi}(x, y))\}_{x,y \in \{0,1\}^*}$$

Where $\equiv^c$ denotes computational *indistinguishability* by any polynomial-time algorithm, we say that π securely computes f.

Goldreich also defines in [3] the semi-honest model in the case of a multi-party computation. The proposed definition is similar except that collusions are also considered (ie. the particpants can collude in order to gain more information). In such a setting a subset of users can combine their observations, to discover information on the other users' inputs.

C. Notations

In addition to the notations used in Yao et al. paper, we use the following notations:

$prior(p_i)$	The priority level of a participant p_i, we can take here: $prior(p_i) > prior(p_j)\ if\ i > j$
$s_{i,j}$	The secret key of $m\ bits$ shared between two participants p_i and p_j. $s_{i,j} = -s_{i,j}$ if $prior(p_i) > prior(p_j)$
$\|s_{i,j}\|$	The number of bit making up $s_{i,j}$
$Rand_{seed}$	PRNG based on $seed$
pk	A public key
sk	A private key
Enc_{pk}	An homomrphic encryption function using the public key pk
Dec_{sk}	An homomrphic decryption function using the private key sk

D. Yao et al. proposal

In order to solve the problem given in section III-A, Yao et al. proposed the protocol presented below:

- With an homomorphic encryption scheme, A encrypts the ponderations x_i and sends them to the B_i
- Each B_i computes an exponentiation of the ponderation, masks the result, and sends the masked result $(Enc_{pk}(x_i y_i - v_i; r_i^{y_i} r_i^{'}))$ to A
- The B_i follow the PMPS protocol to compute $\sum_{i=1}^{n} V_i$ and send the result to A
- A sums the masked results and subtracts the result of the PMPS protocol

Our idea is that having two steps:

- B_i mask the values
- B_i compute securely the sum of the masks

is non-optimal from a communication and computation point of view. Moreover, the PMPS protocol proposed to secure the computation of the sum introduces a vulnerability against collusions as it will be shown below. We therefore propose to replace these steps by a superposed-sending scheme which ensures a simple way to mask values in such a way that the masks cancel each other on the final step with strong security properties and low communication complexity.

To assert our propositions, let's see more closely the security level of Yao et al. proposition.

Proposition 1: During the summation step, a collusion of three users with Alice allows to discover V_i.

Proof:
As we are under a semi-honest model, the participants can record the intermediates values (V_i, R_i) as well as the partial results involved by the computation of $V + R$ and R (see section II-B). They have, also, the capacity to collude by exchanging their intermediates results.
In order to prove this proposition, we give two instances of attacks. The first one against an even-numbered set of users and the second one against an odd-numbered set. We present one instance of each attack.

1. **If Alice wants to know the private value** V_{2i+1}, she creates a collusion with $(2i+3,\ 2i,\ 2i-1)$:
   - $2i+3$ supplies: $A = \sum_{i=1}^{2i+1}(V_i + R_i)$
   - $B = \sum_{i=1}^{2i-1}(V_i + R_i)$ is provided by $2i-1$
   - $C = (V_{2i} + R_{2i})$ and R_{2i+1} are provided by $2i$
   - Finally we obtain: $V_{2i+1} = A - (B + C + R_{2i+1})$
2. **If Alice wants to know the private value** V_{2i}, she creates a collusion with $(2i+1,\ 2i+2,\ 2i-2)$:
   At $2i+1$ we have $V_{2i} + R_{2i}$: it colludes with $2i-2$ and $2i+2$ in order to obtain R_{2i} and therefore it obtains V_{2i}.
   - $2i+2$ supplies: $A = R_1 + ... + R_{2i-2} + R_{2i-1} + R_{2i} + R_{2i+1}$
   - $B = R_1 + ... + R_{2i-2} + R_{2i-1}$ is provided by $2i-2$
   - $C = R_{2i+1}$ is provided by $2i+1$
   - We obtain: $R_{2i} = A - (B + C)$ and thus, we find V_{2i}

■

Proposition 2: The private value y_i of a participant B_i is revealed with just a collusion between *Alice* and three compromised participants.

Proof: V_i is obtained with a collusion of three elements (proposition 1). As we are under a semi-honest model, *Alice* has the capacity to store the intermediate results namely $x_i y_i - V_i$, this fact leads to the knowledge of y_i. ■

E. Our proposal

In this section, we first give an overview of a preliminary version of our protocol using a fully-connected DC-net that has the advantage of ensuring unconditional security against collusions of users, but hard to implement from a practical point of view. Then, we propose a computationally secure variant, taking into account performance issues. This second scheme is based on the Diffie-Hellman key exchange protocol and on a pseudo-random number generator.

1) Description of the protocol: The initial version of our protocol is presented as follows:

Fully-connected Protocol

```
For $i$ from $1$ to $n$
  1. Alice generates a private and public key pair $(s_k, p_k)$ and
     sends $p_k$ to $B_i$ (this is done just once or periodically).
  2. Alice encrypts the element $x_i$ of her vector $X$ with her
     public key in homomorphic encryption. The ciphertext $c_i =
     Enc_{p_k}(x_i; r_i)$ is sent to $B_i$ where $r_i$ is a random string.
  3. As we use an homomorphic encryption, $B_i$ does not know
     Alice's private key. However he is able to compute the ciphertext
     corresponding to $x_i y_i$, even though he does not know $x_i$. $B_i$
     computes the ciphertext $w_i$ as follow:
       • $w_i = c_i^{y_i} \, mod \, m$, we obtain: $w_i = Enc_{p_k}(x_i.y_i; r_i^{y_i})$.
  4. For $j$ from $1$ to $n$
       • If $s_{i,j}$ doesn't exist yet, $B_i$ and $B_j$ randomly generate it
         (we assume the existence of an unconditionally secrecy
         channel between all players in order to securely exchange
         the shared keys).
  5. In order to hide $y_i$ to Alice, $B_i$ computes like a DC-net
     protocol:
       • The mask $MSK_i = \sum_{j=1, j \neq i}^{n} s_{i,j}$
       • The encryption $w_i^{'} = w_i \ . \ Enc_{p_k}(MSK_i; r_i^{'})$.
  6. Then $B_i$ sends $w_i^{'}$ to Alice.
When Alice receives all $w_i^{'}$, she computes:
$\prod_{i=1}^{n} w_i^{'} \, mod \, m =$
$\prod_{i=1}^{n} Enc_{p_k}(x_i.y_i; r_i^{y_i}).Enc_{p_k}(MSK_i; r_i^{'}) =$
$\prod_{i=1}^{n} Enc_{p_k}(x_i.y_i + MSK_i; r_i^{y_i}.r_i^{'}) =$
$Enc_{p_k}(\sum_{i=1}^{n} x_i y_i + \sum_{i=1}^{n} MSK_i; \prod_{i=1}^{n} r_i^{y_i}.r_i^{'}) =$
$Enc_{p_k}(X.Y + \sum_{i=1}^{n} MSK_i; \prod_{i=1}^{n} r_i^{y_i}.r_i^{'})$ $(*)$

Alice uses her private key $s_k$ and $Dec$ function to decrypt $(*)$. Since
every $s_{i,j}$ appears in two forms (once inserted by $B_i$ $(s_{i,j})$ and once
inserted by $B_j$ $(-s_{i,j})$ ), the sum $\sum_{i=1}^{n} MSK_i$ will be equal to
zero, thus Alice obtains immediately the final result $X.Y$
```

In order to analyze the security of our technique, we now examine the protocol more closely. Our protocol is based on two primitives namely homomorphic encryption and DC-net protocol. The first one is used to ensure the privacy of Alice's input (x_i) and the second one to hide the B_i's input (y_i). Hence, it is obvious to say that the security of our private scalar product depends on the security of both homomorphic encryption and DC-net protocol.

In what follows, we try to demonstrate the privacy level of the protocol under a semi-honest model. Before, we consider these two lemmas.

Lemma 1: Consider the modular summation in z_m and assume that the secret $s_{i,j}$ randomly generated and renewed each round. The sum $x_i.y_i + \sum_{j=1, j \neq i}^{n} s_{i,j} \, mod \, m$ provides perfect secrecy.

Proof: As $s_{i,j}$ randomly generated and renewed each round, the $\sum_{j \neq i}^{n} s_{i,j}$ is also. The length of $x_i.y_i$ equals to $\sum_{j \neq i}^{n} s_{i,j} = m$. We have here the whole conditions of a one-time pad encryption. On the other hand, Shannon has proved the unconditional security of one-time pad encryption. Hence, the modular summation $x_i.y_i + \sum_{j \neq i}^{n} s_{i,j} \, mod \, m$ provides perfect secrecy. ■

Lemma 2: The privacy of y_i value associated to a participant B_i is resistant to a collusion between $Alice$ and $n-2$ compromised players B_j.

Proof: Alice has $x_i y_i + MSK_i$. Let $G(V, E)$ be the *key graph* representing our first version. Let be a collusion between A and all players in the set $\{B_j \neq B_i \, / \, 1 \leq j \leq n-2\}$ (Alice colludes with B_j by pooling their secret keys). Thus B_i and the remaining not compromised player build an anonymity set seen by the pooled keys (the secret keys which link the A's colluders and the anonymity set). Following theorem 1, Alice gain nothing about $x_i y_i$ and thus about y_i, since the A's colluders haven't any information about the shared keys between the anonymity set members. ■

Theorem 2: In the semi-honest model, our distributed private scalar product is secure.

Proof: Each entity B_i only sees a random ciphertext from Alice. B_i cannot guess the plaintext, since the homomorphic encryption scheme is assumed to be semantically secure. Hence, B_i cannot guess Alice's value x_i.

A collusion against Alice which means to try to guess the x_i value with a collusion is also protected by the semantic security property.

Alice records all intermediate results $x_i.y_i + \sum_{j\neq i}^{n} s_{i,j}$, according lemma 1, it is impossible for Alice to guess any y_i value.

The full collusion against B_i involves all other $(n-1)$ participants. It is right that our proposition can't protect against this attack. However, in practice, it is impossible to have all participants who are mutually mistrustful colluding against just one in an anonymous system. Hence, a more realistic attack is the partial collusion, which involves only some of the participants.

Under a partial collusion, our solution protects up to a collusion of $n-2$ B_j with Alice. Lemma 2 shows clearly this property. ■

2) Practical consideration: The unconditional security of the previous superposed sending scheme relies on the unconditionally secrecy channels. However, this is impractical, especially, with the increasing network size, as does the total number of the shared keys.

The problem can be solved via the public key cryptography. The mechanism chosen here to distribute the secrets is the Diffie-Hellman key exchange protocol (see section II-A2). The only thing that will be changed compared to the previous version is the key distribution mode. Each B_i initiates, at each round, a new DH-SkE with every participant B_j in order to establish the new secret key s_{B_i,B_j}. This can be done by using a trust third party such as a central PKI, yet this manner goes against the distributed aspect of a protocol. As the adversary is a passive attacker in the sense that the *man in the middle attack* is not feasible, we can avoid the use of PKI by using the *zero knowledge proof protocol* [27] so as to ensure, at least, whether each party knows its secret during the DH exchange.

This second approach requires $n(n-1)/2$ keys, approximately $O(n^2)$ keys. This leads the scalability problem in practice when applied to a large system. About the computation complexity, Each participant B_i must perform $O(n)$ exponentiations in order to compute the shared keys with the remaining $n-1$ participants.

Another improvement is to launch the above setting secrets procedure only once, at the protocol initialization. Every pair of participants (B_i, B_j) keeps the initial shared secret key as a seed ($seed_{B_i,B_j}$) to the future secret key generations. Thus, at each round, every pair (B_i, B_j) pseudo randomly generates its shared secret key s_{B_i,B_j} based on its seed.

Indeed, by choosing an adequate group, the shared key exchanged during the DH protocol has all the requirements to be a seed (see section II-A3).

By doing so, we drop the communication overhead and the expensive exponentiation computations generated by the DH exchange during each round. Nevertheless, the overhead generated by the PRNG will always remain.

We can decrease both the communication and the computation overhead by using a less dense *key graph*. To build this graph, for instance in a distributed manner, every participant B_i chooses randomly k distinct participants B_j. Therefore, each key graph's node will have an average density d approximately equals to $2k$ (see proposition 3 bellow).

Proposition 3: The probability for a player B_i to be chosen by δk other players for $\delta > 1$ is in $O(e^{-\delta k})$

Proof: A player chooses k distinct nodes among $n-1$ (he doesn't choose himself). As players are chosen randomly, a given player B_i will have a probability $k/(n-1)$ of being chosen by another player B_j. Let Consider now the random variable X_i which equals to 1 whether a player being chosen and 0 else. Which means that X_i is a Poisson trials with a distribution $Prob[X_i = 1] = k/n-1$. As each node operates independently, sequence $X_1, ..., X_{n-1}$ is an independent Poisson trials with $E[\sum_{i\in[n-1]} X_i] = k$. Using the Chernoff bound, is proved for $\delta > 1$ that: $prob[\sum_{i\in[n-1]} X_i > \delta k] < (e^{\delta-1}/\delta^{\delta})^k$. Hence, we can infer that the probability to be chosen by more than k players is in $O(e^{-\delta k})$. ■

A summary of one variant of a protocol is depicted bellow:

The Practical version (version 2)

```
Let $P = \{p_1, p_2, ..., p_N\}$. $p_i$ can play the role of $A$, $E$ or $B_i$.
$G = Z_p^*$ denote a finite multiplicative cyclic group of prime order $p$.
Let $g$ be the generator and $|G| > 2^{2m}$.
$k$ is a parameter taken relatively to the security level that we aim to reach.
At the protocol initialization (round r=1): For each (A, E and $B = \{B_1, B_2, ..., B_n\}$) do
Each pair $(B_i, B_j)$ agrees on $(G, g)$.
For all $Bi \in B$ do
  • $B_i$ selects a random secret: $e_i \in Z_p$.
  • $B_i$ chooses randomly among the $n-1$ participants, a set of $k$ distinct friends $\{f_1, f_2, ..., f_k\}$.
  • For $j$ from 1 to $k$ do
      – $B_i$ sends to $f_j$ the public party $g^{e_i}$ and the knowledge proof of $e_i$.
      – $f_j$ receives $g^{e_i}$, then:
          ∗ Computes $(g^{e_i})^{e_{f_j}}$.
          ∗ Replies to $B_i$ by sending him the public party $g^{e_{f_j}}$ and the knowledge proof of $e_{f_j}$.
      – $B_i$ computes $(g^{e_{f_j}})^{e_i}$.
      – Finally, $B_i$ and $f_j$ hash $g^{e_i e_{f_j}}$ to an $m$-bit string: $seed_{i,f_j} = h(g^{e_i e_{f_j}})$ which will be the secret seed between $B_i$ and $f_j$.
Now each $B_i \in B$ shares a secret seed with approximately $d_i = 2k$ participants (proposition 3)

At each round $r \geq 1$: (Distributed scalar product phase see section III-A)
Consider $n \geq 2$ and $s^{r=0}_{B_i,B_j} = seed_{B_i,B_j}$.
For $i$ from 1 to $n$ do
Consider the set $BF_{B_i} = \{B_j / s^r_{B_i B_j} exists\}$.
  • Perform the steps 1, 2, 3 of version 1.
  • For $j$ from 1 to $|BF_{B_i}|$ do
```

  - B_i and B_j pseudo randomly generate a new shared secret: $s^r_{B_i,B_j} = Rand_{seed_{B_i,B_j}}(s^{r-1}_{B_i,B_j})$.
- B_i Computes the mask: $MSK_{B_i} = \sum_{B_j \in BF_{B_i}} s^r_{B_iB_j}$
- Resume the private distributed scalar product at step 5 of the first version of a protocol.

In this paragraph, we are interested by the security proofs. Before proving the security of our protocol by taking into account these practical improvements, we consider this theorem.

Theorem 3: Consider these two sets: $BF_{B_i} = \{B_j/s_{B_iB_j} exists\}$, $S_{B_i} = \{s_{B_iB_j}/B_j \in BF_{B_i}\}$. The y_i value associated to a participant B_i is discovered if and only if the whole elements of BF_{B_i} collude with A.

Proof: Let $G(V,E)$ be the *key graph* where $V = \{B_1, B_2, ..., B_n\}$ and $E = \bigcup_{i=1}^n S_{B_i}$.

Consider a collusion between A and some $B_j \notin BF_{B_i}$. The exchanged keys inside this set sees the anonymity set $BF_{B_i} \bigcup \{B_i\}$ (def. 1). Following theorem 1, A gains nothing about $x_i.y_i$ and thus about y_i.

Now, we consider a collusion between A and $B_j \in BF_{B_i}$ for j from 1 to $|BF_{B_i}| - 1$ by pooling their secrets keys $s_{B_iB_j}$. B_i and the remaining not compromised player build an anonymity set seen by the pooled keys. Following theorem 1, Alice gains nothing about y_i, since the colluders haven't any information about the shared keys between the anonymity set members. ■

Regarding the security of the homomorphic encryption, it remains the same ie. semantically secure. Hence, the privacy of A's input namely x_i is preserved even against a full collusion.

Alice records all the intermediate results $x_i.y_i + \sum_{B_j \in BF_{B_i}} s_{B_i,B_j}$, following lemma 1 except that now it is no longer a question here of unconditional security, yet the security of this sum depends on the security of the DH-SKE and the PRNG function. Consequently, the privacy of y_i values are protected by the number-theoretic assumptions behind these two primitives.

About the resilient of B_i's value namely y_i to collusions, let us consider theorem 3. This theorem proves that the second version of our scheme presented above is resistant to a collusion between A and $|BF_{B_i}| - 1$ elements.

On the one side, we note that by increasing the length of BF_{B_i} we increase at the same time the resilient of a protocol to collusions. On the other side, the length of BF_{B_i} depends on the k value. Hence, we can say that by increasing k we increase the resistance of a protocol to collusions and thus rising the security level of a protocol.

In conclusion, under a semi-honest model, the security of our protocol proportionally increases by increasing the density d_i per node of a key graph. For instance, if the density d_i of a participant B_i equals to $n-1$, we reach a protection against a collusion of $(n-2)$ B_i with Alice.

IV. Performance analysis and comparison

Despite that the detailed performance analysis can be made only in a real trust model, we will try to give, here, some performance comparisons between our practical model and Yao et al. proposal. It concerns the communication and computation complexity of each model.

Protocol	Operation	Overhead
Yao et Al. Protocol	Comp. (Alice) Comp. (B_i)	n homomorphic op. $log\, y_i$ homomorphic op. + RG op. (PMPSP)
Our Protocol	Comp. (Alice) Comp. (B_i)	n homomorphic op. $log\, y_i$ homomorphic op. + $O(k)$ PRNG

TABLE I
Computation overhead comparison

Computation(comp.) complexity of the yao et al. proposition and our protocol. We denote by n the length of alice's vector X and by RG the random generation operation. The logarithmic factor is due to using multiplications to compute exponentiation.

We do not consider here the overhead generated by the initialization phase, since it achieved only once. Regarding the computation complexity, this step generates one random generation and performs $O(k)$ exponentiation per participant. Regarding the communication overhead, this step generates approximately $O(kn)$ communications.

In terms of communication cost, in the case of yao et al. model, we have a total of $4n$ communications:

- $Alice$ and all B_i perform $2n$ emissions per round when sending their encrypted private values;
- During the summation step the B_i (for i from 1 to n) perform at least $2n$ communications per round. This is due essentially to the sending of V_i, R_i values and to the tree-based summation step.

Whereas, in our case, we just perform $2n$ communications per round which represent a reduction of 50% of communications compared to the Yao et al. model. We can be much more precise. If we consider the simultaneous exchange of secrets using the bit by bit exchange, which is recommended in the last step of PMPS protocol for security considerations, we note an added overhead estimated up to m communications.

In terms of computation cost, the computation overhead is approximately equivalent. In both protocols see table I: A uses n homomorphic encryption operations per round and each B_i performs one exponentiation per round. In our case, the only additional overheads are those caused by the PRNG computations performed by each B_i. Indeed, every B_i (for i from 1 to n) pseudo randomly generates d_i secrets per round. consequently, each scalar product phase performs approximately nd_i PRNG computations. Whereas in the Yao et al. model there is also the computations overhead generated by the PMPS protocol which is estimated to n random generation operations per round.

V. Conclusion

Secure the computation of scalar product is an important issue in practice. In this paper, we have discussed one instance of scalar product applied to the privacy preserving computation trust.

We have considerably improved Yao et al. proposal from the security point of view. Indeed, after proving that Yao et al.

solution doesn't protect against collusions, we have presented a new easily scalable secure model. We have also proved that this scheme is optimal in terms of resistance to collusions.

We have proposed an approach based on DC-net scheme which presents some practical difficulties then, based on this idea, we have presented another efficient scheme by considering some practical considerations such as: the communication and computation complexity. This scheme uses the DH-SKE combined with the PRNG. We have proved that this model is much more secure and generates less communication overhead than the Yao et al. solution. Its security varies, under a semi-honest model, proportionally to the density per node of its key graph representation. Actually, the resistance to collusions increases by rising the value of the security parameter k.

As regarding the computation and communication complexity, we have also obtained good results. We have decreased 50% of communications compared to Yao et al. proposal.

REFERENCES

[1] D. Yao, R. Tamassia, and S. Proctor, "Private distributed scalar product protocol with application to privacy-preserving computation of trust," in *IFIPTM 2007 – Joint iTrust and PST Conferences on Privacy, Trust Management and Security*, July 2007.

[2] A. Yao, "Protocols for secure computations," in *the twenty-third annual IEEE Symposium on Foundations of Computer Science*. IEEE Computer Society, 1982, pp. 160–164.

[3] O. Goldreich, "Secure multi-party computation," Final Draft, 2002.

[4] O. Goldreich, S. Micali, and A. Wigderson, "How to play any mental game," in *STOC '87: Proceedings of the nineteenth annual ACM symposium on Theory of computing*. New York, NY, USA: ACM, 1987, pp. 218–229.

[5] W. Du and M. J. Atallah, "Secure multi-party computation problems and their applications: a review and open problems," in *NSPW '01: Proceedings of the 2001 workshop on New security paradigms*. New York, NY, USA: ACM, 2001, pp. 13–22.

[6] M. J. Atallah and W. Du, "Secure multi-party computational geometry." Springer-Verlag, 2001, pp. 165–179.

[7] B. Goethals, S. Laur, H. Lipmaa, and T. Mielikäinen, "On private scalar product computation for privacy-preserving data mining." in *Information Security and Cryptology - ICISC 2004, 7th International Conference, Seoul, Korea, December 2-3, 2004, Revised Selected Papers*, ser. Lecture Notes in Computer Science, C. Park and S. Chee, Eds., vol. 3506. Springer, 2004, pp. 104–120.

[8] M. Atallah, H. Elmongui, V. Deshpande, and L. Schwarz, "Secure supply-chain protocols," in *2003 IEEE Conference on Electronic Commerce (CEC)*. IEEE Computer Society, 06 2003, pp. 293–302.

[9] D. Chaum, "The dining cryptographers problem: Unconditional sender and recipient untraceability," *Journal of Cryptology*, vol. 1, pp. 65–75, 1988.

[10] I. Damgard and M. Jurik, "A generalisation, a simplification and some applications of paillier's probabilistic public-key system," in *PKC '01: Proceedings of the 4th International Workshop on Practice and Theory in Public Key Cryptography*. London, UK: Springer-Verlag, 2001, pp. 119–136.

[11] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," 1999, pp. 223–238.

[12] W. Diffie and M. E. Hellman, "New directions in cryptography," *IEEE Transactions on Information Theory*, vol. IT-22, no. 6, pp. 644–654, 1976.

[13] D. Boneh and R. J. Lipton, "Algorithms for black-box fields and their application to cryptography (extended abstract)," in *CRYPTO '96: Proceedings of the 16th Annual International Cryptology Conference on Advances in Cryptology*. London, UK: Springer-Verlag, 1996, pp. 283–297.

[14] U. Maurer and S. Wolf, "The relationship between breaking the diffie-hellman protocol and computing discrete logarithms," *SIAM Journal on Computing*, vol. 28, no. 5, pp. 1689–1721, Apr 1999.

[15] D. Boneh, "The decision diffie-hellman problem," in *ANTS-III: Proceedings of the Third International Symposium on Algorithmic Number Theory*. London, UK: Springer-Verlag, 1998, pp. 48–63.

[16] O. Goldreich, S. Goldwasser, and S. Micali, "How to construct random functions," *J. ACM*, vol. 33, no. 4, pp. 792–807, 1986.

[17] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone, *Handbook of Applied Cryptography*. CRC Press, 2001.

[18] E. Barker and J. Kelsey, "Recommendation for random number generation using deterministic random bit generators," U.S. DoC/National Institute of Standards and Technology, SP-800-90, 2006, http://csrc.nist.gov/publications/nistpubs/.

[19] S. Patel and G. S. Sundaram, "An efficient discrete log pseudo random generator," in *CRYPTO '98: Proceedings of the 18th Annual International Cryptology Conference on Advances in Cryptology*. London, UK: Springer-Verlag, 1998, pp. 304–317.

[20] R. Gennaro, "An improved pseudo-random generator based on the discrete logarithm problem," *J. Cryptology*, vol. 18, no. 2, pp. 91–110, 2005.

[21] R. Steinfeld, J. Pieprzyk, and H. Wang, "On the provable security of an efficient rsa-based pseudorandom generator," in *ASIACRYPT*, 2006, pp. 194–209.

[22] M. Naor and O. Reingold, "Number-theoretic constructions of efficient pseudo-random functions," in *FOCS '97: Proceedings of the 38th Annual Symposium on Foundations of Computer Science*. Washington, DC, USA: IEEE Computer Society, 1997, p. 458.

[23] A. Pfitzmann and M. Köhntopp, "Anonymity, unobservability, and pseudonymity - a proposal for terminology." in *Workshop on Design Issues in Anonymity and Unobservability*, 2000, pp. 1–9.

[24] M. Waidner, "Unconditional sender and recipient untraceability in spite of active attacks," in *EUROCRYPT '89: Proceedings of the workshop on the theory and application of cryptographic techniques on Advances in cryptology*. New York, NY, USA: Springer-Verlag New York, Inc., 1990, pp. 302–319.

[25] J. Bos and B. den Boer, "Detection of disrupters in the dc protocol," in *EUROCRYPT '89: Proceedings of the workshop on the theory and application of cryptographic techniques on Advances in cryptology*. New York, NY, USA: Springer-Verlag New York, Inc., 1990, pp. 320–327.

[26] S. H. Low and N. F. Maxemchuk, "Modeling cryptographic protocols and their collusion analysis," in *Proceedings of the First International Workshop on Information Hiding*. London, UK: Springer-Verlag, 1996, pp. 169–184.

[27] J.-J. Quisquater, L. Guillou, M. Annick, and T. Berson, "How to explain zero-knowledge protocols to your children," in *CRYPTO '89: Proceedings on Advances in cryptology*. New York, NY, USA: Springer-Verlag New York, Inc., 1989, pp. 628–631.