Efficient Private Multi-Party Computations of Trust

in the Presence of Curious and Malicious Users [*]
(Preliminary Version)

Shlomi Dolev
Ben-Gurion University
Beer-Sheva, 84105, Israel
dolev@cs.bgu.ac.il

Niv Gilboa
Ben-Gurion University
Beer-Sheva, 84105, Israel
niv.gilboa@gmail.com

Marina Kopeetsky
Sami Shamoon College
Beer-Sheva, 84105, Israel
marinako@sce.ac.il

ABSTRACT

Schemes for multi-party trust computation are presented. The schemes do not make use of a Trusted Authority. The schemes are more efficient than previous schemes by the number of messages exchanged which is proportional to the number of participants rather than to a quadratic number of the participants. We note that in our schemes the length of each message may be larger than the message length of previous schemes. The calculation of a trust, in a specific user by a group of community members, starts upon a request of an initiating user. The trust computation is provided in a completely distributed manner, while each user calculates its trust value privately. Given a community C and its members (users) $U_1, \dots, U_n$, we present computationally secure schemes for trust computation. The first Accumulated Protocol AP computes the average trust in a specific user U_t upon the trust evaluation request initiated by a user U_n. The exact trust values of each queried user are not disclosed to U_n. The next Weighted Accumulated protocol *WAP* protocol generates the average weighted trust in a specific user U_t taking into consideration the unrevealed trust that U_n has in each user participating in the trust process evaluation. The Vector Protocol VP outputs a set of the exact trust values given by the users without linking the user that contributed a specific trust value to the trust this user contributed. The obtained vector of trust values assists in removing outliers. Given the set of trust values, the outliers which provide extremely low or high trust values, can be removed from the trust evaluation process. We extend our schemes to the case when the initiating user U_n can be compromised by the adversary, and we introduce the Multiple Private Keys $MPKP$ and the Multiple Private Keys Weighted $MPWP$ protocols for computing average unweighted and weighted trust, respectively. The computation of all our algorithms requires the transmission of $O(n)$ (large) messages.

[*]Supported by Deutsche Telekom Laboratories at Ben-Gurion University of the Negev, Israel.
Partially supported by Rita Altura Trust Chair in Computer Sciences, the ICT Programme of the European Union under contract number FP7-215270 (FRONTS), Lynne and William Frankel Center for Computer Sciences, and the internal research program of the Sami Shamoon College of Engineering.

1. INTRODUCTION

The purpose of this paper is generating new schemes for the decentralized reputation systems. These schemes do not make use of a Trusted Authority to compute trust in a particular user by the community of users. Our purpose is to compute trust while preserving user privacy. The use of the homomorphic cryptosystems in general Multiparty Computation (MPC) model is presented in [5]. In [5] it is demonstrated that given keys for any sufficiently efficient homomorphic cryptosystem, general MPC protocols for n players can be devised which are secure against an active adversary that corrupts any minority of the players. The problem stated and solved in [5] is as follows: given encryptions of two numbers, say a and b (where each player knows only its input), compute securely an encryption of $c = ab$. The correctness of the result is verified. The total number of bits sent is $O(nkC)$, where k is a security parameter and C is the size of a Boolean circuit computing the function to be securely evaluated. An earlier scheme proposed in [8] with the same complexity was only secure for passive adversaries. Earlier protocols had complexity at least quadratic in n. In [5] two examples of threshold homomorphic cryptosystems that lead to the claimed communication complexity are presented. The proposed schemes are based on public key infrastructure and use Zero Knowledge proofs (ZKP) as building blocks. When compared to [5], our schemes privately compute the average unweighted (additive) and weighted (non additive) characteristics, respectively without using such relatively complicated to implement techniques as ZKP.
The closest works to our work are [18] and [13]. In [18] several privacy and anonymity preserving protocols are suggested for Additive Reputation System. A decentralized reputation system is defined as additive/non additive ([18]) if feedback collection, combination, and propagation are implemented in a decentralized way, and combination of feedbacks provided by agents is calculated in an additive/non additive manner, respectively. The authors state that supporting perfect privacy in decentralized reputation systems is impossible, nevertheless they present alternative probabilistic schemes for preserving privacy. A probabilistic "witness selection" method is proposed in [18] in order to reduce the risk of selecting dishonest witnesses. Two schemes are proposed. The first scheme is very efficient in terms of communication overhead, nevertheless this scheme is vulnerable to collusion of even two witnesses. The second scheme is more resistant toward curious users, but still vulnerable to collusions. It is based on a secret splitting scheme. This scheme provides secure protocol based on the verifiable secret sharing scheme ([19]) derived from Shamir's secret sharing scheme ([20]). The number of dishonest users is heavily restricted and must be no more than $\frac{n}{2}$, where n is the number of contributing users. The communication overhead of this scheme is

rather high and requires $O(n^3)$ messages.
Enhanced model for reputation computation that extends the results of [18] is introduced in [13]. The main enhancement of [18] is that non additive (weighted) trust and reputation can be computed privately in Non Additive Reputation System. Three algorithms for computing non additive reputation are proposed in [13]. The algorithms have various degrees of privacy and different level of protection against adversarial users. These schemes are computationally secure regardless the number of dishonest users.
We propose new efficient trust computation schemes that can replace any of the above schemes. Our schemes enable the initiating user to compute unweighted (additive) and weighted (non additive) trust with low communication complexity of $O(n)$ (large) messages.
Our contribution. We present new efficient schemes for calculating a trust in a specific user by a group of community members upon a request of initiating user. The trust computation is provided in a completely distributed manner, while each user calculates its trust value privately. The user privacy is preserved in a computationally secure manner. Assume a community of users $C = \{U_1,\ U_2, ...,\ U_n\}$. Let U_n be an initiating user. The goal of U_n is to get the assessment of the trust in a certain user, U_t by a group consisting of $U_1,\ U_2, ...,\ U_{n-1}$ users from C. The first Accumulated Protocol AP calculates the average trust (or the sum of trust levels) in the user U_t. The AP protocol is based on a computationally secure homomorphic cryptosystem, e.g., the Paillier cryptosystem [17] which provides homomorphic encryption of the secure trust levels $T_1, \ldots,\ T_{n-1}$ calculated by each user $U_1,\ U_2, ...,\ U_{n-1}$ from C. The AP protocol satisfies the features of the Additive Reputation System [18] and does not take into consideration $U_n^{\prime}s$ subjective trust values in the queried users $U_1,\ U_2, ...,\ U_{n-1}$. The Weighted Accumulated Protocol *WAP* carries out non additive trust computation. *WAP* outputs the weighted average trust which is based on the trust given by the initiating user U_n in each C member participating in the feedback. The *WAP* protocol is the enhanced version of the AP protocol. The AP and *WAP* protocols cope with curious adversary and are restricted to the case of uncompromised initiating user U_n. The Multiple Private Keys $MPKP$ and Multiple Private Keys Weighted $MPWP$ protocols use additional communication to relax the condition that the initiating user U_n is uncompromised and provide average unweighted and weighted trust private computation, respectively.
Compared with the recent results in [18] and [13], our schemes have several advantages.
Private Trust scheme is resistant against either curious or semi-malicious users. The AP and *WAP* protocols preserve user privacy in a computationally secure manner. Our protocols cope with any number of curious but honest adversarial users. Moreover, the VP protocol is resistant against semi-malicious users which return false trust values. The VP protocol supports outliers removal. The general case when the initiating user U_n can be compromised by the adversary is addressed by $MPKP$ and $MPWP$ protocols. Unlike our model, [18] suggests protocols resistant against curious agents which only try to collude in order to reveal private trust information. Moreover, the reputation computation in some of the algorithms of [13] contains a random parameter that reveals information about the reputation range of the queried users.
Low communicational overhead. The proposed schemes require only $O(n)$ large messages sent, while the protocols of [18] and [13] require $O(n^3)$ communication messages.
No limitations on the number of curious users. The computational security of the proposed schemes does not depend on the number of the curious users in the community. Moreover, the privacy is preserved regardless of the size of the coalition of the curious users. Note that the number of the curious users should be no greater than half of the community users in the model presented in [18].

Paper organization. The formal system description appears in Section 2. The computationally resistant against curious but honest adversary private trust protocol AP, is introduced in Section 3. The enhanced version of AP, *WAP* is presented in Section 4. The resistant against semi-malicious users VP protocol, and the scheme for removing outliers are presented in Section 5. The generalized $MPKP$ protocol and the weighted $MPWP$ protocol are introduced in Section 6. Conclusions appear in Section 7.

2. PRIVATE TRUST SETTINGS

The purpose of this paper is to generate the new schemes for the private trust computation within a community. The contribution of our work is as follows: (a) The trust computation is performed in a completely distributed manner without involving a Trusted Authority. (b) The trust in a particular user within the community is computed privately. The privacy of trust values, held by the community users is preserved given standard cryptographic assumptions, when the adversary is computationally bounded. (c) The proposed protocols are resistant against curious but honest poly-bounded k-listening adversary Ad [7]. Such an adversary, Ad may perform the following: Ad may trace all the network links in the system and Ad may compromise up to k users, $k < n$.
We require that an adversary Ad compromising an intermediate node can only learn the node's trust values and an adversary Ad compromising the initiating node U_n can learn the output of the protocol, namely the average trust.
We distinguish between two categories of adversaries: honest but curious adversaries, and semi-malicious adversaries [18]. An honest but curious k-listening adversary follows the protocol by providing correct input, nevertheless it might try to learn trust values in different ways, including collusion by at most k compromised users. While an honest but curious adversary does not try to modify the correct output of the protocol, a semi-malicious adversary may provide dishonest input in order to bias the average trust value. A Vector Protocol that can cope with such an adversary in the cost of larger message is omitted from this extended abstract.
Let $C = U_1, \ldots,\ U_n$ be a community of users such that each pair of users is connected via an authenticated channel. Assume that the purpose of a user U_n from C is to get the unweighted T_t^{avr} or weighted average trust wT_t^{avr} in a specific user U_t evaluated by the community of users.
Denote by $T^i,\ i = 1..n$ the trust of user U_i in U_t, and by $T_t^{avr} = \frac{\sum_{i=1}^{n} T^i}{n}$ and $wT_t^{avr} = 1/10 \sum_{i=1}^{n} w_i T^i$ the unweighted and weighted average trust in U_t, respectively. Here $w_i = 1, 2,\ \ldots, 10$ is the subjective trust of the initiating user U_n in U_i in the form of an integer that facilitate our secure computation. In the sequel we always assume that w_i is an integer in this range. Denote by M_t the message sent by U_{init} to the first member of the community C.
Our definitions of computational indistinguishability, simulation and private computation follow the definitions of [10]. Informally speaking, two probability ensembles are *computationally indistinguishable* if no polynomial time, probabilistic algorithm can decide with non-negligible probability if a given input is drawn from the first or the second ensemble. A distributed protocol computes a function *f privately* if an adversary cannot obtain any information on the input and output of other parties, beyond what is implicit in the adversary's own input and output. The way to prove that a protocol is private is to show that there exists a polynomial time, prob-

abilistic *simulator* that receives as input the same input and output as an adversary and generates a string that is computationally indistinguishable from the whole view of the adversary, including every message that the adversary received in the protocol. Intuitively, the existence of a simulator implies that the adversary learns nothing from the execution of the protocol except for its input and output.

The main tool we use in our schemes is public-key, homomorphic encryption. In such an encryption scheme there is a modulus M and an efficiently computable function ϕ that maps a pair of encrypted values $(E_K(x), E_K(y))$, where $0 \leq x, y < M$, to a single encrypted element $\phi(E_K(x), E_K(y)) = E_K(x + y \bmod M)$. In many homomorphic encryption systems the function ϕ is multiplication modulo some integer N. Given a natural number c and an encryption $E_K(x)$, it is possible to compute $E_K(c \cdot x \bmod M)$, without knowing the private key. Set $\beta = E_K(1)$ and let the binary representation of c be $c = c_k c_{k-1} \dots c_0$. Go over the bits $c_k, \dots, c_0$ in descending order. If $c_j = 0$ set $\beta = \phi(\beta, \beta)$ and if $c_j = 1$ set $\beta = \phi(\phi(\beta, \beta), E_K(x))$. If ϕ is modular multiplication, this algorithm is identical to standard modular exponentiation. There are quite a few examples of homomorphic encryption schemes known in the cryptographic literature, including [12, 3, 15, 16] and [17]. There are also systems that allow both addition and multiplication of two encrypted plaintexts, e.g. [4] where only a single multiplication is possible for a pair of ciphertexts, and [9]. All of these examples of homomorphic cryptosystems are currently assumed to be semantically secure [12].

3. ACCUMULATED PROTOCOL AP

The AP protocol may be based on any homomorphic encryption scheme such that the modulus N satisfies $N > \sum_{i=1}^{n} T_i$. We illustrate the protocol by using the semantically secure Paillier cryptosystem [17]. This cryptosystem possesses a homomorphic property and is based on the Decisional Composite Residuosity assumption.

Let p and q be large prime numbers, and $N = pq$. Let g be some element of $Z^*_{N^2}$. Note that the base g should be chosen properly by checking whether $gcd(L(g^\lambda mod\ N^2), N) = 1$, where $\lambda = lcm(p-1, q-1)$, and the L function is defined as $L(u) = \frac{u-1}{N}$. The public key is the (N, g) pair, while the (p, q) pair is the secret private key. The ciphertext c for the plaintext message $m < N$ is generated by the sender as $c = g^m r^N \bmod N^2$, where $r < N$ is a randomly chosen number. The decryption is performed as $m = \frac{L(c^\lambda \bmod N^2)}{L(g^\lambda \bmod N^2)} \bmod N$ at the destination.

Our schemes are based on the homomorphic property of the Paillier cryptosystem. Namely, the multiplication of two encrypted plaintexts m_1 and m_2 is decrypted as the sum $m_1 + m_2 \bmod N$ of the plaintexts. Thus, $E(m_1) \cdot E(m_2) \equiv E(m_1 + m_2 \bmod m) \bmod N^2$ and $E(m_1)^{m_2} \equiv E(m_1 \cdot m_2 \bmod N) \bmod N^2$.

The AP protocol is described in Figures 1 and 2.

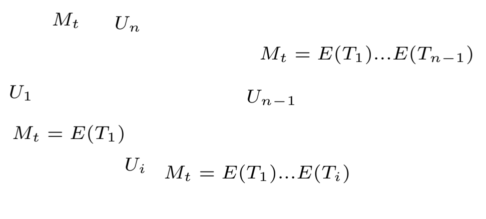

Figure 1: Accumulated protocol AP.

Assume that the initiating user U_n has generated a pair of its public and private keys as described above, and it has shared its public key with each community user. Then, U_n initializes to 1 the single entry trust message M_t and sends it to the first U_1 user (lines 1-3). Upon receiving the message M_t each node U_i encrypts its trust in U_t as $E(T_i) = g^{T_i} r_i^N \bmod N^2$. Here T_i is a secret $U_i's$ trust level in U_t, and r_i is a randomly generated number. The $U_i's$ output is accumulated in the accumulated variable A multiplying its current value by the new encrypted $U_i{-}th$ trust $E(T_i)$ from the $i{-}th$ entry as $A = A \cdot (E(T_i))$. Then U_i sends the updated M_t message to the next user U_{i+1}. This procedure is repeated until all trust values are accumulated in A (lines 4-9). The final M_t message received by the the initiating user U_n is $M_t = A = \prod_{i=1}^{n} E(T_i) \bmod N^2$. As a result, the U_n user decrypts the value accumulated in the M message as the sum of trusts $S_t = D(M_t) = \sum_{i=1}^{n} T_i$. Hence the average trust is $T_t^{avr} = \frac{S_t}{n-1}$ (Figure 2, lines 10-12). Proposition 1 proves that AP is a computationally private protocol to compute the trust of a community in U_t.

```
1:  $AP$ Initialization :
2:    $U_n$ sets $A = 1$ and $M_t = A$
3:    $U_n$ sends $M_t$ to $U_1$
4:  $AP$ Execution :
5:    for $i = 1 \dots n-1$
6:      $A = A \cdot E(T_i) \bmod N^2$
7:      $M_t = A$
8:      $U_i$ sends $M_t$ to $U_{i+1}$
9:    end for
10:   Upon $M_t$ receipt at $U_n$
11:   $S_t = D(M_t) = \sum_{i=1}^{n-1} T_i$
12:   $T_t^{avr} = \frac{S_t}{n-1}$
```

Figure 2: Accumulated Protocol.

Proposition 1. Assume that an honest but curious adversary corrupts at most k users out of a community of n users, $k < n$. Then, AP privately computes T^{avr}, the average trust in user U_t.

Proof:
In order to prove the proposition, we have to prove that for every adversary there exists a simulator that given only the adversary's input and output, generates a string that is computationally indistinguishable from the adversary's view in AP.

Let $I = \{U_{i_1}, U_{i_2}, \dots, U_{i_k}\}$ denote the set of users that the adversary controls. Let $view_I^{AP}(X_I, 1^n)$ denote the combined view of all users in I. $view_I^{AP}$ includes the input, $X_I = \{T_{i_1}, \dots, T_{i_k}\}$, of all users in I, and a sequence of messages $E(\sum_{j=1}^{i_1} T_j), \dots, E(\sum_{j=1}^{i_k} T_j)$ received by users in I. A simulator cannot generate the exact sequence $E(\sum_{j=1}^{i_1} T_j), \dots, E(\sum_{j=1}^{i_k} T_j)$, since it does not have the input of uncorrupted users. Instead, the simulator chooses a random value α_j for any user $U_j \notin I$ from the distribution of trust values D. The simulator denotes $\alpha_{i_1} = T_{i_1}, \dots, \alpha_{i_k} = T_{i_k}$ and computes $E(\alpha_j)$ for $j = 1, \dots, n-1$. The simulator now computes: $\prod_{j=1}^{i_1} E(\alpha_j) \equiv E(\sum_{j=1}^{i_1} \alpha_j) \bmod N^2, \dots, \prod_{j=1}^{i_k} E(\alpha_j) \equiv E(\sum_{j=1}^{i_k} \alpha_j) \bmod N^2$. Hence, a simulator replaces $E(\sum_{j=1}^{i_k} T_j)$ by $E(\sum_{j=1}^{i_k} \alpha_j)$.

Assume towards a contradiction that there exists an algorithm DIS that distinguishes between the encryption of partial sums $E(\sum_{j=1}^{i_1} T_j), \cdots E(\sum_{j=1}^{i_k} T_j)$ of the correct trust values and the values $E(\sum_{j=1}^{i_1} \alpha_j), \cdots\ E(\sum_{j=1}^{i_k} \alpha_j)$ randomly produced by a simulator. We construct an algorithm B that distinguishes between the two sequences $E(T_1), \cdots\ E(T_{n-1})$ and $E(\alpha_1), \cdots, E(\alpha_k)$, contradicting the semantic security property of E. The input to algorithm B is a sequence of values $E(x_1), \cdots\ E(x_{n-1})$ and it attempts to determine whether the values $x_1, \dots, x_{n-1}$ are equal to the value $T_1, \dots, T_{n-1}$ that the users provide, or is a sequence of random value chosen from the distribution D. The algorithm B

computes for every $\ell = 1, \ldots, k$

$$\prod_{j=1}^{i_\ell} E(x_j) \equiv E(\sum_{j=1}^{i_\ell} x_j) \bmod N^2,$$

and provides the encryption of partial sums $E(\sum_{j=1}^{i_1} x_j), \ldots E(\sum_{j=1}^{i_k} x_j)$ as input to DIS. B returns as output the same output as DIS. Since the input of DIS is $E(\sum_{j=1}^{i_1} T_j), \ldots E(\sum_{j=1}^{i_k} T_j)$ if and only if the input of B is $E(T_1), \ldots\ E(T_{n-1})$, we have that B distinguishes between its two possible input distributions with the same probability that DIS distinguishes between its input distributions.

AP uses $O(n)$ messages.

4. WEIGHTED ACCUMULATED PROTOCOL WAP

The Weighted Accumulated *WAP* protocol, in addition to the AP protocol, generates the weighted average trust in a specific user U_t by the users in the community. The *WAP* protocol is based on an anonymous communications protocol proposed in [1] and on the homomorphic cryptosystem, e.g., Paillier cryptosystem [17]. It is described in Figures 4 and 3.

The initiating node U_n generates $n-1$ weights $w_1, \ldots, w_{n-1}$. Each w_i value reflects the $U_n's$ subjective trust level in U_i user. U_n initializes the accumulated variable, A, to 1, encrypts each w_i value by means of, e.g., the Paillier cryptosystem([17]) as $E(w_i) = g^{w_i} h^{r_{n,i}} (mod\ N^2)$, composes a Trust Vector $TV = [E(w_1)..E(w_{n-1})]$ and sends the message $M_t = (TV, A)$ to U_1. Here, as in the AP case, $p,\ q$ are large prime numbers which compose the Paillier cryptosystem, $N = (p-1)(q-1)$, and g and h are properly chosen parameters of the Paillier cryptosystem. $r_{n,i}$ is a random degree of h chosen by U_n for each U_i from C. Note that the AP protocol is the private case of the *WAP* protocol while all weights w_i are equal to 1.

As in the AP case the M_t message is received by the community users in the prescribed order. Each U_i user encrypts its weighted trust in U_t as $E(T_i) = E(w_i)^{T_i} E(\overline{0})$ and accumulates it in the accumulated variable A (lines 6-10).

```
1:  WAP Initialization:
2:    $U_n$ generates $TV = [w_1..w_{n-1}]$
3:    $U_n$ sets $A = 1$ and $M_t = (TV, A)$
4:  WAP execution:
5:    $U_n$ sends $M_t$ to $U_1$
6:    for $i = 1 \ldots n-1$
7:      $A = AE(w_i)^{T_i} E(\overline{0}) \bmod (N^2)$
8:      Delete $TV[i]$
9:      $U_i$ sends $M_t$ to $U_{i+1}$
10:   end for
11: Upon $M_t$ reception at $U_n$:
12:   $S_t = D(A) = \sum_{i=1}^{n} w_i T_i$
13:   $wT_t^{avr} = \frac{1}{10} \frac{S_t}{n-1}$
```

Figure 4: Weighted Accumulated Protocol *WAP*.

Note that the multiplying by the random encryption of zero $E(\overline{0})$ ensures semantic security of the WAP protocol since the user's output cannot be distinguished from a simulated random string. As a result, the initiating user U_n receives the M_t message and decrypts the value accumulated in A as the weighted sum of trust $S_t = D(A) = \sum_{i=1}^{n-1} w_i T_i$. Hence, the average trust is equal to $wT_t^{avr} = 1/10 \sum_{i=1}^{n} w_i T^i$. Proposition 2 proves the privacy of the weighted average trust wT_t^{avr} in the U_t user by the community users in a computationally secure manner.

Proposition 2. *WAP protocol performs computationally secure anonymous computation of the weighted average trust wT^{avr} under the assumption of uncompromised initiating user U_n in the Non Additive Reputation System.*

Proof:
The proof is similar to the proof of Proposition 1. View of adversary includes the input of compromised users $T_{i_1}, \ldots,\ T_{i_k}$, trust vector TV, and the accumulated variable A. Each compromised user U_{i_j} from I receives $TV = [E(w_{i_j}),\ E(w_{i_{j+1}}) \ldots,\ E(w_n)]$ and $A = \prod_{i=1}^{i_j} E(w_i)^{T_i} E(\overline{0})$.
A simulator for the adversary simulates $view_I^{WAP}$ as follows. The simulator input $T_{i_1}, \ldots,\ T_{i_k}$ is the same as the input of the compromised users. A simulator chooses at random $v_1, \ldots,\ v_n$ according to a distribution W of weights, and $\widetilde{T_1}, \ldots,\ \widetilde{T_n}$ according a distribution D of trust values. Here $\widetilde{T_{i_1}} = T_{i_1}, \ldots,\ \widetilde{T_{i_k}} = T_{i_k}$. Due to the semantic security of the homomorphic cryptosystem, the encrypted random values $E(v_1), \ldots,\ E(v_n)$ are indistinguishable from the encrypted correct weights $E(w_{i_1}), \ldots,\ E(w_{i_n})$.
The randomization of any $U_i - th$ user output is performed by multiplying its secret $w_i^{T_i}$ by the random encryption of zero string $E(\overline{0})$. Given $E(w)$, the two values $E(w)^T$ and $E(u)$, where u is chosen at random from the distribution of wT, can be distinguished since T is chosen from a small domain of trust values. Given $E(w)$, the values $E(w)^T E(\overline{0})$ are distributed identically to an encryption $E(w)^T = E(wT \bmod N)$. Based on the semantic security of the homomorphic cryptosystem, $E(u)$ and $E(wT)$ cannot be distinguished even given $E(w)$.
WAP uses $O(n)$ messages each of length $O(n)$.

5. VECTOR PROTOCOL VP

The Vector Protocol VP is performed in two rounds (Figure 5). At the initialization stage U_n initializes the $n-1$-entry vector $TV[1..n-1]$ and sends it to the community of users in the prescribed order in the $M_t^w = (TV[1..n])$ message (lines 1-2).
At the first round upon message M_t^w reception each user U_i encrypts its trust T_i in the corresponding $TV[i]'s$ entry as $E(T_i) = g^{T_i} h^{r_i} (\bmod N^2)$, and sends the updated message M_t^w to the next user (lines 3-7).
The second round of the VP protocol is performed when the updated $TV[1..n-1]$ vector returns from the last querying user U_{n-1} to the first user U_1. Note that the TV vector does not visit the initiating node U_n after the first round execution. Each user U_i act as follows at the second round: (a) U_i performs a permutation of its $i-th$ entry with the randomly chosen $i_j - th$ entry, (b) U_i updates all entries by multiplying them by a random encryption of zero $E(\overline{0}) = h^{r_{0,i}}$. After that the newly updated M_t^w vector-message is sent to the next U_{i+1} user (lines 8-14).

As a result of the second round execution, the initiating node U_n receives the $TV[1..n]$ vector while each of its $TV[i_j]$ entry is equal to the $U_i - th$ encrypted trust T_i in

```
1:  Initialization:
2:    $U_n$ initializes $TV = [1..n-1]$
3:  Round 1:
4:    $U_n$ sends $M_t = TV[1..n-1]$ into $C$
5:    for $i = 1 \ldots (n-1)$
6:      $TV[i] = E(T_i)$
7:    end for
8:  Round 2:
9:    for $i = 1 \ldots (n-1)$
10:     $swap(TV[i], TV[i_j])$
11:   end for
12:   for $j = 1 \ldots (n-1)$
13:     $TV[j] = TV[j]E(\overline{0})$
14:   end for
15: Upon $M_t = (TV[1..n-1])$ reception at $U_n$:
16:   $D(M) = [T_1, ..\ T_{n-1}]$
```

Figure 5: Vector Protocol VP.

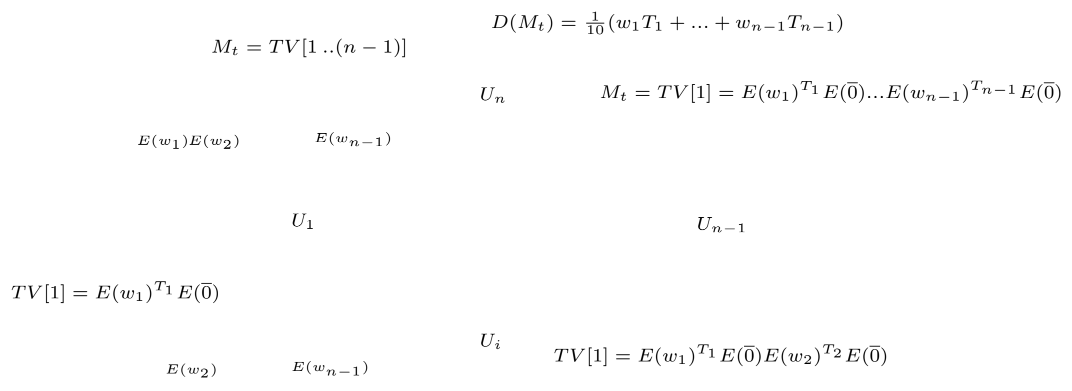
$D(M_t) = \frac{1}{10}(w_1T_1 + ... + w_{n-1}T_{n-1})$
$M_t = TV[1\,..(n-1)]$
U_n
$M_t = TV[1] = E(w_1)^{T_1}E(\overline{0})...E(w_{n-1})^{T_{n-1}}E(\overline{0})$
$E(w_1)E(w_2)$
$E(w_{n-1})$
U_1
U_{n-1}
$TV[1] = E(w_1)^{T_1}E(\overline{0})$
U_i
$TV[1] = E(w_1)^{T_1}E(\overline{0})E(w_2)^{T_2}E(\overline{0})$
$E(w_2)$
$E(w_{n-1})$

Figure 3: Weighted Accumulated protocol WAP.

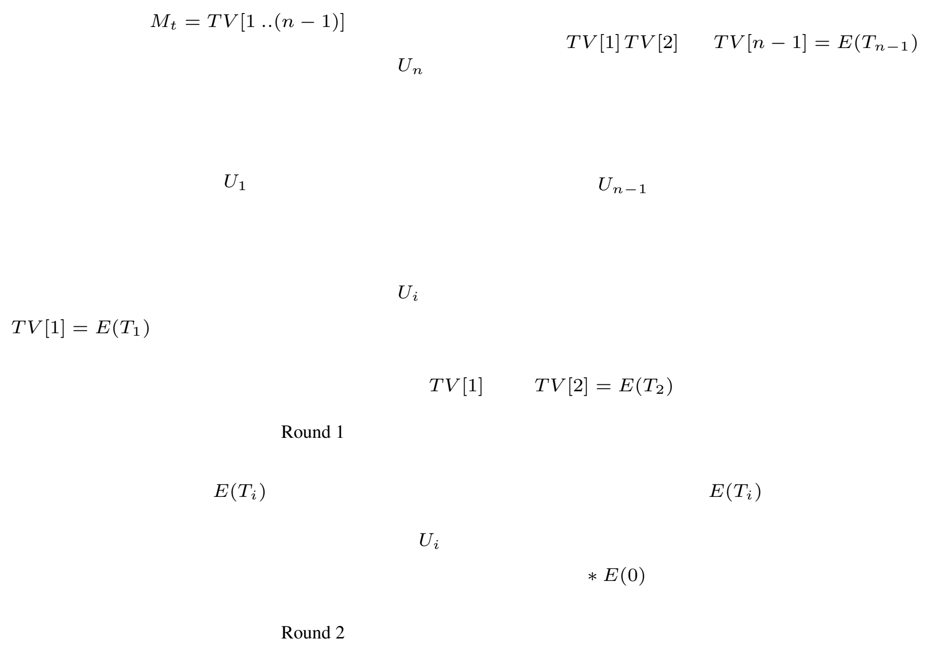

Figure 6: Vector Protocol VP.

U_t. Hence, by applying the decryption procedure, all encrypted trust values $T_1, \ \ldots, T_{n-1}$ are revealed (lines 15-16). Moreover, the random permutation performed at the second round, preserves the unlinkability of the user identity.
The Proposition 3 states the correctness of the VP protocol.

Proposition 3. VP protocol performs computationally secure computation of the exact private trust values under the assumption of the non compromised initiating user U_n in the Additive Reputation System.

The suitable for the Non Additive Reputation system weighted VP protocol can be implemented, as well. In order to generate the average weighted private trust, the initialized $TV[1..n]$ vector sent by U_n to the community of users in M_t message (Figure 5, lines 1-4) must be replaced by the vector of the encrypted weights $TV[1..n] = [E(w_1)..E(w_{n-1})]$ as in the *WAP* protocol. Moreover, the encryption performed by any user U_i should be $W[i] = E(w_i)^{T_i}$ as in the *WAP* protocol. The computationally secure privacy of the weighted VP protocol is a derived extension of the Proposition 3.

6. MULTIPLE PRIVATE KEYS PROTOCOL MPKP

The AP and *WAP* protocols introduced in the previous sections carry out private trust computation assuming that the initiating node U_n is not compromised and does not share its private key with other users. In the rest of this work we assume now that any community user, including U_n may be compromised by a poly-bounded k-listening curious adversary.

The generalized Multiple Private Keys Protocol $MPKP$ copes with this problem and outputs the average trust. The idea of the $MPKP$ protocol is as follows. During the initialization stage the U_n user initializes all entries of trust vector TV and accumulated vector AV to 1, sets the accumulated variable A to 1, and sends $M_t = (TV, AV, A)$ message to the first community user U_1 as in the previous protocols. During the first round of the $MPKP$ protocol execution each user U_i randomly fragments its secret trust T_i to a sum of $n-1$ shares, encrypts corresponding share by public key of each $U_j, \ j = 1..n-1$ user and accumulates its encrypted shares (multiplying each of them with the corresponding entries) in the accumulated vector AV. After the first round execution the updated AV vector does not return to the initiating user U_n. The AV vector visits each community user, while each U_i opens the $i-th$ entry (that is encrypted by $U_i - th$ public key) revealing a sum of decrypted shares, encrypts this sum by the public key of the initiating user U_n, accumulates this sum in the accumulated variable A, and deletes the $i-th$ entry of the AV vector.

```
 1: **MPKP Initialization:**
 2:    $U_n$ generates $TV = [1..1]$
 3:    $U_n$ sets $AV = [1..1]$, $A = 1$ and
       $M_t = (TV, AV, A)$
 4:    $U_n$ sends $M_t$ to $U_1$
 5: **Round 1:**
 6:    *for* $i = 1 \ldots (n-1)$
 7:      $T_i = \sum_{j=1}^{n-1} r_j^i$
 8:      *for* $j = 1 \ldots (n-1)$
 9:        $AV[j] = AV[j]E_j(r_j^i)$
10:      *end for*
11:      $U_i$ sends $M_t$ to $U_{i+1}$
12:    *end for*
13: **Round 2:**
14:    *for* $i = 1 \ldots (n-1)$
15:    $D_i(AV[i]) = \sum_{j=1}^{n-1} r_i^j$
16:    $A = AE_n(\sum_{j=1}^{n-1} r_i^j)$
17:    Delete $AV[i]$
18:    *end for*
19: **Upon $M_t = (A)$ reception at $U_n$:**
20:    $A = \prod_{i=1}^{n-1} E_n(\sum_{j=1}^{n-1} r_i^j)$
21:    $S_t = D_n(A)$
22:    $T_t^{avr} = \frac{S_t}{n-1}$
```

Figure 7: Multiple Private Keys Protocol MPKP.

The detailed description of the $MPKP$ protocol follows. Assume that each community user $U_i, \ i = 1..n-1$ generates its personal pair (P_i^+, P_i^-) of private and public keys. Denote by E_i and D_i the encryption and decryption algorithms produced by U_i. The private key (P_i^+) is kept secret, while the public key P_i^- is shared with all other users $U_1, \ldots, \ U_{i-1}, \ U_{i+1} \ldots \ U_n$. As in the previous schemes, the cryptosystem must be homomorphic. An additional requirement is that the homomorphism modulus, m, must be identical for all users. One possibility is to use the Benaloh cryptosystem ([2, 3]) for which many different key pairs are possible for every homomorphism modulus.

The system works as follows. Select two large primes p, q such that: $N \triangleq pq$, $m|p-1$, $\gcd(m, (p-1)/m) = 1$ and $\gcd(m, q-1) = 1$, which implies that m is odd. The density of such primes along appropriate arithmetic sequences is large enough to ensure efficient generation of multiple p, q (see [2] for details). Select $y \in \mathcal{Z}_N^*$ such that $y^{\phi(N)/m} \not\equiv 1 \bmod N$. The public key is N, y, and encryption of $M \in \mathcal{Z}_m$ is performed by choosing a random $u \in \mathcal{Z}_m^*$ and sending $y^M u^m \bmod N$. In order to decrypt, the holder of the secret key computes at a preprocessing stage $T_M \triangleq y^{M\phi(N)/m} \bmod N$ for every $M \in \mathcal{Z}_m$. Hence, m is small enough that m exponentiations can be performed. Decryption of z is by computing $z^{\phi(N)/n} \bmod N$ and finding the unique T_M to which it is equal.

The $MPKP$ protocol is performed in two rounds (Figure 7). The initialization procedure is shown in lines 1-4. The first round is the accumulation round while all users share their secret trust T_i values with other users. Upon a reception of a message M_t each user U_i proceeds as follows: (a) U_i chooses $r_1^i, \ldots, \ r_{n-1}^i$ uniformly at random such that $T_i = \sum_{j=1}^{n-1} r_j^i$; (b) U_i encrypts each $r_j^i, \ j = 1.. \ n-1$ by the public key P_j^- of the U_j user and multiplies it by the current value stored in $j-th$ entry of AV. Hence, the output AV vector contains the accumulated product $\prod_{k=1}^{n-1} E_j(r_j^k)$ in each $j-th$ entry (lines 5-12).

Upon M_t message reception at the second round each U_i user decrypts the corresponding $i-th$ entry by its private key P_i^+, computes the $\sum_{j=1}^{n-1} r_i^j$ sum, encrypts it by the $U_n's$ public key P_n^- as $E_n(\sum_{j=1}^{n-1} r_i^j)$, accumulates this sum in the accumulated variable A, deletes $i-th$ entry and sends the updated TV vector to the next U_{i+1} user. Note that the partial sum $\sum_{j=1}^{n-1} r_i^j$ that U_i decrypts reveals no information about correct trust values. As a result of the second round the initiating user U_n receives $A = \prod_{i=1}^{n-1} E_n(\sum_{j=1}^{n-1} r_i^j)$ (lines 13-19). U_n decrypts $\prod_{j=1}^{n-1} E_n(r_i^j)$, and computes the sum of trusts as $S_t = \sum_{i=1}^{n-1} \sum_{j=1}^{n-1} r_i^j$. Actually, the average trust T^{avr} is equal to $\frac{S_t}{n}$ (lines 20-22). Proposition 4 states the privacy of $MPKP$ protocol. The communication complexity of $MPKP$ protocol, is $O(n)$ messages, each of length $O(n)$.

Proposition 4. $MPKP$ protocol performs computationally secure computation of the exact private trust values in the Additive Reputation System. No restriction is imposed on the initiating user U_n.

The last introduced protocol is the $MPWP$ for the weighted aver-

age trust wT_t^{avr} computation. The idea of the $MPWP$ is as follows. During the initialization stage the U_n user generates a vector TV such that each $i-th$ entry contains U_i-th weight w_i encrypted by U_n-th public key. U_n sends TV and a $(n-1)\times(n-1)$ matrix SM with all entries initialized to 1 to the first community user U_1 as in the previous protocols. During the first round of the $MPWP$ execution each U_i computes its encrypted weight in the power of its secret trust $E_n(w_i)^{T_i}$, multiplies it by a randomly chosen number (bias) z_i, and accumulates the product in the accumulated entry (by multiplying the entry by the obtained result). In addition, U_i fragments its bias z_i into $n-1$ shares, encrypts each $j-th$ share by the public key of U_j, and inserts it in the $j-th$ location of $i-th$ matrix row. At the end of the first round U_n decrypts the total biased weighted trust. The total random bias is removed during the second round of the $MPWP$ execution when each U_j decrypts the entries of $j-th$ matrix column, encrypts the sum of these values by the public key of the initiating user, accumulates it in an accumulation variable A, and deletes $j-th$ column. The details follow. The initiating user U_n starts the first round by generating the encryption of the $n-1$ entries trust vector $TV=[E_n(w_1)..E_n(w_{n-1})]$. Note, that each weight w_i is encrypted by U_n-th public key P_n^-. In addition, U_n initializes to 1 each entry of the $(n-1)\times(n-1)$ matrix of shares SM. The M_t^w message sent by U_n to the community users is $M=(TV,SM)$. Upon the TV vector reception each U_i user proceeds as follows: (a) U_i computes $E_n(w_i)^{T_i}\cdot z_i$. Here z_i is a randomly generated by U_i number that provides the secret bias. (b) U_i accumulates its encrypted weighted trust in the accumulated variable A by setting $A=A\cdot E_n(w_i)^{T_i}\cdot z_i$. After that, the $i-th$ entry of TV is deleted. (c) U_i shares z_i in the $i-th$ row of the SM shares matrix as $SM[i][]=[E_1(z_i^1)..E_{n-1}(z_i^{n-1})]$. At the end of the first round U_n receives the $TV[]$ entry that is equal to the encrypted by its public key biased product $BT=\prod_{j=1}^{n}E_n(w_i)^{T_i}z_i$ and the updated shares matrix SM while $SM[i][j]=E_j(z_i^j)$. Actually, the decryption procedure applied on the $TV[]$ vector outputs the decrypted sum $D(TV[])=\sum_{i=1}^{n-1}w_iT_i+\sum_{i=1}^{n-1}z_i$.
A second round is performed in order to subtract the random bias $\sum_{i=1}^{n-1}z_i$ from the correct weighted average trust wT^{avr}. The second round of the $MPWP$ protocol is identical to the corresponding round of the $MPKP$ protocol. Upon reception of the SM matrix each user U_i decrypts the corresponding $i-th$ column $E_i(z_1^i)\ E_i(z_2^i)\dots\ E_i(z_{n-1}^i)$, encrypted by all community users by U_i-th public key P_i^-. Each $U_i,\ i=1..\ n-1$ computes the sum of the partial shares $PSS_i=\sum_{j=1}^{n-1}z_j^i$, encrypts it by U_n-th public key P_n^-, and accumulates it in the accumulated variable A. After that $i-th\ SM's$ column $SM[][i]$is deleted. As a result of the second round, the initiating user U_n receives the accumulated variable $A=\prod_{i=1}^{n-1}E_i(PSS_i)$. The encrypted bias BT is decrypted as $D(A)=\sum_{i=1}^{n-1}\sum_{j=1}^{n-1}z_j^i$. Finally, the weighted average trust wT^{avr} is equal to $wT^{avr}=TV-A$.
The private trust computation carried out by the $MPKP$ and the $MPWP$ protocols is preserved in the computationally secure manner due to the following reasons:
(a) Each community user U_i fragments its trust T_i randomly into $n-1$ shares (Figure 7, lines 6-8).
(b) Each r_i^j encrypted by U_i by U_j-th public key P_j^-, shared with each $U_j,\ j=1,\dots,\ n-1$ user and accumulated in the TV vector, reveals no information about the exact T_i value to U_j (lines 9-14).
(c) The decryption performed by each $U_i,\ \ i=1,\dots,\ n-1$ by its private key P_i^+ at the second round, outputs the sum of the partial shares $D_i(TV[i])=\sum_{j=1}^{n-1}r_j^i$ of all community users. In essence, the $\sum_{j=1}^{n-1}r_j^i$ value reveals no information about the secret trust values $T_1,\dots,\ T_{i-1},\ T_{i+1},\dots,\ T_{n-1}$.
(d) The encryption $E_n(\sum_{j=1}^{n-1}r_j^i)$ of the partial shares sum performed by each U_i with the initiating node U_n public key P_n^- and accumulated in A, can be decrypted by U_n only.
(e) Assume a coalition $U_{j_i},\dots,\ U_{j_{i+k-1}}$ of at most $k<n$ curious adversarial users, possibly including the initiating user U_n. Then the exact trust values revealed by the coalition, are the coalition members trust only. The privacy of the uncorrupted users is preserved by the homomorphic encryption scheme which generates for each user its secret private key, and by the random fragmentation of the secret trust.
In $MPWP$ $O(n)$ messages of length $O(n^2)$ are sent.

7. CONCLUSIONS

We derived a number of schemes for private computation of trust in a given user by community of users. Trust computation is performed in a fully distributed manner without involving a Trusted Authority. The AP and *WAP* protocols are computationally secure under the assumption of uncompromised initiating user U_n. The AP and *WAP* protocols compute the average unweighted and weighted trust, respectively. The generalized $MPKP$ and $MPWP$ protocols relax the assumption of the non compromised U_n. They carry out the private unweighted and weighted trust computation, respectively without limitations imposed on U_n. The number of messages sent in the proposed protocols is $O(n)$ (large) messages. The schemes proposed in this paper are not restricted to trust computation only. They might be extended to other models that compute privately sensitive information with only $O(n)$ messages.
In case the trust is represented by several values rather then a single value, one can apply our techniques to each such value independently.

8. REFERENCES

[1] A. Beimel, S. Dolev, “Buses for Anonymous Message Delivery”, Journal of Cryptology, Vol. 16, No. 1, pages 25-39, 2003.

[2] J. Benaloh, “Verifiable Secret-Ballot Elections”, *Ph.D. thesis*, Yale University, 1987.

[3] J. Benaloh, “Dense Probabilistic Encryption”, *Proceedings of the Workshop on Selected Areas of Cryptography*, pages 120-128, Kingston, May 1994.

[4] D. Boneh, Eu-Jin Goh, K. Nissim, “Evaluating 2-DNF Formulas on Ciphertexts”, *TCC*, pages 325-341, 2005.

[5] R. Cramer, I. B. Damgard, J. Buus Nielsen, “Multiparty Computation from Threshold Homomorphic Encryption”, *EUROCRYPT ’01: Proceedings of the International Conference on the Theory and Application of Cryptographic Techniques*, pages 280-299, 2001.

[6] D. Dolev, N. A. Lynch, S. S. Pinter, E. W. Stark, W. E. Weihl, “Reaching Approximate Agreement in the Presence of Faults”, *Journal of the Association for Computing Machinery*, Vol. 33, No. 3, pages 499-516, 1986.

[7] S. Dolev, R.Ostrovsky, “Xor-Trees for Efficient Anonymous Multicast and reception”, *ACM Transactions on Information and Systems Security*, Vol. 3, No. 2, pages 63-84, 2000.

[8] M. Franklin, S. Haber, “Joint Encryption and Message-Efficient Secure Computation”, *Journal of Cryptology*, Vol. 9, No. 4, pages 217-232, 1996.

[9] C. Gentry, “Fully homomorphic encryption using ideal lattices”, *STOC*, pages 169-178, 2009.

[10] O. Goldreich, *“Foundations of Cryptography: Volume 1, Basic Tools”*, Cambridge University Press, New York, NY, USA, 2000.

[11] O. Goldreich, *“Foundations of Cryptography: Volume 2, Basic Applications”*, Cambridge University Press, New York, NY, USA, 2004.

[12] S. Goldwasser, S. Micali, “Probabilistic encryption”,*Journal of Computer and systems science*, Vol.28, pages 108-119, 2004.

[13] E. Gudes, N. Gal-Oz, A. Grubshtein, “Methods for Computing Trust and Reputation while Preserving Privacy”, Accepted for publication in *Proceedings of 23rd Annual IFIP WG 11.3 Working Conference on Data and Applications Security*, 2009.

[14] A. Josang, R. Ismail “The Beta Reputation System”, 2002.

[15] D. Naccache and J. Stern, “A New Public Key Cryptosystem Based on Higher Residues”, *ACM Conference on Computer and Communications Security*, pages 59-66, 1998.

[16] T. Okamoto, S. Uchiyama, “A New Public-Key Cryptosystem as Secure as Factoring”, *EUROCRYPT 1998*, pages 308-318, 1998.

[17] P. Paillier, “Public-Key Cryptosystems Based on Composite Degree Residuosity Classes”, *EUROCRYPT 1999*, pages 223-238.

[18] E. Pavlov, J. S. Rosenschein, Z. Topol, “Supporting Privacy in Decentralized Additive Reputation Systems”, “iTrust 2004”, LNCS 2995, pp. 108-119, 2004.

[19] T. P. Pedersen, “Non-Interactive and Information Theoretic Secure Verifiable Secret Sharing”, 1991.

[20] A. Shamir, “How to Share a Secret”, *Commun. ACM*, No. 22, Vol. 11, pages 612-613, 1979.