Vulnerabilities of decentralized additive reputation systems regarding the privacy of individual votes

Antonis Michalas[1, 2, *]
Tassos Dimitriou[1]
Thanassis Giannetsos[1]
Nikos Komninos[1]
Neeli R. Prasad[2]

[1]Athens Information Technology, Algorithms & Security Group, Athens, Greece
[2] Department of Electronic Systems, Aalborg University, Denmark

*Now working in the Faculty of Science and Technology, University of Westminster, UK

This is an author's accepted manuscript of an article published in Wireless Personal Communications, 66 (3) pp. 559-575, October 2012.

The final publication is available at Springer via doi:10.1007/s11277-012-0734-z

Vulnerabilities of Decentralized Additive Reputation Systems Regarding the Privacy of Individual Votes

Antonis Michalas[1,2], Tassos Dimitriou[1], Thanassis Giannetsos[1], Nikos Komninos[1], and Neeli R. Prasad[2]

[1] Athens Information Technology, Algorithms & Security Group, Athens, Greece
{amic,tdim,agia,nkom}@ait.edu.gr
[2] University of Aalborg, Department of Electronic Systems, Aalborg, Denmark
np@es.aau.dk

Abstract. In this paper, we focus on attacks and defense mechanisms in additive reputation systems. We start by surveying the most important protocols that aim to provide privacy between individual voters. Then, we categorize attacks against additive reputation systems considering both malicious querying nodes and malicious reporting nodes that collaborate in order to undermine the vote privacy of the remaining users. To the best of our knowledge this is the first work that provides a description of such malicious behavior under both semi-honest and malicious model. In light of this analysis we demonstrate the inefficiencies of existing protocols.

Key words: Decentralized Reputation Systems, Security, Voter Privacy

1 Introduction

During the last few years, online communities have experienced a significant amount of growth. Among the main factors contributing to their increased popularity is user-friendliness and ease of understanding but also accessibility and availability of information and services. These characteristics make it easy, even for novice users, to exchange information with strangers in way that guarantees a certain degree of anonymity. However, these features can be abused by malicious users who can either impersonate other entities and launch various types of attacks under fake identities or provide negative feedback for well behaving users, irrespective of the service they have received.

Reputation systems have been proposed as the means to protect online communities from such malicious behavior. The main goal of a reputation system is to reduce the risk involved in interactions between strangers by collecting, distributing and aggregating feedback about participants' past behavior in order to predict possible future behavior and identify dishonest community members [6]. However, one concern about reputation systems, which has received relatively little attention in the literature, is that of *feedback providers' privacy*. Although there are many reputation and trust establishment schemes, only some of them deal with the problem of securing the votes or ratings of participating nodes. This lack of privacy can lead to several problems including the proper functioning of the network. For example, it has been observed in [5] that users of a reputation system may avoid providing honest feedback in fear of retaliation, if reputation scores cannot be computed in a privacy-preserving manner. Additionally, the absence of schemes that provide (partial) privacy in decentralized environments, such as ad hoc networks, is even larger.

Hence the development of reputation protocols that can be used to provide anonymous feedback is essential to the survivability of online communities and electronic marketplaces. In some sense, provision of anonymous feedback to a reputation system is analogous to that of anonymous voting in electronic elections. It potentially encourages truthfulness by guaranteeing secrecy and freedom from explicit or implicit influence. Although this freedom might be exploited by dishonest feedback providers, who tend to report exaggerated feedbacks, it seems highly beneficial for honest users, protecting the latter from being influenced by malicious behavior [6].

In this invited paper we present a theoretical analysis of the vulnerabilities of existing *decentralized* additive reputation systems, regarding the privacy of individual votes. A decentralized system is one in which there is no central repository to collect and report reputation scores. In such a system, the users *themselves* are responsible for maintaining a local repository of trust ratings and providing feedback when queried by other users. To the best of our knowledge this is the first work that provides a description of malicious behavior/attacks against such systems. We use this categorization to demonstrate the inefficiencies of existing protocols in the hope to spawn further research in the area.

The paper is organized as follows. In Section 2 we define the problem of secure trust aggregation and we define the basic terms that we use in the rest of the paper. In Section 3 we present the details of the most important protocols that allow ratings to be (partially) private in decentralized additive reputation systems under the semi-honest model while in Section 4 we move one step further and we present protocols that preserve voters privacy under the malicious model. In Section 5, we present attacks that can break the privacy of the presented protocols and in Section 6 we conclude the paper.

2 Problem Statement & Definitions

We start by providing a definition of decentralized additive reputation systems as described in [6].

Definition 1: *A Reputation System R is said to be a Decentralized Additive Reputation System if it satisfies the following two requirements:*

1. *Feedback collection, combination and propagation are implemented in a decentralized way.*

2. *Combination of feedbacks provided by nodes is calculated in an additive manner.*

Regarding trust management and the use of reputation schemes in networks, we observe two general methods for collecting information on other nodes. Each member of a network evaluates other nodes based on first-hand information (Direct Trust) or second-hand (Third-Party Trust) information. A framework for assessing trust between neighboring nodes is based on direct observations, while trust between nodes that have no information from previous interactions, are built through a combination of information from intermediate nodes. The problem (in its general form) of secure private voting in decentralized environments is as follows:

Basic Problem Statement: *A querying node A_q, receives a service request from a target node A_t. Since A_q has incomplete information about A_t, she asks other nodes in the network to give their votes about A_t. Let $U = \{U_1, \cdots, U_n\}$ be the set of all nodes that will provide an opinion to A_q. The problem is to find a way that each vote (v_i) remains private while at the same time A_q would be in position of understanding what voters, as a whole, believe about A_t, by evaluating the sum of all votes ($\sum_{i=1}^{n} v_i$).*

While research in the direction of the *semi-honest* model has been very active with numerous approaches presented and adopted, this is not the case for the malicious model, which has not been studied extensively.

Semi-Honest Model: In the semi-honest adversarial model, even malicious nodes correctly follow the protocol specification. However, malicious nodes overhear all messages and attempt to use them in order to learn information that otherwise should remain private. Semi-honest adversaries are also called *honest-but-curious.*

Malicious Model: In the malicious model, an adversary, not only can overhear all messages that are exchanged between nodes, but can also compromise the correctness of the protocol. One way of achieving this, is by giving a non valid vote (e.g vote that does not belong to a predefined interval), or by replacing the intermediate computations of other nodes with fake ones, or even by voting multiple times or not voting at all. So, the main aim of this model is not (only) to break the privacy of a protocol, but mainly to make it nonfunctional.

Problem Statement Under the Malicious Model: *The vote v_i of each U_i in $U = \{U_1, \cdots, U_n\}$ must remain private while at the same time, A_q must be in position of verifying that each participant follows the protocol correctly. This means that each U_i must be in position of proving that her vote v_i is valid as well as that they do not try to influence the correctness of the protocol by corrupting data of other nodes.*

All the protocols that are presented in this paper assume that the adversary is *semi-honest*. For the following sections, we assume that each node (A_q, U_i, $i \in [1, n]$) has generated a public/private key pair (k_{A_q}/K_{A_q}, k_{U_i}/K_{U_i}). The private key is kept secret, while the public key is shared with the rest of the nodes. The vote of U_i concerning A_t is denoted by v_i.

For the following sections, we assume that the reader is familiar with the concept of public key cryptography. Let G_q be a group of prime order q, such that computing discrete logarithms in G_q is infeasible. In addition, lets suppose that via an appropriate public procedure, two generators (g, G) of G_q have been selected. Each node (A_q, U_i, $i \in [1, n]$) has generated a private key $K_{U_i} \in_R {}^*_q$ and a public key $k_{U_i} = G^{K_{U_i}}$. The private key is kept secret, while the public key is shared with the rest of the nodes[1]. These public keys will be used to secure communications between the nodes, hence the communication lines between parties are assumed to be secure. All the presented protocols also rely on the use of homomorphic encryption[2] for the collection of votes by the querying agent A_q. The vote of U_i concerning A_t is denoted by v_i.

Definition 1 (Homomorphic Encryption). *Let $E(.)$ be an encryption function. We say that $E(.)$ is additive homomorphic iff for two messages m_1, m_2 the following holds:*

$$E(m_1) \cdot E(m_2) = E(m_1 + m_2).$$

The notation $E(.)$ will refer to the results of the application of an homomorphic encryption function (as described in Definition 1) that A_q can decrypt with her private key.

Apart from that, the protocols that provides defense mechanism under the malicious model rely on secret verifying sharing (VSS) techniques.

Definition 2 (Verifiable Secret Sharing). *As first introduced by B. Chor et al. in [4] a VSS protocol consists of a two stage protocol. Informally, there are n participants, m ($m < n$) of which may be compromised and deviate from the protocol. One of the participants is considered as the dealer who holds a secret value s. In the first stage, the dealer commits to a unique value v and it always holds $v = s$ if the dealer is honest. In the second stage, the already committed value v will be recovered by all good participants, no matter what the compromised participants might do.*

3 Protocols Under the Semi-Honest Model

3.1 Pavlov *et al.* Protocols [6]

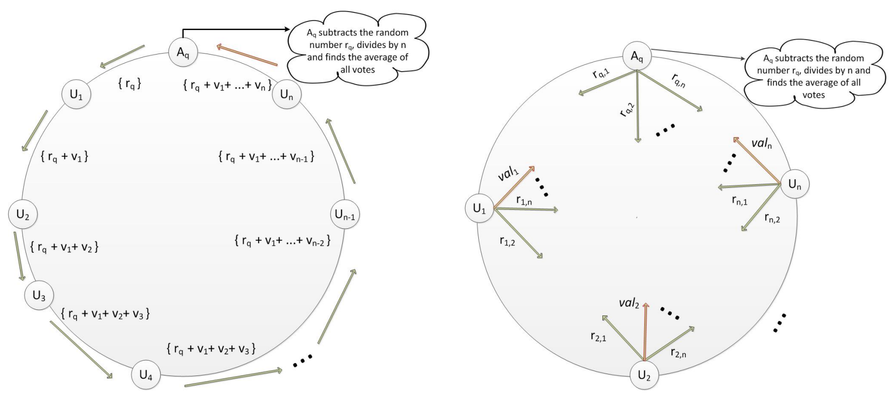
Pavlov *et al.* [6] showed that there are limits on supporting perfect privacy in decentralized reputation systems. More precisely, they showed that when $n - 1$

[1] Key distribution techniques have already been extensively discussed in security literature regarding decentralized environments (e.g P2P and Ad Hoc Networks). Here, the focus is more on the privacy challenges associated with the collection and aggregation of votes.

[2] Pailler's Cryptosystem [3] is an example of cryptosystem where the trapdoor mechanism is based on a homomorphic function.

dishonest peers collude with the querying node to reveal the reputation rating of the remaining honest node then perfect privacy is not feasible. In addition, they suggested a probabilistic scheme for peers selection to ensure that such a scenario will occur with small probability and they proposed three protocols[3] that allow ratings to be privately provided in decentralized additive reputation systems.

Protocol 1 (Figure 1(a)) During the initialization step, A_q creates the set U with all voters, orders them in a circle: $A_q \rightarrow U_1 \rightarrow \cdots \rightarrow U_n$ and sends to each U_i the identity of his successor in the circle. Next, A_q generates a random number r_q such that $r_q \neq 0$ and sends it to the first node in the circle, U_1. Upon reception, U_1 adds his vote v_1 and sends to his successor the sum $r_q + v_1$. Each remaining node in the list follows the same procedure. Finally, the last node will send back to A_q the sum $r_q + \sum_{i=0}^{n} v_i$. Upon reception, A_q will subtract r_q and will divide the remaining number by n. The result will be the average of all votes in the set U.



(a) First Protocol (b) Second Protocol

Fig. 1. Pavlov *et al.* protocols

Protocol 2 (Figure 1(b)) During the initialization step, A_q creates the set U, sends to each $U_i, i \in [1, n]$ the whole list U and generates a random number r_q such that $r_q \neq 0$. Each of the $n + 1$ nodes (including A_q) split their votes (A_q splits $r_q = r_{q,1} + \cdots + r_{q,n}$) into $n + 1$ shares in the following way: U_i chooses n random numbers $r_{i,1}, \cdots, r_{i,n}$ such that $v_i = r_{i,1} + \cdots + r_{i,n}$ and calculates $r_i = r_{q,i} - \sum_{k=1}^{n} r_{i,k}$. He keeps r_i and sends $r_{i,1}, \cdots, r_{i,n}$ to the n other nodes, such that each node U_j receives $r_{i,j}$. At the next step, each U_j calculates $val_j = \sum_{i=1}^{n}(r_{i,j}) + r_j$ and sends val_j to A_q. Upon reception, A_q

[3] The first two protocols corresponds to the semi-honest model, while the third targets the malicious model, thus it is presented in Subsection 4.1.

calculates the sum of n votes $\sum_{i=1}^{n}(val_i) - r_q$, divides by n and finds the average of votes.

3.2 k-Shares Protocol [8]

Hasan *et al.* [8] proposed a privacy preserving reputation protocol under the semi-honest model. The authors were inspired from the second Pavlov protocol and their goal was to reduce the message complexity to $O(n)$. It's main difference from the protocol of Section 3.1 is that each user U_i sends its shares to at most $k < n - 1$ "trustworthy" agents whose behavior in the context of preserving privacy can be "assured" by U_i.

During initialization, A_q sends to each U_i the whole list U. Each U_i selects up to k nodes from U in such a way that the probability that all the selected nodes will collude to break U_i's privacy, is low. Let $A_i = \{U_m, \cdots, U_{m+k}\}$ be the k nodes that were selected by U_i. At this point, U_i prepares $k+1$ shares as follows: The first k shares are random numbers $(r_{i,1}, \cdots, r_{i,n})$ uniformly distributed over a large interval while the last one is selected such that: $v_i = \sum_{j=1}^{n+1} r_{i,j}$. U_i sends to A_q the set A_i and sends $r_{i,j}$ to each U_j , $j \in [m, m+k]$. At this point A_q has also received the A_i sets and can, thus, calculate the list of nodes that each U_i should expect to receive shares from. A_q sends this list to each U_i which in turns proceeds to receive shares from the nodes of the list that A_q provided with. U_i computes the sum of all shares that were received as well as his own share $r_{i,k+1}$. The last step for each voter is to send back to A_q the previous calculated sum σ_i. A_q calculates the sum $\sum_{i=1}^{n} \sigma_i$ and divides it by n in order to find the average of all the votes.

3.3 Dolev *et al.* Protocols

S. Dolev *et al.* [9] proposed four decentralized schemes where the number of messages exchanged is proportional to the number of participants. The first two protocols (AP and WAP protocol) assume that A_q is not compromised while the next two protocols, namely MPKP and MPWP assume that any node that participates in the protocol can act maliciously.

Apart from that, all the proposed schemes are based heavily on a secure homomorphic cryptosystem. More precisely, the AP and WAP protocols are based on the Paillier cryptosystem [3], while MPKP and MPWP are based on the Benaloh cryptosystem [2].

Multiple Private Keys Protocol (MPKP) During initialization, A_q creates two $(1 \times n)$ vectors. The trust vector $TV = [1 \dots 1]$ and the accumulated vector $AV = [1 \dots 1]$. In addition, she creates an accumulated variable σ with initial value equal to 1.

MPKP is divided into two rounds. During the first round each U_i splits his vote v_i in n-shares $(r_{i,1}, \cdots, r_{i,n})$. More precisely, U_i selects his n-shares at random such that $v_i = \sum_{j=1}^{n} r_{i,j}$, encrypts each $r_{i,j}$ with the public key k_j of

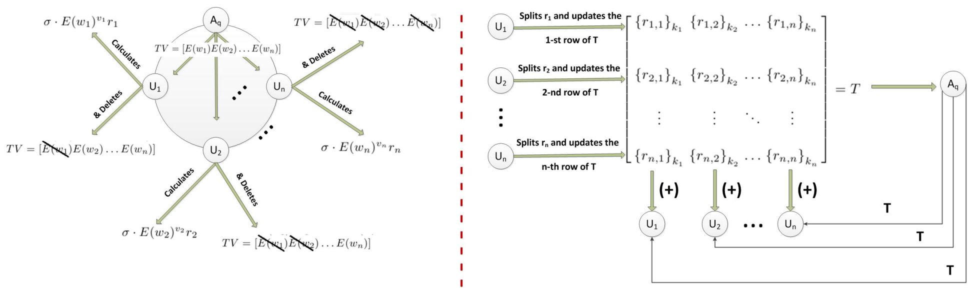


Fig. 2. Basic Steps of MPWP Protocol

user U_j and multiplies it with $AV[j]$. At the end of the first round we will have that $AV = \left[\prod_{k=1}^{n} \{r_{1,k}\}_{k_1} \cdots \prod_{k=1}^{n} \{r_{n,k}\}_{k_n}\right]$.

At this point, the second round begins. Each U_i decrypts $AV[i]$ with his private key K_{U_i}, finds $\sum_{j=1}^{n} r_{j,i}$, encrypts it with the public key k_{A_q} of A_q and adds the encrypted value to σ. Furthermore, he deletes the i-th entry and sends the updated TV vector to the next node in U. At the last step, A_q will receive $\prod_{i=1}^{n} E(\sum_{j=1}^{n} r_{i,j})$ which decrypts it, divides it by n and finds the average of the votes.

Multiple Private Keys Weighted Protocol (MPWP - Figure 2) This is the weighted version of MPKP protocol where the weights w_i correspond to the trust level that A_q has assigned to each U_i, respectively. MPWP computes the weighted average of votes that are given by each individual U_i.

At the initialization stage, A_q creates a $(1 \times n)$ vector $TV = [E(w_1) \dots E(w_n)]$, where $w_i, i \in [1, n]$ is the trust level of U_i. Additionally, A_q initializes a $(n \times n)$ matrix of shares T, where

$$T = \begin{bmatrix} 1 & 1 & \dots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & \dots & 1 \end{bmatrix}$$

and sets the accumulated value $\sigma = 1$. A_q sends to each U_i the TV vector and the matrix T. Upon reception, each U_i generates a random number r_i and calculates $E(w_i)^{v_i} r_i$. Then he adds it to σ by calculating $\sigma = \sigma \cdot E(w_i)^{v_i} r_i$ and deletes the corresponding entry from TV. At this point, U_i shares his random number r_i by replacing the i-th row of T with $S_i = \left[\{r_{i,1}\}_{k_1} \dots \{r_{i,n}\}_{k_n}\right]$. At the end of the first round, A_q receives the updated TV entry that is equal to $\prod_{i=1}^{n} E(w_i)^{v_i} r_i$ and the updated shares matrix T, where

$$T = \begin{bmatrix} \{r_{1,1}\}_{k_1} & \{r_{1,2}\}_{k_2} & \dots & \{r_{1,n}\}_{k_n} \\ \vdots & \vdots & \ddots & \vdots \\ \{r_{n,1}\}_{k_1} & \{r_{n,2}\}_{k_2} & \dots & \{r_{n,n}\}_{k_n} \end{bmatrix}.$$

A_q, by decrypting TV will obtain $\sum_{i=1}^{n} w_i v_i + r_i$.

So, at this point A_q knows the sum of all weighted votes along with the random numbers. This means that she needs to subtract $\sum_{i=1}^{n} r_i$ in order to calculate the average votes. In order to do so, a second round of the protocol begins where each U_i receives T, decrypts $T[][i]$ with K_{U_i} and calculates $\sum_{j=1}^{n} r_{j,i}$. Then he encrypts it with k_{A_q}, adds it to σ and deletes the i-th column from T. After that, A_q will receive $\sigma = \prod_{i=1}^{n} E(\sum_{j=1}^{n} r_{j,i})$, which decrypts with K_{A_q} and finds the sum of all random numbers. Finally, she subtracts the result from TV and finds the weighted average of the votes.

4 Protocols Under the Malicious Model

The main drawback of the protocols described in the previous section is the fact that they are effective only under the not-so-realistic semi-honest model. However, if we wish to impede real malicious behaviors, we have to build protocols that will assume that every adversary acts under the malicious model. It is obvious that in comparison to the semi-honest model, secure protocols within the malicious model enhance security. However it is important to note that a malicious model may provide tighter security, at a greater computational cost. In this section we are presenting four protocols that try to overcome the problem of secure (and private) voting in decentralized systems.

4.1 Pavlov *et al.* Protocol [6]

The goal of this protocol is to ensure that reputation ratings lie within a predefined range. It uses Pederson's [1] verifiable secret sharing scheme to support validity checking of the feedback values provided by voters.

The authors assume that the values of votes v_i are integers in the G_q group of prime order q. In the initialization step, A_q selects a group G_q of a large prime order q with generators g and h, where $log_g h$ is hard to find. Then she sends to each U_i the list U of all nodes along with g and h.

Each U_i creates two polynomials of degree n: $p^i(x) = p^i_0 + p^i_1 x + p^i_0 x^2 + \cdots + p^i_0 x^n$ such that $v_i = p^i_0$ and $q^i(x) = q^i_0 + q^i_1 x + q^i_0 x^2 + \cdots + q^i_0 x^n$ where all coefficients, except p^i_0 are chosen uniformly at random from G_q.

U_i sends to each node U_j, $j \in [1, i) \cup (i, n+1]$ (U_{n+1} node is considered as A_q) $p^i(j)$ and $q^i(j)$. Apart from that, in order to make the above mentioned shares verifiable, U_i also publishes the commitments[4] $C_j = g^{p^i_j} h^{q^i_j}$ of the coefficients. Each U_j upon reception of $p^1(j), p^2(j), \cdots, p^{j-1}(j), p^{j+1}(j), \cdots, p^n(j)$ and $q^1(j), q^2(j), \cdots, q^{j-1}(j), q^{j+1}(j), \cdots, q^n(j)$, calculates $p^j(j), q^j(j)$, $s_m = \sum_{i=1}^{n} p^i(j)$ and $t_m = \sum_{i=1}^{n} q^i(j)$ and sends s_m and t_m to A_q which calculates $s_{n+1} = \sum_{i=1}^{n} p^i(n+1)$ and $t_{n+1} = \sum_{i=1}^{n} q^i(n+1)$.

[4] The main idea of a commitment scheme is that given a commitment $commit(A)$, one has no idea about the exact value of A. Apart from that, based on the discrete logarithm problem it is hard to find $A' : commit(A) = commit(A'), A \neq A'$.

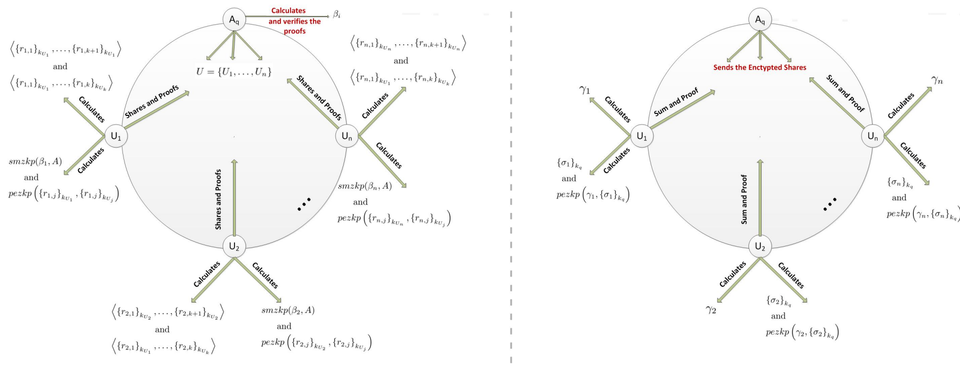


Fig. 3. k-Shares Protocol for the Malicious Model

Upon reception of $s_1, \cdots, s_n$ and $t_1, \cdots, t_n$, A_q obtains $s(0)$, where $s(x) = \sum_{i=1}^{n} p^i(x)$ in the following manner: it computes $\sum_{i=1}^{n+1} s_i L_i(0)$, where $L_i(0)$ is the Lagrange polynomial at 0 and in this case could be expressed by $L_i(0) = \prod_{j=1, j\neq i}^{n+1} \frac{j}{j-i}$.

4.2 k-Shares Protocol for the Malicious Model (Figure 3)

During the initialization step, A_q sends the list U to each U_i. Each U_i selects up to k other nodes in U in such a way that the probability that all of the selected nodes will collude to break U_i's privacy is low. In other words, in this step, each U_i tries to select k trustworthy voters.

Then, U_i creates $k+1$ shares $(r_{i,1}, \ldots, r_{i,k+1})$ such that the first k shares are random numbers uniformly distributed over a large interval. The last share is equal with: $r_{i,k+1} = (v_i - \sum_{j=1}^{k} r_{i,j}) \ mod \ M$ where M is a publicly known modulus. After the calculation of the shares, U_i encrypts the $k+1$ shares with her public key and obtains $\left\langle \{r_{i,1}\}_{k_{U_i}}, \ldots, \{r_{i,k+1}\}_{k_{U_i}} \right\rangle$. Then she also encrypts the first k shares with the public key of the corresponding node to obtain $\left\langle \{r_{i,1}\}_{k_{U_1}}, \ldots, \{r_{i,k}\}_{k_{U_k}} \right\rangle$.

At this point, U_i is responsible for the generation of two non-interactive zero knowledge proofs in order for the shares as well as the vote itself to be tested for their validity. So, U_i computes:

$$\beta_i = \left(\{r_{i,1}\}_{k_{U_i}} \times \ldots \times \{r_{i,k+1}\}_{k_{U_i}} \right)$$

$$\overset{homomorphic}{\Longleftrightarrow} \beta_i = \left\{ \sum_{j=1}^{k+1} r_{i,j} \right\}_{k_{U_i}}$$

and creates a zero knowledge proof of set membership[5] $smzkp(\beta_i, A)$ where A is the interval in which a valid vote should belong to. U_i then generates k non-interactive plaintext equality zero zero knowledge proofs[6]. Each proof contains $\{r_{i,j}\}\, k_{U_i}$ and $\{r_{i,j}\}_{k_{U_j}}$ is denoted by $pezkp\left(\{r_{i,j}\}_{k_{U_i}}, \{r_{i,j}\}_{k_{U_j}}\right)$, where $j \in [1, k]$ and verifies that the two cipehertexts encrypt the same plaintext. With this way, a verifier can be sure that the shares she received are correct.

U_i sends $\left\langle \{r_{i,1}\}_{k_{U_i}}, \ldots, \{r_{i,k+1}\}_{k_{U_i}} \right\rangle$ and $\left\langle \{r_{i,1}\}_{k_{U_1}}, \ldots, \{r_{i,k}\}_{k_{U_k}} \right\rangle$ as well as $smzkp(\beta_i, A)$ and $pezkp\left(\{r_{i,j}\}_{k_{U_i}}, \{r_{i,j}\}_{k_{U_j}}\right)$ to A_q.

Upon reception, A_q calculates β_i on her own and verifies the proofs received from U_i. If the verification is correct, she sends the encrypted shares she received to the corresponding nodes in U. Each U_i that received the encrypted shares from A_q calculates $\gamma_i = \left\{\sum_{j=1}^{k+1} r_{i,j}\right\}_{k_{U_i}}$ by using the additive homomorphic property and with her private key decrypts γ_i and finds the sum of all received shares $\sigma_i = \sum_{j=1}^{k+1} r_{i,j}$. Then, U_i encrypts σ_i with k_q, creates a non-interactive plaintext equality zero zero knowledge proof $pezkp\left(\gamma_i, \{\sigma_i\}_{k_q}\right)$ and sends them to A_q. A_q first computes γ_i and then verifies the zero knowledge proof. In the case where the verification of the proofs are correct, which means that A_q has received the shares correctly and she has also calculated γ_i correctly, A_q decrypts each $\{\sigma_i\}_{k_q}$ and finds the sum of all votes by computing the following $\sum_{i=1}^{k} \sigma_i$.

4.3 Dolev *et al.* Protocols for Malicious Adversaries [10]

S. Dolev *et al.* presented two decentralized protocols, namely *PKEBP* and *CEBP*, that provides partial resistant against malicious users by the mean that A_q can check the validity of votes.

Public Key Encryption Based Protocol (PKEBP - Figure 4): During initialization, A_q creates a $(1 \times n)$ vector and initializes it $TV = [1 \ldots n]$. PKEBP is divided into two rounds. During the first round, A_q sends TV to all nodes in U. Upon reception, each U_i encrypts her vote with the public key of A_q, sets $TV[i] = \{v_i\}_{k_{A_q}}$ and sends the updated vector to U_{i+1}. The result of the first round is a new vector with a sequence of encrypted elements: $TV' = \left[\{v_1\}_{k_{A_q}} \ldots \{v_n\}_{k_{A_q}}\right]$.

The second round of PKEBP is performed when TV' returns from U_n to U_1 (A_q is bypassed in this round). Each U_i performs a random permutation π of her $i-th$ entry with another entry from the vector and sends the updated vector

[5] A zero knowledge proof of set membership denoted as $smzkp(E(m_i), A)$, shows that an ecryption of a message m_i encrypts an element/message from set $A := \{m_1, \ldots, m_p\}$.

[6] Let $E_1(m)$ and $E_2(m)$ be encryptions of a message m with two different public keys. A zero knowledge proof of plaintext, allows a prover to convince a verifier that $D_1(E_1(m)) = m = D_2(E_2(m))$, where $D(.)$ is a decryption function.

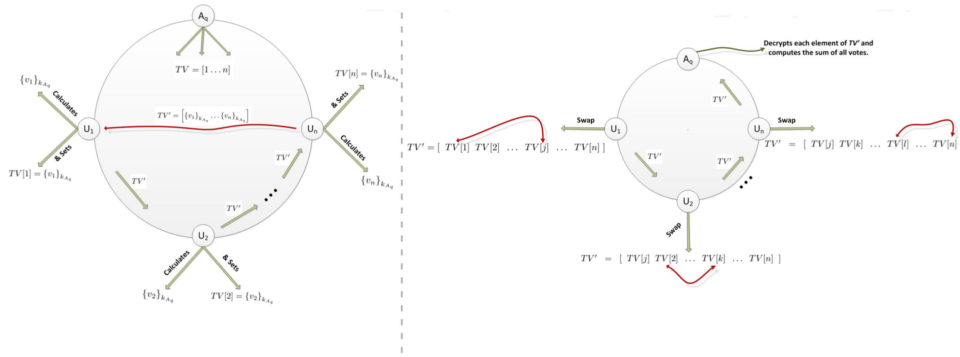


Fig. 4. Public Key Encryption Based Protocol

to U_{i+1}. At the end of round two, U_n sends to A_q the final vector. A_q decrypts each element with her private key and computes the sum of all votes.

Commutative Encryption Based Protocol (CEBP): CEBP is based on commutative encryption. During initialization, A_q creates a $(1 \times n)$ vector and initializes it $TV = [1 \ldots n]$. CEBP is divided into three rounds. During the first round, A_q sends TV to all nodes in U. Upon reception, each U_i encrypts her vote with her public key and with the public key of A_q. So, U_i sets $TV[i] = \left\{\{v_i\}_{k_{U_i}}\right\}_{k_{A_q}}$ and sends the updated vector TV' to U_{i+1}.

In the second round, each U_i encrypts all entries of TV', except the $i-th$ that she had encrypted in the previous round, and sends the updated vector to U_{i+1}. This means that at the end of the second round each entry of the vector will be encrypted with the following keys: $k_{U_1}, k_{U_2}, \ldots, k_{U_n}, k_{A_q}$.

During the third round, each U_i randomly permutes the $i-th$ entry of the vector and decrypts all the entries with K_{U_i}. So, at the end of this round, A_q will receive a vector that contains all the individual votes, encrypted with k_{A_q} and in a random order. Upon reception, A_q decrypts each value and finds the sum of the n votes.

5 Vulnerabilities/Inefficiencies of Reviewed Systems

In this section we describe and categorize the various types of attacks that aim to break the privacy of the above mentioned schemes. All the attacks that are presented in the Section 5.1 assume that the adversary is *semi-honest*. Then, in Section 5.2, we provide a comparison between the protocols that preserve privacy under the malicious model and we expose their inefficiencies.

5.1 Attacks

In all the cases, we assume that A_q is malicious and can overhear every message that is exchanged between voters. If we do not make this assumption, the problem of trust aggregation has a trivial solution.

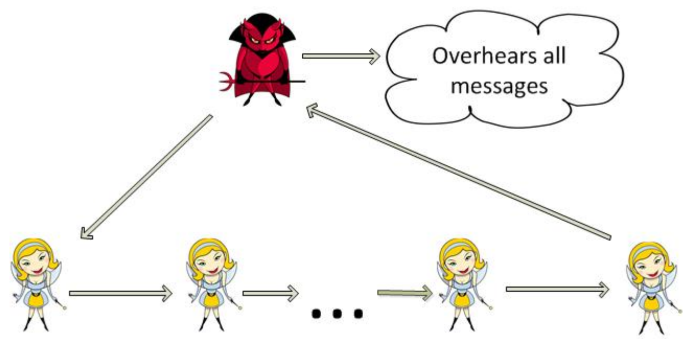


Fig. 5. Querying Node Attack

1. **Querying Node Attack (Figure 5):** In this attack, the only malicious node is A_q, which can overhear all messages that are sent between voters.

   Affected Protocols: *Pavlov Protocols 1, 2 and 3, k-shares protocol, Dolev protocols AP and WAP.*
   - **Querying Node Attack at Pavlov Protocol 1:** A_q has generated a random number r_q at the beginning of the protocol and voters are adding their votes to that number one by one. This means that A_q can find each individual vote by overhearing every message, since she knows r_q.
   - **Querying Node Attack at Pavlov Protocols 2, 3 & k-Shares Protocol:** The random numbers that each node generates do not really offer any protection from A_q or from any other curious adversary who overhears the channel. This is because the parts of the random numbers that are exchanged among the nodes are not encrypted in any way.
   - **Querying Node Attack at AP & WP:** Since all messages are encrypted with k_{A_q} and the voters do not use random numbers, A_q can still decrypt each message one by one in order to find the individual votes for every U_i, $i \in [1, n]$.
2. **Alone in the List Attack (Figure 6):** If A_q is malicious she can ask each node from U to give their vote *separately*. By doing so, she will be able to find the value of all individual votes and thus easily break their privacy.

   Affected Protocols: *All protocols*
   - **Analysis:** Normally, A_q receives a sum of votes and that is the reason why she cannot understand the exact vote of each U_i. In the case where

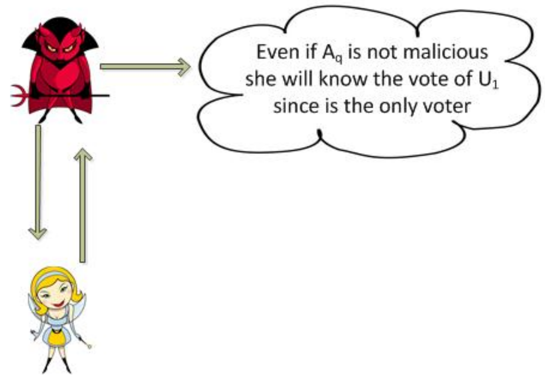


Fig. 6. Alone in the List Attack

A_q asks each U_i to vote individually (size of U is equal to 1), she receives one vote at a time. Thus she knows the vote of each voter.

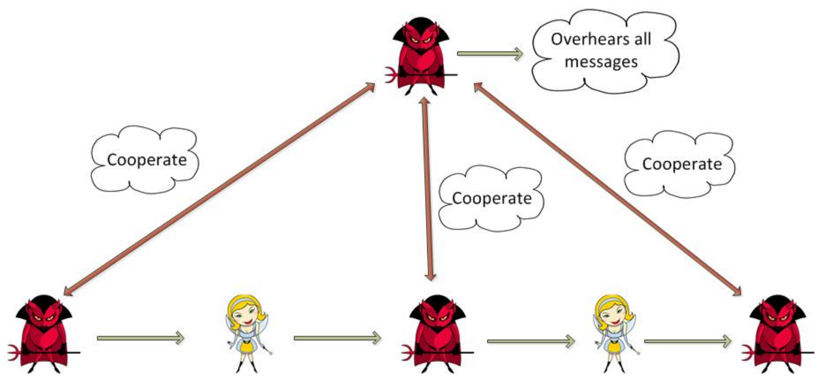


Fig. 7. Sandwich Attack

3. **Sandwich Attack (Figure 7):** In this scenario, A_q is considered as malicious and *arranges* the nodes in U in such a way that all U_{2k+1} or U_{2k}, $k \in \mathbb{N}$ nodes are malicious. By doing so, A_q can use values from adjacent malicious nodes to calculate the random number of the legitimate node situated between them, thus finding all the individual votes in the set. This attack is effective on protocols where each node is sending either a random number that she has generated either a share of her vote to the next node in U.

   Affected Protocols: *Pavlov Protocols 1 2 and 3, k-Shares protocol, AP, WAP.*

   - **Sandwich Attack at Pavlov Protocol 1:** Even if A_q could not overhear all messages, he could cooperate with every malicious voter in order to find the votes of the rest nodes. More precisely, each malicious user would inform A_q about his vote as well as the sum that he received from the previous node. Upon reception, A_q would subtract the vote of the malicious node and the random number r_q that he generated at the initialization step. The result would be the vote of the previous node.

- **Sandwich Attack at Pavlov Protocols 2, k-Shares protocol:** As we mentioned before, the random numbers are not encrypted with any key which means that the whole information is known to everyone who overhears the channel. The cooperation between malicious voters and A_q is not essential, since A_q can find the votes on his own.
- **Sandwich Attack at AP & WP:** In both cases, the sum of votes is encrypted with the public key of A_q and each U_i adds his vote to the previous one, by using the homomorphic property of the underlying cryptosystem. Even though votes are encrypted this time, the encryption does not offer any kind of protection if A_q is adversarial. Also in this case, the cooperation between malicious voters and A_q is not essential, since A_q can find the votes on her own.
- **Sandwich Attack at PKEBP:** The minimum requirements in order for a sandwich attack to be effective in PKEBP protocol is when A_q and at least one node from $U_c = \{U_1, U_2, U_{n-1}, U_n\}$ (preferably U_n) collude.
  - **A_q colludes with U_n:** At the end of the first round U_n receives TV that contains all the individual votes encrypted with k_{A_q}. U_n sends TV to A_q and she decrypts one by one each element of TV in order to find all the individual votes. This means that even before the end of the second round, A_q will have totally break the privacy of the protocol.
  - **A_q colludes with U_1:** At the beginning of the second round, U_1, receives TV that contains all the individual votes encrypted with k_{A_q}. U_n sends TV to A_q and she decrypts one by one each element of TV in order to find all the individual votes.
  - **A_q colludes with U_2 or U_{n-1}:** Lets suppose that A_q colludes with U_{n-1} (the case for U_2 is identical). Before the end of first round A_q will receive TV from U_{n-1}. This means that she will effectively compute the first $n-1$ votes. At the end of first round where A_q will receive the final vector, she will find the missing vote.
- **MPKP & MPWP *Resistance* to Sandwich Attack:** We assume that A_q and U_{2k+1}, $k \in \mathbb{N}$ are malicious (U_1, U_3, U_5, etc). After the first round, malicious nodes will be aware of v_{2k+1}, $r_{2k+1,i}$, $r_{2k,2k+1}$, $k \in \mathbb{N}$, $i \in [1, n]$ values. At the end of the second round, A_q will be aware of the following:
  a) $\sum_{i=1}^{n} v_i$. Since she also knows each v_{2k+1} she can easily calculate $\sum_{i=1}^{n/2} v_{2i}$.
  b) $\sum_{i=1}^{n} r_{i,1}, \cdots, \sum_{i=1}^{n} r_{i,n}$. Since every node adds $\sum_{j=1}^{n} r_{j,i}$ to $E(.)$, A_q can find each individual sum.

Table 1 shows a list of what A_q knows at the end of the protocol and what information she is missing. By using these values, A_q cannot find the individual votes from the legitimate voters. The only thing she can do is to approximately calculate the values since she knows that each vote v_i is bounded from α and β. This is a legitimate assumption since Dolev et al. made the additional requirement that the homomorphic modulus, m must be identical for *all* users. This is possible under the Benaloh cryptosystem [2], however, decryption can only be performed by trying all possible values and finding the unique value that decrypts correctly. Furthermore, a (degenerate version of a) sandwich attack can be successfully lunched only in the case where $n-1$ nodes are compromised (as Dolev *et al.* mention in their paper).

Table 1. Information that A_q has gained at the end of the second round

v_1	$r_{1,1}$	$r_{1,2}$	$r_{1,3}$	$r_{1,4}$	$\cdots$	$r_{1,n}$
	$r_{2,1}$		$r_{2,3}$		$\cdots$	$r_{2,n}$
v_3	$r_{3,1}$	$r_{3,2}$	$r_{3,3}$	$r_{3,4}$	$\cdots$	$r_{3,n}$
	$r_{4,1}$		$r_{4,3}$		$\cdots$	$r_{4,n}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\cdots$	$\vdots$
v_n	$r_{n,1}$	$r_{n,2}$	$r_{n,3}$	$r_{n,4}$	$\cdots$	$r_{n,n}$

The weaknesses of the described protocols are summarized in Table 2.

5.2 Inefficiencies of Protocols Under the Malicious Model

In this subsection, we make a brief comparison between the protocols presented in Section 4.

- The main disadvantage of Pavlov's protocol is the communication complexity. More precisely, it requires $O(n^3)$ messages to be exchanged between nodes that take part in the voting procedure. In addition, there is an insufficient description of the protocol. For example, there is no explanation regarding the zero-knowledge proofs that the protocol requires. Also, it is not clear at all if a vote can belong to any interval $[a, b]$ or should be bounded to a smaller one (e.g $[-1, 1]$). This would change the required computations for the verifier of a vote. As a result, and taking into consideration the poor explanation of crucial parts of the protocol, it is not clear whether it is open to mistakes or not.

Table 2. Protocols Summary – Resistance to Attacks

	Querying	**Alone in the List**	**Sandwich**
Pavlov 1	NO	NO	NO
Pavlov 2	NO	NO	NO
Pavlov 3	**YES**	NO	**YES**
k **Shares**	NO	NO	NO
AP	NO	NO	NO
WAP	NO	NO	NO
MPKP	**YES**	NO	**YES**
k **Shares Malicious**	**YES**	NO	**YES**
CEBP	**YES**	NO	**YES**
PKEBP	**YES**	NO	NO
CEBP	**YES**	NO	**YES**

- k-Shares protocol even though it works with a lower complexity than Pavlov's protocol, it has two basic drawbacks. First, the query agent A_q acts like a central authority since all messages are transfered to her and then she forwards them to the actual receivers. Second, in order for every node to be able to validate the shares as well as the submitted votes, the protocol makes use of non-interactive zero knowledge proofs. More precisely, $O(n)$ non-interactive zero knowledge proofs of set membership and $O(n)$ non-interactive zero knowledge proofs of plaintext equality are required. The use of such techniques, guarantees security (in the sense that the submitted data are valid) but with a higher computational cost, which is not captured in the description of the protocol.
- In CEPB protocol, A_q can validate the submitted vote very easily since at the last round, she receives a list with all the individual votes in a random order. With this way, on one hand authors manage to avoid the complex computations of zero knowledge proofs but on the other their protocol is using commutative encryption schemes, like the Pohlig-Hellman scheme [11] which is based on the assumption of the intractability of the discrete logarithm problem. However, not only Pohlig-Hellman, but the existing commutative encryption schemes in general does not provide formal methods of security [12], and may lead to security breaches in real world applications. More precisely, in [13] it is

shown that Polhig-Hellman encryption scheme, preserves certain attributes of the plaintext. As a result, by matching the characteristics of the plaintext and the ciphertext, the original value of set of encrypted values can be identified.

6 Conclusions

In this invited paper, we have presented a series of protocols aiming to provide privacy between individual voters in an additive reputation system. We have analyzed these protocols in order to see how they react when honest-but-curious nodes try to break the privacy and find the individual votes of other nodes. To this end, we have provided a description of malicious behaviors/attacks against these protocols by utilizing three different attack scenarios. Additionally, we showed that none of the existing protocols can build defensive mechanisms that provide resistance against all possible attacks. More precisely, *all* protocols are vulnerable to an "*alone in the list*" attack which may be the most difficult attack to handle.

We are currently working on the design of a decentralized privacy preserving scheme that will provide effective defense mechanisms against the type of attacks described above.

References

1. T. Pederson, *Non-interactive and information secure veriable secret sharing.* In Advances in Cryptology - Crypto 91, pp.129-140, 1991
2. J. Benaloh, *Dense Probabilistic Encryption.* In Proceedings of the Workshop on Selected Areas of Cryptography, pp.120–128, 1994.
3. P. Paillier, *Public-Key Cryptosystems Based on Composite Degree Residuosity Classes.* In Advances in Cryptology EUROCRYPT '99, pp.223–238, Springer Berlin Heidelberg, 1999.
4. B. Chor, S. Goldwasser, S. Micali and B. Awerbuch, *Verifiable secret sharing and achieving simultaneity in the presence of faults.* In 26th IEEE Symposium on Foundations of Computer Science, pp.383–395, 1985.
5. P. Resnick and R. Zeckhauser. *Trust among strangers in Internet transactions: Empirical analysis of eBay's reputation system.* In The Economics of the Internet and E-Commerce, 2002.
6. E. Pavlov and J. S. Rosenschein and Z. Topol. *Supporting Privacy in Decentralized Additive Reputation*, In Second International Conference on Trust Management (iTrust 2004), 2004.
7. Y. Lindel and B. Pinkas. *Secure Multiparty Computation for Privacy-Preserving Data Mining*, Journal of Privacy and Confidentiality: Vol. 1: Iss. 1, Article 5, 2009.
8. O. Hasan and L. Brunie and E. Bertino. *k-Shares: A Privacy Preserving Reputation Protocol for Decentralized Environments*, In the 25th IFIP International Information Security Conference (SEC 2010), pp.253–264, 2010.
9. S. Dolev and N. Gilboa and M. Kopeetsky. *Computing multi-party trust privately: in $O(n)$ time units sending one (possibly large) message at a time.* In Proceedings of the 2010 ACM Symposium on Applied Computing (SAC '10), pp.1460–1465. ACM, New York, NY, USA, 2010.

10. S. Dolev, N. Gilboa, M. Kopeetsky. *Computing Trust Anonymously in the Presence of Curious Users*. In Proceedings of the International Symposium on Stochastic Models in Reliability Engineering, Life Science and Operations Management, Sami Shamoon College of Engineering, Beer Sheva, Israel, February, 2010.
11. S. C. Pohlig, M. E. Hellman. *An improved algorithm for computing logarithms over GF(p) and its cryptographic significance*. In IEEE Transactions on Information Theory, Vol. 24, No. 1. (1978), pp. 106-110.
12. S. A. Weis. *New Foundations for Efficient Authentication, Commutative Cryptography, and Private Disjointness Testing*. PhD thesis, Massachusetts Institute of Technology, 2006.
13. Y. Zhang, W. K. Wong, S. M. Yiu, N. Mamoulis, and D. W. Cheung, *Lightweight Privacy-Preserving Peer-to-Peer Data Integration*. Technical Report TR-2011-12.