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

Antonis Michalas[1], 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 against such systems. 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 [5]. 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 [4] 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 [5].

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 Sections 3, 4 and 5 we present the details of the most important protocols that allow ratings to be (partially) private in decentralized additive reputation systems. In Section 6, we present attacks that can break the privacy of the presented protocols and in Section 7 we conclude the paper.

2 Problem Statement & Definitions

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.*

In trust management and reputation schemes nodes in a network have in general two ways of collecting information about other nodes. They use either first-hand information (Direct Trust) or second-hand information (Third-Party

Trust) in order to evaluate other nodes. A trust establishment framework evaluates neighboring nodes based on direct observations while trust relations between two nodes with no prior direct interactions are built through a combination of opinions from intermediate nodes. In this work we are concerned with the following problem:

Problem Statement: *A querying node A_q, receives a 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 a set with all nodes that A_q asks to vote. 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 a sum of all votes ($\sum_{i=1}^{n} v_i$).*

All the protocols that are presented in this paper assume that the adversary is *semi-honest* (for a definition see Section 6). 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.

3 Pavlov *et al.* Schemes [5]

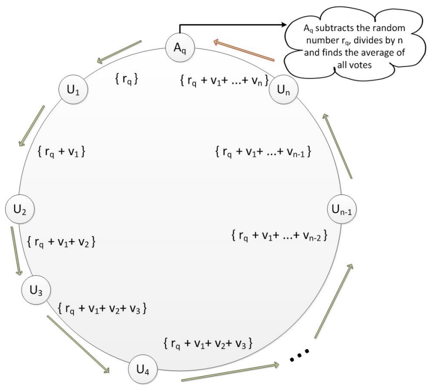
Pavlov *et al.* [5] showed that there are limits on supporting perfect privacy in decentralized reputation systems. More precisely, they showed that when $n-1$ dishonest peers collude with the querying node to reveal the reputation rating of the remaining honest node then perfect privacy is not feasible. 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 that allow ratings to be privately provided in decentralized additive reputation systems.

3.1 Protocol 1 (Figure 1(a))

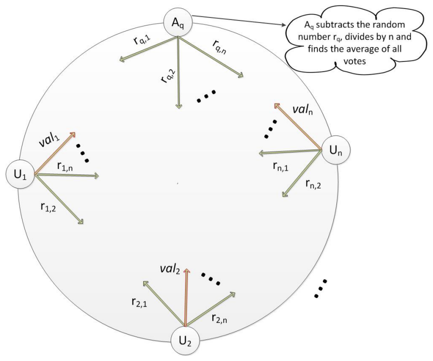
During the initialization step, A_q creates the set U with all voters, orders them in a circle: $A_q \rightarrow U_1 \rightarrow \cdots \rightarrow U_n$ and sends to each U_i the identity of his successor in the circle. Next, A_q generates a random number r_q such that $r_q \neq 0$ and sends it to the 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.

3.2 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


A$_q$ subtracts the random number r$_q$, divides by n and finds the average of all votes
A$_q$
U$_1$
U$_2$
U$_3$
U$_4$
U$_{n-1}$
U$_n$
{ r$_q$ }
{ r$_q$ + v$_1$+ …+ v$_n$ }
{ r$_q$ + v$_1$ }
{ r$_q$ + v$_1$+ …+ v$_{n-1}$ }
{ r$_q$ + v$_1$+ v$_2$ }
{ r$_q$ + v$_1$+ …+ v$_{n-2}$ }
{ r$_q$ + v$_1$+ v$_2$+ v$_3$ }
{ r$_q$ + v$_1$+ v$_2$+ v$_3$ }

(a) First Protocol



(b) Second Protocol

Fig. 1. Pavlov *et al.* protocols

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

3.3 Protocol 3

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: $p^i(x) = p_0^i + p_1^i x + p_0^i x^2 + \cdots + p_0^i x^n$ such that $v_i = p_0^i$ and $q^i(x) = q_0^i + q_1^i x + q_0^i x^2 + \cdots + q_0^i x^n$ where all coefficients, except p_0^i 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 $q^i(j)$ along with the commitments of his polynomials of the form: $g^{p_0^i} h^{q_0^i}, \cdots, g^{p_n^i} h^{q_n^i}$. 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 $t_{n+1} = \sum_{i=1}^{n} q^i(n+1)$.

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 k-Shares Protocol [7]

Hasan *et al.* [7] 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 3.2 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.

5 Dolev *et al.* Protocols

Dolev *et al.* [8] 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].

5.1 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.

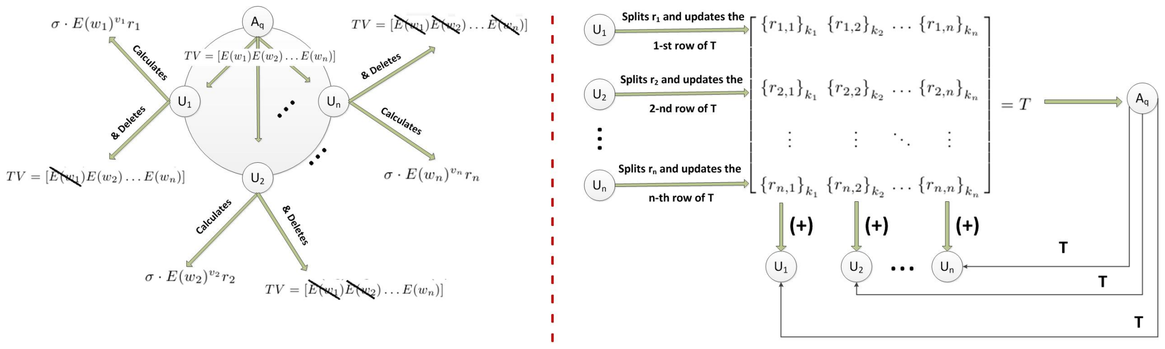

$\sigma \cdot E(w_1)^{v_1} r_1$
Calculates
& Deletes
$TV = [\cancel{E(w_1)} E(w_2) \ldots E(w_n)]$
A_q
$TV = [E(w_1)E(w_2) \ldots E(w_n)]$
U_1
U_2
U_n
$TV = [\cancel{E(w_1)}\cancel{E(w_2)} \ldots \cancel{E(w_n)}]$
& Deletes
Calculates
$\sigma \cdot E(w_n)^{v_n} r_n$
Calculates
& Deletes
$\sigma \cdot E(w_2)^{v_2} r_2$
$TV = [\cancel{E(w_1)}\cancel{E(w_2)} \ldots E(w_n)]$
Splits r_1 and updates the 1-st row of T
Splits r_2 and updates the 2-nd row of T
Splits r_n and updates the n-th row of T
$\{r_{1,1}\}_{k_1}$ $\{r_{1,2}\}_{k_2}$ $\cdots$ $\{r_{1,n}\}_{k_n}$
$\{r_{2,1}\}_{k_1}$ $\{r_{2,2}\}_{k_2}$ $\cdots$ $\{r_{2,n}\}_{k_n}$
$\{r_{n,1}\}_{k_1}$ $\{r_{n,2}\}_{k_2}$ $\cdots$ $\{r_{n,n}\}_{k_n}$
$= T$
(+)
(+)
(+)
T
T
T

Fig. 2. Basic Steps of MPWP Protocol

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 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.

5.2 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 $TV = [E(w_1) \ldots E(w_n)]$, 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

$$T = \begin{bmatrix} 1 \, 1 \ldots 1 \\ \vdots \, \vdots \, \ddots \, \vdots \\ 1 \, 1 \ldots 1 \end{bmatrix}$$

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 $S_i = \left[\{r_{i,1}\}_{k_1} \ldots \{r_{i,n}\}_{k_n}\right]$. 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.

6 Threat Model and Attacks

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 protocols that are presented in this paper assume that the adversary is *semi-honest*. A definition for the semi-honest model follows:

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.*

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.

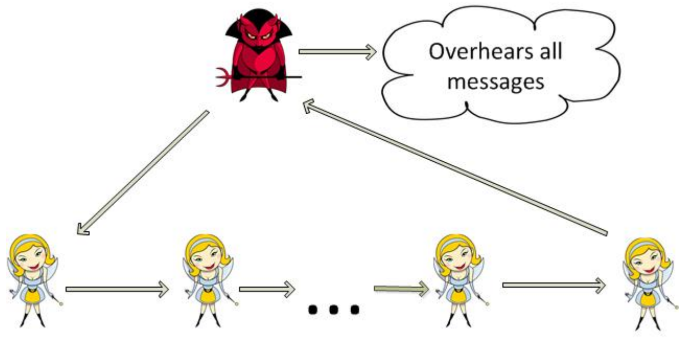


Fig. 3. Querying Node Attack

1. **Querying Node Attack (Figure 3):** 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, $i \in [1, n]$.

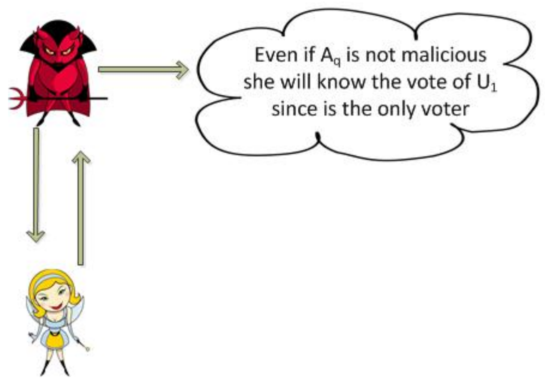

Even if A_q is not malicious she will know the vote of U_1 since is the only voter

Fig. 4. Alone in the List Attack

2. **Alone in the List Attack (Figure 4):** 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 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.

3. **Sandwich Attack (Figure 5):** 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}, $k \in \mathbb{N}$

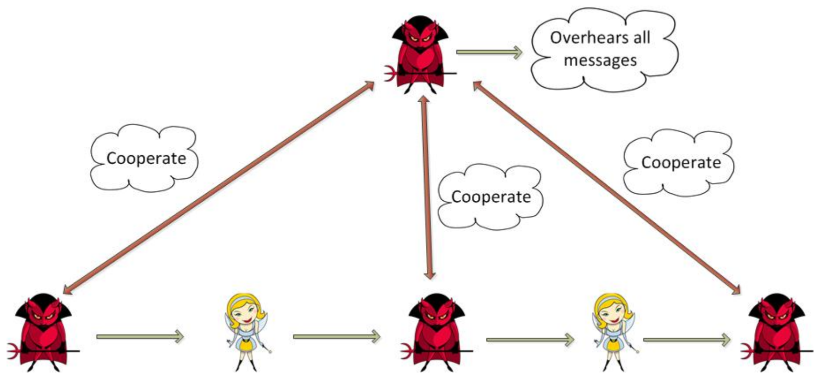


Fig. 5. Sandwich Attack

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 2,3, 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.
- **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.

7 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

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	NO	NO	NO
k **Shares**	NO	NO	NO
AP	NO	NO	NO
WAP	NO	NO	NO
MPKP	**YES**	NO	**YES**
MPWP	**YES**	NO	**YES**

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 based on homomorphic encryption 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. 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.
5. 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.
6. 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.

7. 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.
8. 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.