# A Distributed Privacy-Preserving Reputation Protocol Resistant to Reflection Attacks

Quentin Santos (quentin.santos@ens-lyon.fr)*
Omar Hasan (omar.hasan@insa-lyon.fr)†
Lionel Brunie (lionel.brunie@insa-lyon.fr) †

**Abstract:** We are interested in building a protocol for inferring reputation from a democratic process, like Amazon.com or Ebay.com, but without relying on a centralized node and while preserving privacy. We know from [Yao86] and [GMW87] that such protocols are theoretically possible. Pavlov et al. published a protocol for this problem in [PRT04] yielding complexity $O(n^3 + \log N)$ w.r.t. exchanged messages. Hasan et al. proposed a more efficient protocol in [HBB11] and [HBBS13] using only $O(n + \log N)$ messages. We found that the protocol is vulnerable to a reflection attack when the querying node $q$ is malicious. However, we show that a simple solution makes the protocol resistant to such an attack.

## 1 Introduction

Secure Multi-Party Computation (SMPC) is a field that was introduced by Yao in 1982 [Yao82]. Basically, we consider $n$ nodes with individual private data $(x_i)$ and we are interested in retrieving the result of some computation $f(x_i)_i$ on these data while avoiding sharing information on private data. Through the exchanges, the only thing a node can learn is the result of the computation.

It is common to split the study of such a problem into two steps using two attack models:

- passive ("semi-honest"): the attacker learns the private data and exchange messages of some nodes
- active ("malicious"): the attacker can fully control some nodes to make them abort or tamper data

In the case of reputation, a node $q$ (querier) wishes to know the reputation of $t$ (target). The reputation of $t$ is given as $\langle f_{s,t} \rangle_s$ where $f_{s,t}$ is the trust factor of node $s$ (source) for node $t$. In other words, reputation is the mean trust given to $t$.

Yao [Yao86] first formalized an approach to build a secure privacy-preserving protocol when the number of nodes is 2. Goldreich et al. [GMW87] then found a way to build a protocol for evaluating any function $f$ for an arbitrary number of nodes assuming honest majority (strictly more than half the nodes must be honest; no protocol is possible otherwise).

* École Normale Supérieure de Lyon, France
† University of Lyon, CNRS, INSA Lyon, LIRIS, UMR5205, F-69621, France

However, using this generic approach for the specific problem of retrieving the reputation yields an overwhelming complexity with regards to the number of transmitted messages. The protocol from [PRT04] decreases the complexity down to $O(n^3 + \log N)$ where $n$ is the number of nodes which give their appreciation for $t$, and $N$ is the total number of nodes in the network. Hasan et al. proposed a more efficient approach in [HBB11] and [HBBS13], that further improves the efficiency of the protocol and yields a complexity of $O(n + \log N)$ exchanged messages.

We first expose the internals of the protocol designed by Hasan et al. for the passive attack model. After this, we introduce some required cryptographic material before detailing the protocol for the active attack model. Then, we describe a reflection attack that allows the querier to compromise the privacy of the protocol. Finally, we propose a way to prevent $q$ from performing such an attack.

## 2 Passive attack model

The basic idea of the reputation protocol is that each node $s$ will split its personal data $f_{s,t}$ into parts that bears no information about the original vote when considered individually. Each of these parts will be summed with parts from other nodes before they are reunited. The main steps of the protocol are depicted below (we skip the choosing of trustworthy nodes for the sake of simplicity; refer to original papers for details):

1. $f_{s,t}$ is split as $f_{s,t} = \sum_{1 \leq i \leq k+1} x_{s,i}$
2. $k$ parts $(x_{s,i})_{1 \leq i \leq k}$ are sent and one is kept locally
3. local and received parts are summed $\sigma_s = x_{s,k+1} + \sum x_{\_,s}$
4. $\sigma_s$ is sent to $q$
5. $q$ computes $\sum_s \sigma_s = \sum_s \sum_{1 \leq i \leq k+1} x_{s,i} = \sum_s f_{s,t}$

Step 5 shows how this protocol allows $q$ to retrieve the final reputation. Some details must be given about step 1 to understand why the privacy is preserved: $k$ parts are picked uniformly at random in a big interval $[0, K]$ and the last one is chosen so that the equation is verified.

On the network level, there are two main visible steps: first, source nodes exchange messages between themselves (sharing parts), then the sums are sent to the querier.

## 3 Cryptographic primitives

We now want to avoid any malicious node from tampering data. For this purpose, we will use Paillier's asymetric encryption scheme [Pai99]. The first important property of the scheme is homomorphism: it is an encryption scheme $E$ so that for any key $k$ and plaintext values $a$ and $b$, $E_k(a) \times E_k(b) = E_k(a + b)$ .

Additionally, we need to prove properties on encrypted data. This is done by using Zero-Knowledge Proofs (ZKP). Set Membership ZKPs (SM-ZKP) ensures that the en-

crypted value belongs to a chosen set; Plaintext Equality ZKPs (PE-ZKP), ensures that two messages encrypted with different keys contains the same data.

In Paillier's scheme, these ZKPs can be non-interactive, meaning that the proof can be viewed as an opaque blob of information easily transmitted. An interactive ZKP would need the verifier to choose a challenge and send it to the prover, and the prover to respond to this challenge.

**Note** This scheme works modulo $M$. Thus, in the rest of this document, equality over encrypted values should be interpreted modulo $M$. In case of set-membership, the actual test is done on $h_s M + [0, F]$ instead of $[0, F]$ where $h_s$ is some integer to be transmitted along with the ZKP.

# 4 Active attack model

When using the cryptosystem defined in Section 3, we need a public/secret key pair. We assume that each node $s$ has generated such a pair and made its public part available to others nodes. We will write $E_s$ to refer to the encryption using the public key of node $s$. Decryption (and re-encryption) will implicitly use the secret key; decrypting a message $E_s(\_)$ can only be done at node $s$.

We extend the previous protocol to work in the active attack model. In the following listings, step $n.(a)$ directly maps to the original step $n$. The other sub-steps are the requisite additional operations needed to ensure the security of the protocol.

1. (a) $f_{s,t}$ is split as $f_{s,t} = \sum_{1 \leq i \leq k+1} x_{s,i}$
   (b) each $x_{s,i}$ is encrypted for both $s$ and $u_i$
   (c) a PE-ZKP is generated for each $(E_s(x_{s,i}), E_{u_i}(x_{s,i}))$.
   (d) a SM-ZKP is generated for $E_s(f_{s,t})$ on $[0, F]$
2. (a) $E_s(x_{s,i})$, $E_{u_i}(x_{s,i})$ and ZKPs are sent to $q$
   (b) $q$ transmits $E_{u_i}(x_{s,i})$ to node $u_i$
3. (a) local and received parts are summed $E_s(\sigma_s) = \prod E_s(x_{\_,\_})$
   (b) $q$ checks all ZKPs
4. (a) $\sigma_s$ is re-encrypted for $q$
   (b) a PE-ZKP is generated for $(E_s(\sigma_s), E_q(\sigma_s))$
   (c) $E_q(\sigma_s)$ and ZKP are sent to $q$
5. (a) $q$ computes $\sum_s \sigma_s$
   (b) $q$ check the new PE-ZKPs

**Note:** From this, we can observe that $q$ acts as a central node for the given instance of the protocol and route messages between source nodes. This is because the validity of messages need to be checked. This design makes the protocol vulnerable to a reflection

attack. We describe the attack as well as a simple solution in section 5.

If we ignore the checks, this protocol works like the one for the passive attack model. The checks are necessary to fortify the protocol against active attacks, that could tamper the data being transmitted between nodes. We will now detail the main steps of the extended protocol.

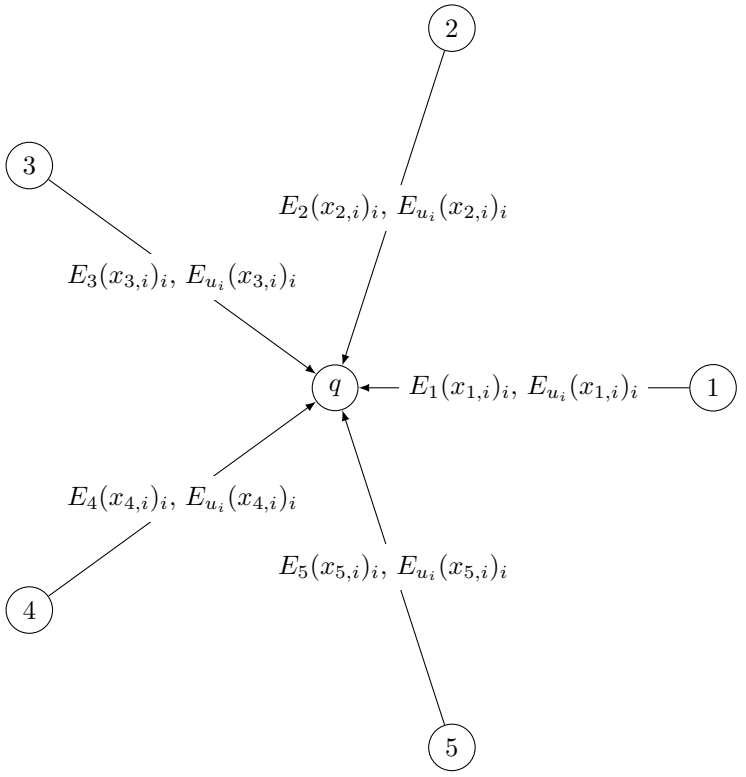
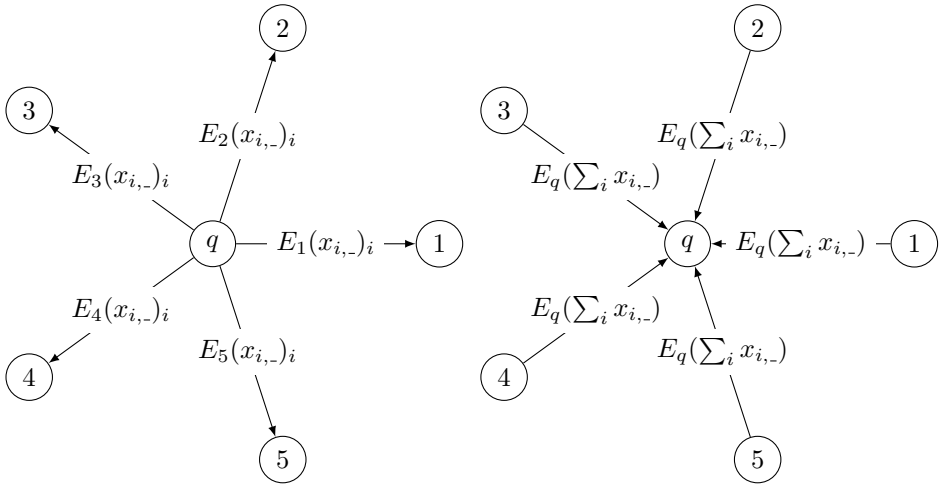



Figure 1: each part $i$ of each vote is sent to $q$, encrypted both for $i$ (destination) and $q$ (verification)

As shown in Figure 1, the first exchanges correspond to step 2. This is conceptually the step where the source nodes vote for the target node. We need to make sure that these votes are correct, i.e. that they belong to the valid range.

To check it, $q$ must compute $E_K(\sum x_{s,i})$ for some key $K$. This is why the parts are duplicated and encrypted with the sender's key: using the homomorphic property of $E$, $q$ can compute $\prod E_s(x_{s,i})$ and use the relevant SM-ZKP to check that the encrypted value belongs to the expected range.

However, this usage of SM-ZKPs only ensures validity for the parts which have encrypted with the sender's key. We need to ensure this for the parts which are transmitted to other source nodes as well. This is actually done by simply checking that the duplicated messages do contain the same content using the PE-ZKPs.




(a) each node gets the parts it must sum

(b) sums are sent back to $q$

Figure 2: in steps 3 (a) and 4 (b), the nodes are given parts (we do not care which) they can decrypt; they sum them and send the results to $q$ by re-encrypting the messages

These two sets of ZKPs ensure the validity of parts sent to other nodes. We must now check that what the node returns at step 4 is actually the sum of the received parts (see Figure 2). For this, $q$ will also compute the $s$-encrypted message containing the sum of the received parts (using homomorphic properties of E) and then compare it to the $q$-encrypted message returned by $s$ using the new PZ-ZKPs.

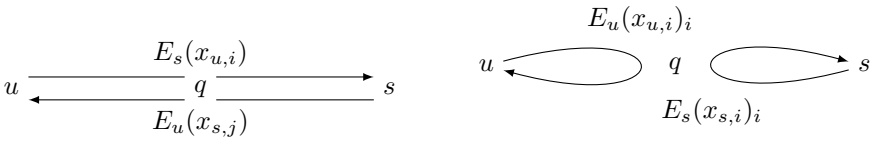
**Note:** $s$ does need to compute the sum itself because, otherwise $q$ would be able to ask $s$ to decrypt arbitrary data

Thus, source nodes cannot tamper with the protocol since they cannot cheat by picking a vote outside of the selected range, nor by returning a value different from the sum of the parts. Moreover, they gain no information on other nodes' private data because each part they are given bears no information on the original vote from the other source node.

Nodes which are not involved in the protocol obviously gain no insight. It remains to show how we can keep $q$ from using its privileged position to trick the nodes into giving him private data using a reflection attack.

## 5 Reflection attack

First, let us explain how $q$ can make a reflection attack to get all trust votes from source nodes in the protocol above. It is actually fairly simple: instead of transmitting messages from one node to others, it sends the $s$-encrypted messages, designed for vote verification, back to $s$. Following the protocol for the active attack model, node $s$ will now compute the sum of these $k$ encrypted parts with its own local part and send the re-encrypted result to $q$. Now, $q$ receives value $\sum_{1 \leq i \leq k+1} x_{s,i} = f_{s,t}$ (after $q$-decryption). The loopback is shown in Figure 3.




(a) $q$ should pass the $u_i$-encrypted messages from $s$ to $u_i$ (intended destination)

(b) instead, $q$ loops back $s$-encrypted messages (dumb encryption for checking)

Figure 3: the protocol expects $q$ to send messages to the right destination (a) but if $q$ loops back messages, $s$ is supposed to sum and decrypt its own parts

## 6 Defense against the attack

At first glance, it would seem that the fix is simple: $s$ must detect when its messages are looped back to him. Alas, because of the homomorphic properties of $E$, $q$ can alter the parts so that they look different from the ones which $s$ sent. Even worse, $q$ can obfuscate the data by adding a random mask before re-sending the parts and removing it after the sum has been computed:

$$\underbrace{x_{s,k+1} + \sum_{1\leq i\leq k} \underbrace{x_{s,i} + r_i}_{\text{data masked by } q}}_{\text{returned by } s} - \sum_{1\leq i\leq k} r_i = \sum_{1\leq i\leq k+1} x_{s,i} = f_{s,t}$$

Ultimately, the problem is that source nodes must have a way to securely check the sender of the message. This immediately leads to the use of cryptographic signatures: when a source node $s$ sends a part to $u_i$ through $q$, it sends $S_s(x_{u_i,i})$ along with it; when $u_i$ gets the part it must check the signature using the public key of $s$ (that he already knows).

Since we have $s$ sending more data, we must have additional checks to verify their consistency. This is done simply by having $q$ check the signature sent by $s$ as well.

An important point to remember is that the protocol relies on the assumption that the nodes' public keys are known by the other nodes. This is a necessary requisite since, without this preliminary knowledge of the keys, $q$ would be able to easily emulate any network to fool a target node. It would be possible to avoid sharing all keys by relying on the channels assumed to be secure, but it would imply numerous additional packets to ensure that all nodes communicate correctly, defeating the purpose of avoiding the initial sharing of keys.

## 7 Conclusion

With the revised protocol, we are ensured of the privacy-wise security of this reputation system, against the reflection attack mounted by a malicious querying node. This protocol is efficient since its complexity is linear with the number of involved nodes.

As future work, it would be interesting to study reputation systems which rely on different evaluation functions: we used the mean of trusts but it may not be the most effective way as extremal trust (0 or 1) is an easy choice for the user while non-extremal trust ($\in ]0, 1[$) may bear more significance.

## References

[GMW87] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game. In *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing*, STOC '87, pages 218–229, New York, NY, USA, 1987. ACM.

[HBB11] Omar Hasan, Lionel Brunie, and Elisa Bertino. Preserving Privacy of Feedback Providers in Decentralized Reputation Systems. *Computers & Security*, 31(7):816–826, October 2011. http://dx.doi.org/10.1016/j.cose.2011.12.003.

[HBBS13] Omar Hasan, Lionel Brunie, Elisa Bertino, and Ning Shang. A Decentralized Privacy Preserving Reputation Protocol for the Malicious Adversarial Model. Technical Report RR-LIRIS-2012-008, LIRIS UMR 5205 CNRS/INSA de Lyon/Université Claude Bernard Lyon 1/Université Lumière Lyon 2/École Centrale de Lyon, June 2013.

[Pai99] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Proceedings of the 17th International Conference on Theory and Application of Cryptographic Techniques*, EUROCRYPT'99, pages 223–238, Berlin, Heidelberg, 1999. Springer-Verlag.

[PRT04] Elan Pavlov, Jeffrey S. Rosenschein, and Zvi Topol. Supporting privacy in decentralized additive reputation. In *Second International Conference on Trust Management*, 2004.

[Yao82] Andrew C. Yao. Protocols for secure computations. In *Proceedings of the 23rd Annual Symposium on Foundations of Computer Science*, SFCS '82, pages 160–164, Washington, DC, USA, 1982. IEEE Computer Society.

[Yao86] Andrew Chi-Chih Yao. How to generate and exchange secrets. In *Proceedings of the 27th Annual Symposium on Foundations of Computer Science*, SFCS '86, pages 162–167, Washington, DC, USA, 1986. IEEE Computer Society.