An anonymous reputation mechanism for cloud computing networks using volunteer resources

Miriam Aguilar Morillo
Universitat Oberta de Catalunya
Av. Tibidabo 39-43, 08035 Barcelona, España
maguilmo@uoc.edu

Abstract

One of the major problems when using non-dedicated volunteer resources in a distributed network is the high volatility of these hosts since they can go offline or become unavailable at any time without control. Furthermore, the use of volunteer resources implies some security issues due to the fact that they are generally anonymous entities which we know nothing about. So, how to trust in someone we do not know?.

Over the last years an important number of reputation-based trust solutions have been designed to evaluate the participants' behavior in a system. However, most of these solutions are addressed to P2P and ad-hoc mobile networks that may not fit well with other kinds of distributed systems that could take advantage of volunteer resources as recent cloud computing infrastructures.

In this paper we propose a first approach to design an anonymous reputation mechanism for *CoDeS* **[1]**, a middleware for building fogs where deploying services using volunteer resources. The participants are reputation clients (RC), a reputation authority (RA) and a certification authority (CA). Users need a valid public key certificate from the CA to register to the RA and obtain the data needed to participate into the system, as now an opaque identifier that we call here pseudonym and an initial reputation value that users provide to other users when interacting together. The mechanism prevents not only the manipulation of the provided reputation values but also any disclosure of the users' identities to any other users or authorities so the anonymity is guaranteed.

Keywords: *anonymity, reputation, cloud computing, blind signature*

1. Introduction

Reputation mechanisms are foreseen to provide trust to those participants in a

system willing to interact with others who are strangers in order to minimize their sense of uncertainty about the results or the risks of such an interaction. In this way, they can be seen as a kind of making-decision helpers about to trust or not in a participant of a network before interacting with him.

A reputation system is implemented using one or more mechanisms that allow a participant from that system to know if another participant can or cannot be trusted. This become specially important in networks where users are anonymous. How to trust in someone we do not know? A reputation system tries to help us to take this decision by making available mechanisms allowing the generation, storage and distribution of ratings about users' behavior. These ratings evaluate their trustworthiness, distinguishing those that behave in a dishonest (voluntary) or nonperformance (involuntary) way.

Dishonest users are often motivated by selfish or malicious intend. A selfish participant manipulates reputation data in its own benefit, for example, increasing its reputation values or the values of his malicious colleges in a coalition, while a malicious user attempts to degrade the reputation values of others to corrupt the entire system. They can work alone or in coalitions of coordinated attackers. Different attacks can be the injection of false information into the system, the manipulation of the reputation system data, refusing to forward data or attempting to make unavailable the system through causing denial of service. Some of these attacks are discussed in **[2]**, identifying which parts of a reputation system are exploited with each attack and exposing some defense techniques to address them.

In general, a reputation system must deal with two main issues: the trust model in charge of transforming the system's available data into inputs to the reputation formulation in order to obtain a global trust value, and the protocol for storing, distributing and accessing these reputation values.

In this work we present a reputation mechanism for distributing and accessing reputation in an anonymous way. As far as we know, this is the first approach to build an anonymous reputation system for cloud computing networks using volunteer resources. Although we initially proposed the reputation system for CoDeS, the mechanism can be used in other applications or kind of networks where participants need to be reliable while preserving their anonymity.

The remaining parts of this paper are organized as follow: In Section II we discuss related work. Section III we introduce the reputation mechanism design. In Section IV we proceed with the validation. We do a security analysis in Section V and conclude the whole paper in Section VI.

2. Related work

A number of reputation protocols have been proposed over the last years, mainly designed for P2P and ad-hoc mobile networks. In this section we take a look at few of them by reviewing a set of features that any reputation protocol should consider and describing how the existent protocols implement them. Next, we quickly discuss which of these properties meet our reputation

mechanism and how they are implemented.

a. Anonymity: A participant is anonymous in a system when no personally identifying information is known by other users. Anonymity is an essential capability in a reputation protocol to give participants more reliability when rating other participant's behavior, because no user should be afraid of being attacked by the users he voted. The anonymity is hard to implement, as long as it can be lost in several ways as described in **[2]**. Most of reputation mechanisms use opaque identifiers as pseudonyms to preserve anonymity as in *P2PRep* **[3]**, *TrustMe* **[4]** and *SuperTrust* **[5].**

b. Privacy and integrity: A reputation protocol should provide mechanisms to guarantee the integrity of the reputation data at distributed and storage levels to avoid votes from being undisclosed or user's reputation data modified. Some attacks as the Man-in-the-Middle can let a user to intercept the communication between two entities, for example, between a participant asking for other participant's reputation and the entity providing the reputation data for that user. To avoid this kind of attack, *TrustMe* **[4]** uses encryption mechanisms to protect integrity data at communication level. Other systems as *SuperTrust* **[5]** uses encryption at communication as well as at storage level, in a way that reputation messages are never disclosed.

c. Efficiency: A reputation protocol should limit the consumption of users resources as network bandwidth and storage or computational capacity. Techniques using message broadcasting, flooding, gossiping and other epidemic based communication to exchange reputation data between users as in *P2PRep* **[3]**, *CONFIDANT* **[6]**, *CORE* **[7]**, **[8]** and **[9]** can slow down the entire system if participants do not have enough connection capacity. More efficient reputation data communication and storage mechanisms are based in the use of some kind of distribution hierarchies as *SuperTrust* **[5]** or *DHTs* as in *TrustMe* **[4]**, *EigenTrust* **[10]**, *PeerTrust* **[11], [12],** *PowerTrust* **[13],** and **[14]**.

d. Persistence: The persistence refers to the availability of the reputation data in the system along the time. The storage of reputation data can be transient or persistent. Transient storage is defined to be non-durable, as for example in a decentralized architecture, participants going out of the system take with them the data they store becoming unavailable for the rest of the system. This is the case for *P2PRep* **[2]**. With a persistence storage, otherwise, the reputation data is maintained accessible in the system along the time, preserving historical data for each participant, as for example in *TrustMe* **[4],** *SuperTrust* **[5]** and *PowerTrust* **[13].** These reputation systems implement mechanisms to make possible the transfer of trust values when a participant leaves the network.

e. Scalability: A reputation protocol must be able to continue working even when the system grows up in terms of number of participants. The capacity of a system to scale well strongly depends on the architecture for data

dissemination which can be a centralized, decentralized or semi-centralized. The first one involves one central entity regarding for the storage and dissemination of data as in *Sarmenta* proposal **[15]**. In a decentralized approach each participant in the system is responsible for some portion of the reputation data. *P2PRep* **[3]**, *TrustMe* **[4]** and *CONFIDANT* **[6]** are examples of decentralized reputation systems. In the semi-centralized or hybrid systems the responsibility for the storage and distribution of reputation data is shared between a group of entities as in *SuperTrust* **[5]**.

f. Legitimacy: Only users who have interacted with other users should be able to vote them. Furthermore, the reputation protocol should assure that only one vote per interaction is done. In *SuperTrust* [17] users who have interacted together in the network exchange a proof of interaction in form of a private key signed message including peers identifiers and the time at which the interaction took place.

g. Redundancy: The reputation protocol must be robust in front of users giving false low ratings to other users to intentionally degrade their reputation. One of the typical ways for building robustness in a reputation system is the use of techniques based in data duplication as *majority voting* in **[15].**

h. Easy contribution: The way in which a participant must rate another should not be tedious nor time consuming to let the user easily contribute to the reputation system. In *TrustMe* **[4]** and *SuperTrust* **[5]**, a user only needs to send one message to vote the behavior of a participant.

In its first stage, the reputation mechanism here presented satisfies most of the properties described above:

(a) We use temporal pseudonyms to identify participants in the system that need to be renewed by the participants themselves before a given expiration time. This makes hard to trace users by their pseudonyms as they are required to change them frequently so the mechanism guarantees their anonymity. **(b)** Our mechanism makes use of public/key encryption schemes to protect communication between the different system entities. **(c)** Each participant only stores his own reputation data which is directly transmitted to the participant he want to interact with. Thus, we make an efficient usage of users' resources since it is not storage nor network bandwidth consuming. **(d)** The reputation mechanism is persistent. All votes received from participants about other participant's behavior are taken into account to compute his next reputation value. **(f)** Users who have interacted together exchange a proof that is unequivocally associated to their pseudonyms so when submitting a vote the system can check if there really was an interaction with the user being voted. **(h)** Only one message is necessary to be able to vote a participant and contribute to the reputation system.

3. Mechanism description

The anonymous reputation mechanism presented is based in the existence of the following participants:

1. A Certification Authority (CA) in charge of providing valid public key certificates to all other participants in the system.

2. A Reputation Authority (RA). This is the key agent in our protocol. It acts as a centralized entity that is trusted by clients. The Reputation Authority is responsible of the following actions:

    a. Register new participants into the reputation system, providing them with the reputation data needed to interact with the others.

    b. Receives and store reputation votes sent by participants about others participants behavior.

    c. Renew participants pseudonyms when requested.

3. The reputation client (RC). They are all the other participants in the system who can register into the RA after obtaining a valid public key certificate from the CA. The client interacts into the system with the reputation data they have obtained from the RA after the registration process or after a renewal pseudonym action.

In this paper we assume that both the RA and reputation clients have a public/private key pair and that they have already obtained a valid public key certificate from the CA.

The mechanism is composed of three different protocols:

1. Registration protocol

2. Pseudonym renewal protocol

3. Voting protocol

In this section we first describe the cryptographic techniques used in this work and next we present the different protocols that make up our anonymous reputation mechanism.

3.1 Cryptographic techniques used

The protocol takes advantage of the following cryptographic functionalities:

Signatures on messages

By signing a message, a sender gives the receiver reason to believe that the message was created by himself and that it was not altered in transit. It is based in the use of asymmetric cryptographic keys as those obtained with the RSA algorithm. In our design messages sent out are signed by the sender's private key and authenticated by the receiver using the sender's public key.

Encryption of messages

All messages are encrypted with the recipient's public key before being sent to the network. The encryption assures the privacy and confidentially of the data since nobody excepts the recipient itself can disclose the message by decrypting it with his private key.

Digest of messages

The protocol uses a cryptographic one-way hash function to generate a digest of the data exchanged between the RA and the reputation clients. Digests of messages are used together with signatures to verify the authenticity and the integrity of such a messages, detecting any corruption or manipulation of the data transmitted.

Blind RSA signatures

Blind signatures as proposed by Chaum **[16]** are the base of the anonymity achieved with our reputation protocol. As we will explain later, each user in the system has a pseudonym that need to be signed by the RA before it can be used to interact in the system with other participants. In order for a user to remain entirely anonymous, even the RA must know anything about it. To make that possible, the user blinds the pseudonym with a random sequence of bits that he only knows before transmitting it to the RA. Thus, what the user obtains from the RA is a blind signature of his pseudonym that he can unblind using the same random factor previously applied to finally get the RA's true signature of his pseudonym.

Actually, the user not only get the blind signature of this pseudonym but also of his reputation value that RA adds to the message being signed. We will refer to this kind of signature as an hybrid blind signature because it contains data providing from two different entities, the RA and the user.

3.2 Registration Protocol

Before a user can interact with other participants in the system, he previously needs to register into the RA to obtain the reputation data.

The steps to register in the RA are described hereafter.

1. The reputation client starts the protocol by sending his public key certificate to the RA.

2. After verifying the validity of the certificate, the RA sends to the client his public key certificate and a challenge to be signed by the client in order to prove the ownership of the certificate he has previously sent.

3. The client verifies the RA certificate received, signs the challenge and sends a registration message containing this signature and the digest of a programatically obtained pseudonym encrypted with the public key of RA and blinded with a random value r.

4. The RA validates the challenge signature and if valid, sends to the client a confirmation message. This confirmation message is composed of a reputation initial value of 0, an expiration time for the client's pseudonym and an hybrid blind signature containing the pseudonym digest previously received together with the digest of the reputation and pseudonym expiration time assigned to the client.

5. The reputation client unblinds the message and obtain what we call here the authentication data, that is, the RA's signature of a digest formed by the client pseudonym, his reputation and his pseudonym expiration time.

Figure 1 shows the sequence of steps and participants involved at each step in the registration protocol.

3.1.1. Registration protocol messages

A. Certificate message

The certification message (**ms_{CERT}**) is used to exchange public key certificates between the Reputation Authority and the Reputation Clients at the beginning of the communication between them.

The message simply contains the public key certificate issued by the Certification Authority of the entity sending the message.

B. Challenge message

The challenge message (**$ms_{CHALLENGE}$**) is sent by the Reputation Authority to the client. It contains a ms_{CERT} with the Reputation Authority public key certificate and a challenge to be signed by the Reputation Client with his private key in order to prove the ownership of the certificate he previously sent in a ms_{CERT}.

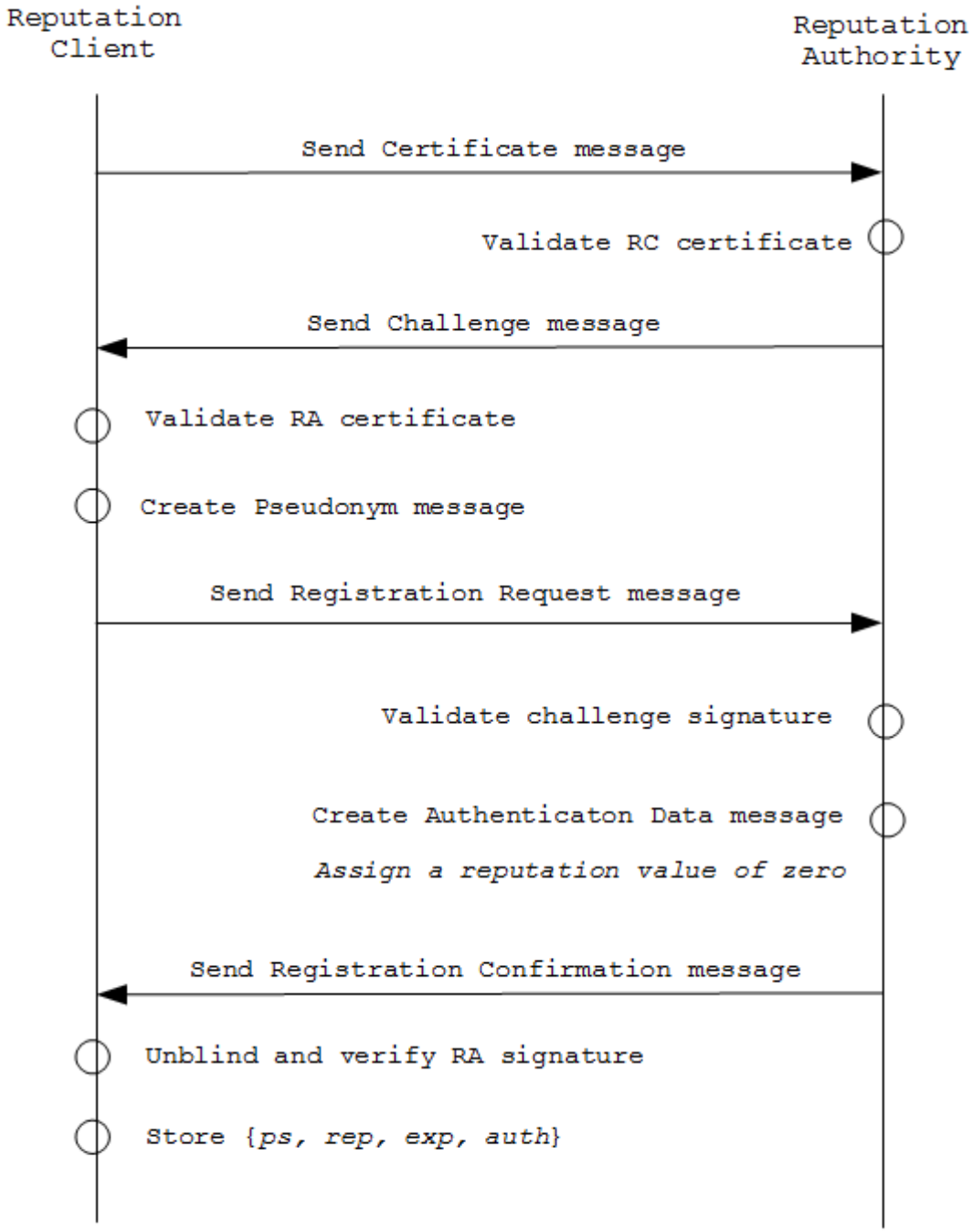

Figure 1: Sequence of messages in the Registration protocol

The message is composed as follows:

$\mathbf{ms_{CHALLENGE}} = \mathbf{ms_{CERT}}, \mathbf{ch}$

where:

$\mathbf{ms_{CERT}}$: A certification message containing the public key certificate of the RA.

ch: A challenge to be signed by the RC.

C. Pseudonym message

The pseudonym message (**ms_{PSEUDO}**) is part of the registration request message and of the pseudonym renew request message that are sent by the reputation client to the RA in order to register or renew a pseudonym respectively.

This message is composed as follows:

$$ms_{PSEUDO} = hash(ps) \cdot r^{eRA} \mod N^{RA}$$

where:

ps: client pseudonym.

r: random number less than N

eRA: RA public key exponent.

N: RA public key modulus.

The pseudonym digest is blinded by the random number r in such a way that RA cannot obtain any useful information about ps.

D. Registration Request message

The registration request message (**ms_{REGIS}**) is sent by the client to the RA in order to be registered in the system with a programatically obtained pseudonym.

This message is built in the following way:

$$ms_{REGIS} = sign_{A}(ch),\ ms_{PSEUDO}$$

where:

$sign_{A}$ (ch): the signature of the challenge sent by the RA to the client in the challenge message.

ms_{PSEUDO}: client pseudonym message.

E. Authentication Data message

The authentication data message (**ms_{AUTH}**) is integrated in the confirmation message (see below) sent by the RA to the client.

The client will use this message as part of his authentication to later interact in the system with other clients.

This message is composed as follows:

$$\mathbf{ms_{AUTH}} = \mathbf{(hash(ps) \cdot r^{eRA} \cdot hash(rep,exp))^{dRA} \bmod N^{RA}}$$

where:

ps: client pseudonym.

r: random number chosen by the client and less than N.

rep: initial client reputation

exp: expiration time of client pseudonym.

eRA: RA public key exponent

dRA: RA private key exponent

N: RA private key modulus

The authentication message is an hybrid blind signature: it contains information about the RA and the reputation client.

F. Registration Confirmation message

The registration confirmation message (**$ms_{CONFIRM}$**) is sent by the RA to the client to confirm its registration in the reputation system and to provide him its initial reputation, the expiration time of its pseudonym, as well as the authentication message that client will use to interact in the system.

This message is formatted as follows:

$$\mathbf{ms_{CONFIRM} = rep, exp, ms_{AUTH}}$$

where:

rep: reputation assigned to client

exp: client pseudonym expiration time

$ms_{AUTH:}$ client authentication data message

The initial assigned reputation will be 0.

3.3 Pseudonym Renewal Protocol

Every reputation client will have to renew the pseudonym before the pseudonym validity time expires.

Every time a client renews his pseudonym, the RA computes his new reputation taking into account all the votes received for that client about his behavior.

If the client does not renew this pseudonym before the assigned expiration time, the client will need to register again. In this case, the reputation clients will be initialize to zero.

Note: It is out of this work the computing algorithm used to calculate the new reputation value based in all reputation votes received.

The steps to renew a pseudonym are described hereafter:

1. The reputation client creates a new pseudonym and generates a pseudonym message as in step 3 for registration protocol.

2. The reputation client sends a pseudonym renew request message containing current authentication data and encrypted information about the new pseudonym.

3. The reputation authority verifies that client authentication data has not yet expired and recalculates the new reputation for this client based in the reputation information received from the rest of entities in the system since client registration or client last pseudonym renewal.

4. The reputation authority sends a pseudonym renew confirmation message to the client, providing him with the new reputation value and the expiration time for the new pseudonym. This message format is the same as the registration confirmation message.

5. The client decrypts the message and updates his authentication data with the new received one that it will use from now to interact in the reputation system.

Figure 2 shows the sequence of steps and participants involved at each step in the pseudonym renewal protocol.

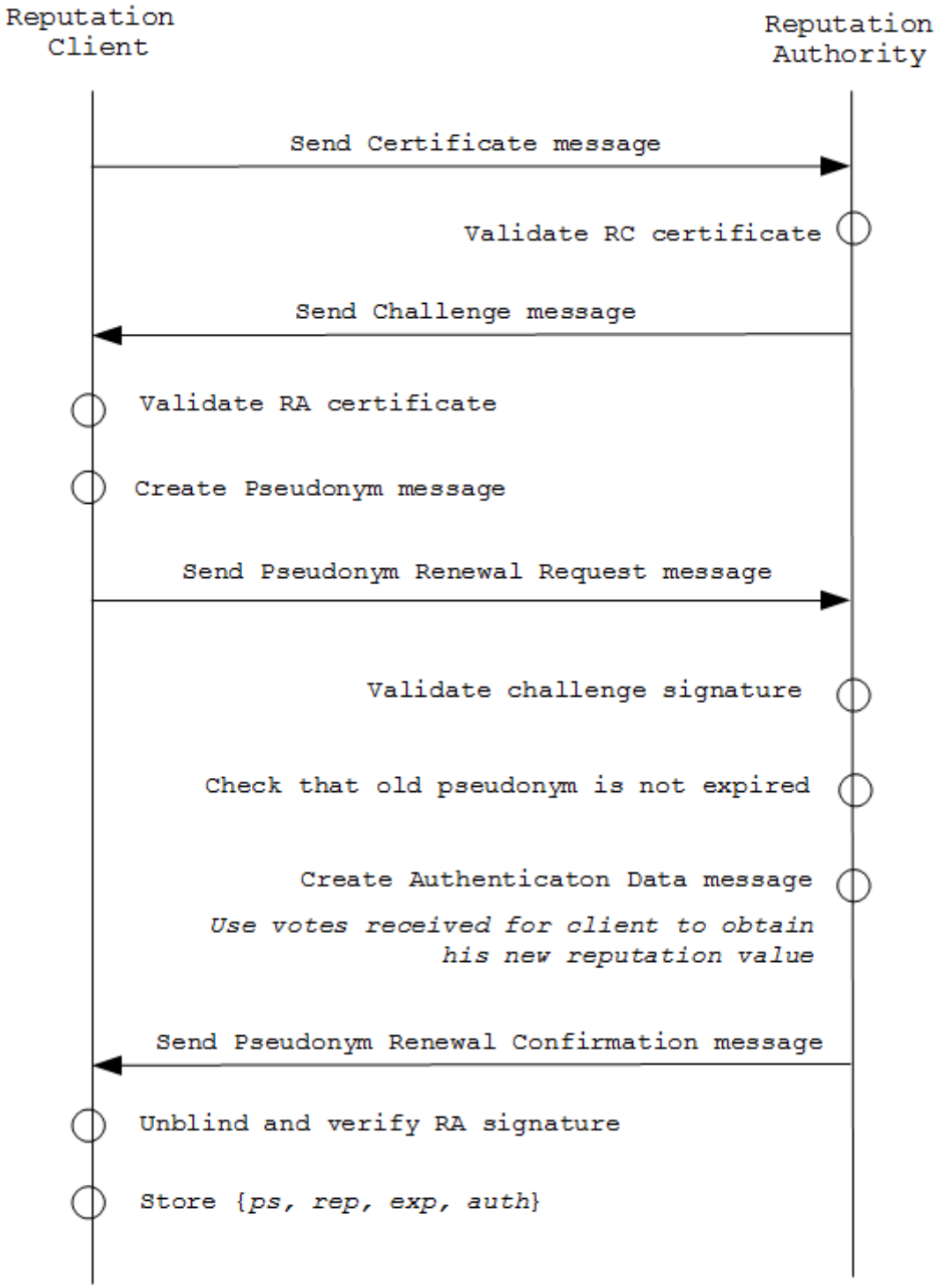

Figure 2: Sequence of messages in the Pseudonym Renewal protocol

3.3.1. Pseudonym Renewal protocol messages

A. Pseudonym Renewal Request message

The pseudonym renew request message (**ms_{RENEW}**) is sent by the client to the

RA in order to register a new pseudonym before his current pseudonym expires.

This message is formatted as follows:

ms_{RENEW}=ps, rep, exp, ms_{AUTH}, ms_{PSEUDO}

where:

ps: current client pseudonym

rep: current client reputation

exp: current client pseudonym expiration time

$ms_{AUTH:}$ current client authentication data message

$ms_{PSEUDO:}$ new client pseudonym message

B. Pseudonym Renewal Confirmation message

The pseudonym renewal confirmation message is sent by the RA to the client in order to confirm him that the pseudonym renewal has been successfully done and to provide the new reputation, pseudonym expiration time and authentication data message to the client.

This message is equivalent to the registration confirmation message (**$ms_{CONFIRM}$**).

3.4 Voting Protocol

The reputation clients are known in the system by their pseudonyms. Each time a client A interacts with a client B, both clients have the possibility to vote about the others behavior.

The votes are sent to the RA, who, after validating that the client who votes is a registered client and verifying that his pseudonym is still valid (not used and not expired) , stores the vote received in a table with pairs *{ps, vote}.*

3.4.1. Voting protocol messages

The Voting message (**ms_{VOTING}**) is sent by the client to the RA in order to give a reputation vote about another client's behavior after interacting with him. Before being able to submit a vote, both clients have to exchange a proof of interaction. As a first approach, we have considered the *seed* with which the participant's pseudonym was created. Since there is no way to get the seed of a pseudonym from the pseudonym itself, we can be sure that an interaction took place between two users if both have the other pseudonym's seed.

The message is composed as follows:

$$\mathbf{ms_{VOTING} = auth_A\ , (hash(ps_A).hash(\ rep_{A,}, exp_{A,}), vote_B, ps_B, seed_B)^{eRA}\ mod\ N_{RA}}$$

where:

$auth_A$: client A authentication data

ps_A: client A pseudonym

rep_A: client A reputation

exp_A: expiration time for ps_A

$vote_B$: vote given by A to client B

ps_B : client B pseudonym

$seed_B$: client B pseudonym's seed

e_{RA} : RA public key exponent

N_{RA}: RA public key modulus

4. Validation

We have built a prototype in order to validate the proposed reputation protocol design. The prototype has been developed in Java language.

The ReputationAuthority has been implemented as a highly scalable, asynchronous event-driven network server using the Reactor pattern **[17]**. The RA server listens for incoming requests from reputation clients, accepts connections, reads and process requests and creates and sends the response with the reputation data for the client. It has a thread pool that can be configured to optimize the CPU utilization.

During the simulation we have used 2048-bit RSA keys as recommended by *RSA Laboratories* in **[18]**. Due to the high number of asymmetric encryption/decryption operations that are done in the Register and Pseudonym Renewal protocols and since they are high CPU consuming, we have deployed the RA in a powerful server with two dual-core AMD Opteron processors at 2,6Ghz.

For the client side, we have implemented a multithreaded environment in a second server to launch reputation client instances in scheduling independent processes at a configurable rate.

We model response time as the elapsed time from a reputation client

connection to the RA to initiate a registration or pseudonym renewal till the time the client receives the reputation data to interact in the system. Since both servers used for our simulations are located in the same internal network and the RTT between them is not significant – about 0.210 ms – we obtain the protocol's response times without the impact of the network latency. The validation of the reputation mechanism in a more realistic environment is left as a future work.

We have measured the average response time in relation to the total number of requests. Our measurements were done for various number of requests sent at different requests/second rates. Figures 3a and 3b depict our findings.

Every data point in these figures has been averaged over 5 runs, in order to ensure consistent measurements.

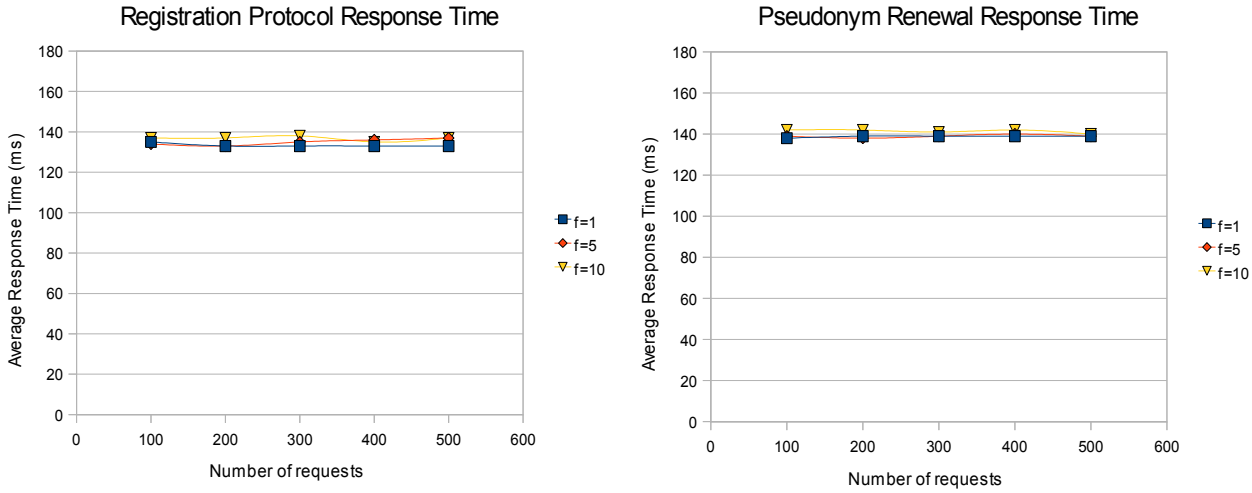

Figure 3: (a) Average response time in the Registration protocol. (b) Average response time in the Pseudonym Renewal protocol.

The graphs reveals an average response time around 140ms in both protocols which we consider acceptable. We also observe a slightly higher response time for the Pseudonym Renewal protocol. We believe that this may probably be due to the time used in the check done in the Pseudonym Renewal protocol to verify that the pseudonym to be renewed has not yet expired.

The results show that our reputation mechanism is able to continue working efficiently even when the number of requests and/or participants increase.

5. Security analysis

We have already discussed the security properties achieved by our reputation mechanism in section II. In this section we will limit to describe the various possible vulnerabilities that should be addressed in a second stage of design.

One of the main shortcoming of using our reputation protocol is that it is not a pure decentralized system. The RA acts like a trust server that stores the reputation votes and provides new reputation data where users renew pseudonyms, thus it suffers from the typical shortcomings characteristic of any centralized system, including presenting a single-point of failure that make the system stops working in case of denial of service.

Other important shortcoming is that malicious users can manipulate the reputation of other users by reporting false votes to lower their reputations. Our protocol is extremely vulnerable to this aspect since it does not authenticate the origin of the received votes nor use any redundancy technique as majority voting to validate them. In the same way, the reputation of a user can be positively exaggerated by one or more users with similar interests working together in a coalition in order, for example, of carrying out fraudulent actions once their reputations become high enough to be trusted by others.

Another question not related to security issue but that potentially represents a problem is the cold start. A user who has just joined the system has a reputation value of zero so existing users in the system may tend to isolate the new user since they lack trust information about the new participant. One solution here should be to introduce in the system an initial list of trusted peers with whom they can interact.

6. Conclusions and Future work

In this work we have introduced a reputation protocol for CoDeS, a middleware for building fogs where deploying services using available resources. As far as we know, this is the first approach to build a reputation based-trust system for cloud computing networks exploiting voluntary resources. Although the protocol has been designed with the previous mentioned goal in mind, it can be used in other applications or kind of networks where participants need to be reliable while preserving their anonymity.

In its first stage of design, the mechanism satisfies most of features that were identified in section II as necessary in a reputation system – anonymity, privacy and integrity, efficiency, persistence, legitimacy and easy contribution. We have shown in section IV that our reputation mechanism has acceptable average response times even when the number of participants in the system increase. However we think that simulations should be repeated in a more realistic environment to study the impact of the network latency in the system's response time.

We have also identified a significant number of shortcomings in section V that should be addressed in the reputation mechanism future work.

References

[1] Daniel Lázaro, Joan Manuel Marquès and Xavier Vilajosana. *Fog computing: creating a cloud out of enterprise resources.* IEICE TRANSACTIONS on Information and Systems Vol.EXX-D No.X pp.aaaa-eeee ISSN:0916-8532. - 2010 (IF=0,44) (Under review)

[2] Kevin Hoffman, David Zage, Cristina Nita-Rotaru. A *Survey of Attack and Defense Techniques for Reputation Systems*. ACM Computing Surveys (CSUR) , Volume 42 Issue 1. December 2009.

[3] F. Cornelli, E. Damiani, S.D.C di Vimercati, S. Parasboschi and P. Samarati. *Choosing Reputable Servents in a P2P Network*. Proc. 11th Int'l World Wide Web Conference, Hawaii, USA. May 2002.

[4] Aameek Singh, Ling Liu. *TrustMe: Anonymous Management of Trust Relationships in Decentralized P2P Systems*. Proc. 3rd Int'l IEEE Conference on Peer-to-Peer Computing. September 2003.

[5] Tassos Dimitriou, Ghassan Karame and Ioannis Christou. *SuperTrust – A Secure and Efficient Framework for Handling Trust in Super Peer Networks.* Proc. of ACM PODC 2007, 2007.

[6] Sonja Buchegger, Jean-Yves Le Boudec. *Performance Analysis of the CONFIDANT Protocol (Cooperation Of Nodes: Fairness In Dynamic Ad-hoc Networks)*. Proc. 3rd ACM Int'l symposium on Mobile ad hoc networking & computing, Lausanne, Switzerland. June 2002.

[7] Pietro Michiardi, Refik Molva. *CORE: A Collaborative Reputation Mechanism to enforce node cooperation in Mobile Ad hoc Networks*. Proc. IFIP TC6/TC11 6th Joint Working Conference on Communications and Multimedia Security. 2002

[8] Ernesto Damiani, Sabrina De Capitani di Vimercati, Stefano Paraboschi, Pierangela Samarati, Fabio Violante. *A reputation-based approach for choosing reliable resources in peer-to-peer networks*. Proceedings of the 9th ACM conference on Computer and communications security, Washington, DC, USA. 2002

[9] Ali Aydn Selçuk, Ersin Uzun and Mark Resat Pariente. *A Reputation-Based Trust Management System for P2P Networks*. Proc. 4th Int'l Workshop on Global and Peer-to-Peer Computing, Chicago, USA. April 2004.

[10] Sepandar D. Kamvar, Mario T. Schlosser, Hector Garcia-Molina. *The EigenTrust Algorithm for Reputation Management in P2P Networks*. Proc. 12th International World Wide Web Conference. 2003.

[11] Li Xiong and Ling Liu. *PeerTrust: Supporting Reputation-Based Trust for Peer-to-Peer Electronic Communities*. IEEE Transactions on Knowledge and Data Engineering, vol. 16, no. 7, pp. 843–857. July 2004.

[12] So Young Lee, O-Hoon Kwon, Jong Kim and Sung Je Hong. A Reputation Management System in Structured Peer-to-Peer Networks. Proc. 14th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprise, Washington, DC, USA. 2005.

[13] Runfang Zhou, Kai Hwang. *PowerTrust: A Robust and Scalable Reputation System for Trusted Peer-to-Peer Computing*. Parallel and Distributed Systems, IEEE Transactions on, vol. 18, no. 4. 2007.

[14] Natalya Fedotova, Luca Veltri. *Reputation Management for DHT-based Collaborative Environments*. Computer Communications, Volume 32, Issue 12. 2009

[15] Luis F.G Sarmenta. *Sabotage-Tolerance Mechanisms for Volunteer Computing Systems*. Proc. 1st International Symposium on Cluster Computing and the Grid. 2001

[16] D. Chaum. *Blind Signatures for Untraceable Payments*. In Advances in Cryptology – CRYPTO '82, pages 199–203. Springer-Verlag, Berlin, 1983.

[17] Schmidt D.C. *Experience using design patterns to develop reusable object-oriented communication software*. *CACM* 38(10), pp 65-74 , 1995

[18] http://www.rsa.com/rsalabs/node.asp?id=2264