Privacy-Preserving Reputation Mechanism: A Usable Solution Handling Negative Ratings

Paul Lajoie-Mazenc, Emmanuelle Anceaume, Gilles Guette, Thomas Sirvent, Valérie Viet Triem Tong

Privacy-Preserving Reputation Mechanism: A Usable Solution Handling Negative Ratings*

Paul Lajoie-Mazenc[1], Emmanuelle Anceaume[2], Gilles Guette[1], Thomas Sirvent[3], and Valérie Viet Triem Tong[4]

[1] Université de Rennes-1/IRISA firstname.lastname@irisa.fr
[2] CNRS/IRISA emmanuelle.anceaume@irisa.fr
[3] DGA Maîtrise de l'Information/IRISA thomas.sirvent@m4x.org
[4] CentraleSupélec/IRISA valerie.viettriemtong@supelec.fr

Abstract. Reputation mechanisms allow users to mutually evaluate their trust. This is achieved through the computation of a reputation score summarizing their past behaviors. Depending on these scores, users are free to accept or refuse to interact with each other. When users are virtual, volatile, or distant, an accurate evaluation of reputation scores is complex. Furthermore, users expect reputation mechanisms to preserve the privacy of their interactions and of their feedback. Existing solutions often rely on costly cryptographic tools that may lead to impractical solutions. In this article, we propose a usable privacy preserving reputation mechanism. This mechanism is distributed and handles non-monotonic ratings. Its implementation on cheap single board computers validates its adequacy to large-scale systems.

1 Introduction

Reputation mechanisms tend to be an effective tool to encourage trust and cooperation in electronic environments [1]. This is achieved by enabling users to rate services or people, based on their past experience. These ratings or feedback are aggregated to derive publicly available reputation scores. Reputation mechanisms either rely on a central authority or take advantage of the participating users to compute reputation scores. To circumvent the vulnerability of the former approach, both in terms of privacy and fault-tolerance, we present a reputation mechanism that meets security and trust requirements through distributed computations. While aggregating ratings is necessary to derive reputation scores, identifiers and ratings are personal data, whose collection and usage may fall under legislation [2]. Furthermore, as shown by recent works [3], solely relying on pseudonyms to interact is not sufficient to guarantee user privacy [4]. This has given rise to the proposition of a series of reputation mechanisms which address either the non-exposure of the history of raters [5], the non-disclosure of individual feedback [6–8], the secrecy of ratings and the k-anonymity of ratees [9], or the

* This work was partially supported by the ANR French project AMORES (grant #ANR-11-INSE-010).

anonymity and unlinkability of both raters and ratees [5, 10]. Regrettably, the search for privacy has led to algorithmic restrictions, in the sense that handling solely non-negative ratings seems to be the *sine qua non* condition to preserve user privacy [5, 10]: existing privacy-preserving mechanisms give their users the opportunity to skip some of the received ratings to increase their privacy, which is unfortunately not compatible with negative ratings. Furthermore, Baumeister *et al.* explain that "bad feedback has stronger effects than good feedback" on our opinions [11]. Thus, it is crucial to allow clients to issue negative ratings.

In the remaining of the article, we present the design and evaluation of a non-monotonic distributed reputation mechanism preserving the privacy of both parties. This work is the continuation of our preliminary work [12]. After having presented the state of the art in Section 2, we present in Section 3 the properties that should be met by a reputation mechanism to be secure, to preserve the privacy of all parties, and to handle non-monotonic ratings. Section 4 provides a description of the main principles of our approach to build such a mechanism, and their orchestration is presented in Section 5. The main contribution of this paper is presented in Section 6. This section shows that this unprecedented mechanism is computationally efficient, and thus implementable in large-scale applications. Finally, Section 7 concludes.

2 State of the Art

One of the first examples of reputation mechanisms has been set up by eBay. In this mechanism, clients and service providers rate each other after each transaction: ratings are either $+1$, 0, or -1 according to the (dis)satisfaction of users. The reputation score of a user is simply the sum of the received ratings. Resnick and Zeckhauser have analyzed this mechanism and the effects of reputation on eBay [13], and have highlighted a strong bias toward positive ratings. More elaborated reputations mechanisms have been proposed, such as the Beta Reputation System [14], methods based on the Dempster-Shafer theory of belief [15], or based on distributed hash tables [16–18]. Jøsang *et al.* propose a broad survey of reputation mechanisms [19], while Marti and Garcia-Molina focus on their implementation in P2P systems [20]. Indubitably, the nature of ratings and the computation of reputation scores have been thoroughly researched. In this work, we do not make any assumptions regarding the function that computes reputation scores. Indeed, our solution handles both positive and negative ratings, and may thus use any computation function.

One of the first known reputation mechanism taking the privacy of users into account has been proposed by Pavlov *et al.* [6]. Their solution presents a series of distributed algorithms for computing the reputation score of service providers without divulging the ratings issued by clients. Their solution has been improved by Hasan *et al.* [7, 18] for different adversary models, and stronger privacy guarantees. Similarly, Kerschbaum proposes a centralized mechanism computing the reputation scores of service providers without disclosing the individual ratings of the clients [8]. The secrecy of ratings contributes to the privacy of users,

but is clearly insufficient: service providers can still discriminate their clients according to their identity or to additional information unrelated to the transaction. As we previously mentioned, identifiers and ratings can be considered personal data. Steinbrecher argues that reputation mechanisms must guarantee both the anonymity of their users, and the unlinkability of their transactions to be fully adopted [4]. Both properties have been lately formalized by Pfitzmann and Hansen [21]. Namely, a user is *anonymous* if this user is not identifiable within a set of users, called the *anonymity set*. The transactions of a user are *unlinkable* if the participants in two different transactions cannot be distinguished. Hence, Clauß *et al.* [9] propose a centralized mechanism guaranteeing both the secrecy of ratings and the k-anonymity of service providers. However, beyond being centralized, this mechanism does not preserve the privacy of clients. Androulaki *et al.* [10] also propose a centralized reputation mechanism guaranteeing both the anonymity and the unlinkability of both parties. However, since providers send a request to the central bank for their ratings to be taken into account, only positive ratings are handled. In addition, this mechanism is vulnerable to *ballot-stuffing* attacks [1], that is, a single client can issue many ratings on a provider to bias her reputation. Whitby *et al.* [22] propose a technique mitigating ballot-stuffing attacks, however their technique requires the ability to link the ratings concerning the same provider. Bethencourt *et al.* [5] propose to compute such a link. That is, they propose a mechanism linking all the transactions that have occurred with the same partners, while preserving their privacy. However, beyond handling only positive ratings, their reputation mechanism requires high computational power, bandwidth and storage capacity. For instance, when proving their reputation score, providers must send about 500 KiB per received rating, which is unbearable from a practical point of view.

So far, preserving the privacy of both raters and ratees and handling both positive and negative ratings has been recognized as a complex challenge. Quoting Bethencourt *et al.*, "Most importantly, how can we support non-monotonic reputation systems, which can express and enforce bad reputation as well as good? Answering this question will require innovative definitions as well as cryptographic constructions" [5]. To the best of our knowledge, no distributed reputation mechanism preserves the privacy of its users and allow clients to efficiently issue both positive and negative ratings. This is the objective of this paper.

3 Model and Properties

Terminology. In the following, we differentiate transactions from interactions. A *transaction* corresponds to the exchange of a service between a client and a service provider, while an *interaction* is the whole protocol followed by the client and the provider, during which the clients get the provider's reputation and the client issues a *rating* on the provider. Note that we make no assumption about the nature of transactions: they can be, for example, web-based community applications or e-commerce ones. Once a transaction is over, the client is expected to issue a rating representative of the provider's behavior during the transaction.

Nevertheless, clients can omit to issue such a rating, deliberately or not. While dissatisfied clients almost always issue a rating, satisfied clients seldom do it. To cope with this asymmetry, we introduce the notion of *proofs of transaction*: a proof of transaction is a token delivered to providers for transactions during which the client did not issue a rating. Such proofs of transaction allow clients to distinguish between multiple providers that have the same reputation. We denote by *report* the proof of transaction associated with the client's rating, if any. These reports serve as the basis to compute reputation scores. Finally, we say that a user is *honest* if this user follows the protocol of the reputation mechanism. Otherwise, this user is *malicious*.

Model of the System. We consider an open system populated by a large number of users. A proportion of these users can be malicious (more details are given below). Before entering the system, users register to a central authority $\mathcal{C}$, that gives them identifiers and certificates. Once registered, users do not need to interact with $\mathcal{C}$ anymore. A user can act as a client, as a service provider, or as both, and obtains credentials for both roles. We also assume that users communicate over an anonymous communication network, *e.g.* Tor [23].

Properties of our Reputation Mechanism. Our reputation mechanism aims at offering three main guarantees to users. First and foremost, the privacy of users must be preserved. Second, users must always be able to cast their report. Finally, every data needed for the computation of reputation scores must be available and unforgeable. Privacy properties are stated in Properties 1 and 2, while Properties 3 and 4 are related to the undeniability of reports. Both properties expect that providers obtain proofs of transaction, and that clients are always able to cast ratings. Property 5 deals with reports unforgeability. Finally, Properties 6 and 7 respectively stipulate that the computation of the reputation scores cannot be biased by ballot-stuffing attacks, and that reputation scores are unforgeable. Note that since clients do not know the provider they are interacting with, targeted bad-mouthing attacks cannot be launched.

Property 1. *Privacy of service providers. When a client rates an honest service provider, this service provider is anonymous among all honest service providers with an equivalent reputation.*

Property 2. *Privacy of clients. When a provider conducts a transaction with an honest client, this client is anonymous among all honest clients. Furthermore, the interactions of honest clients with different providers are unlinkable.*

Property 3. *Undeniability of ratings. At the end of a transaction between a client and a provider, the client can issue a valid rating, which will be taken into account in the reputation score of the provider.*

Property 4. *Undeniability of proofs of transaction. At the end of a transaction between a client and a provider, the provider can obtain a valid proof of transaction.*

Property 5. *Unforgeability of reports. Let r be a report involving a client and a service provider. If r is valid and either the client or the provider is honest, then r was issued at the end of an interaction between both users.*

Property 6. *Linkability of reports. Two valid reports emitted by the same client on the same service provider are publicly linkable.*

Property 7. *Unforgeability of reputation scores. A provider cannot forge a valid reputation score different from the one computed from all the reports assigned to this provider.*

4 Building Blocks

4.1 Distributed Trusted Third-Parties

As explained in Section 1, service providers must not manage themselves their reputation score to guarantee their reliability. To solve this issue, we propose to construct a distributed trusted authority in charge of updating and certifying reputation scores. We call *accredited signers* the entities constituting this authority. This first distributed authority has two main features. Firstly, this authority must involve fairly trusted entities or enough entities to guarantee that the malicious behavior of some of them never compromises the computation of reputation scores. Secondly, this authority must ensure that providers remain indistinguishable from each other. Moreover, to ensure the undeniability of ratings, a client must be able to issue his report, even if the service provider does not complete the interaction. However, the precautions taken for that purpose must not imply sending identifying data before the transaction. In the same way, data identifying the client must not be sent before the transaction, even to ensure the undeniability of proof of transactions. To solve all these issues, we propose a distributed trusted authority in charge of guaranteeing that reports can be built. This distributed authority must collect information before the transaction, and potentially help one of the two parties afterwards; it must thus be online. We call *share carriers* the entities constituting this authority.

Both distributed authorities could be gathered in a single one. The drawback of this approach is that this distributed trusted authority should be simultaneously online, unique, and fairly trusted or reasonably large. The uniqueness and the participation in each interaction would induce an excessive load on each entity of this distributed authority. For efficiency reasons, we thus suggest distinct authorities. Accredited signers are then a unique set of fairly trusted or numerous entities, periodically updating the reputation scores of all providers. On the other hand, share carriers are chosen dynamically during each interaction among all service providers. Accredited signers manage every reputation score, and are thus critical in our mechanism. On the other side, share carriers are responsible for the issuing of a single report. Hence, they do not need to be as trustworthy as the accredited signers.

To deal with the privacy of both clients and providers, share carriers use *verifiable secret sharing* [24]. This basically consists in disseminating shares of a

secret to the share carriers, so that they cannot individually recover the secret, but allow the collaborative reconstruction of this secret.

4.2 Cryptographic tools

Our mechanism relies on cryptographic tools to guarantee its properties. The underlying structure of those tools is a bilinear group $\Lambda = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, G_1, G_2)$ in which $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ are three groups of prime order p that we write multiplicatively. The map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is non-degenerate and bilinear. $G_1 \in \mathbb{G}_1$ (resp. $G_2 \in \mathbb{G}_2$) is a group generator of $\mathbb{G}_1$ (resp. $\mathbb{G}_2$).

First, our mechanism uses *SXDH commitments* [25]. To commit to a value in $\mathbb{G}_1$ or $\mathbb{G}_2$, one needs two random scalars. Then, our mechanism relies on the *Non-Interactive Zero-Knowledge* (NIZK) proof system proposed by Groth and Sahai [25], which allows users to prove their possession of secrets without revealing the secrets. Instead, the secrets are masked by SXDH commitments. For instance, this proof system allows users to compute *Anonymous Proxy Signatures* [26], *i.e.* to sign messages without revealing the message, the signature, or their verification key. This requires particular signature schemes, *e.g. Structure-Preserving Signatures* [27]. Finally, as previously mentioned, our mechanism relies on verifiable secret sharing. Such a scheme allows a prover to split a secret into n shares, and to reconstruct the secret from t shares (with $t \leqslant n$). More specifically, the prover sends a share to n *share carriers*, and convinces a verifier that the verifier will be able to reconstruct the prover secret. To convince the verifier, the prover uses NIZKs to prove (a) the correctness of the secret, and (b) the consistency of the shares. An optimal choice for t is $t = \lceil n/3 \rceil$, which tolerates up to $t - 1$ malicious share carriers. In this case, the verifier accepts the sharing as soon as $2t - 1$ share carriers have confirmed the reception of their share. The analysis leading to this choice is detailed in the companion paper [28].

As explained in Section 2, reputation mechanisms must defend themselves against ballot-stuffing attacks. Bethencourt *et al.* [5] propose such a method by computing a value that depends only on the client and the provider, but that does not allow different providers to compare their clients. We propose a similar method, yet simpler, allowing to compute such an *invariant*. Let $\mathsf{Id}_{\mathrm{SP}} \in \mathbb{G}_1$ (resp. $\mathsf{id}_{\mathrm{Cl}} \in \mathbb{Z}_p$) be the identifier of the provider (resp. client). We then define the invariant as $\mathsf{inv} = {\mathsf{Id}_{\mathrm{SP}}}^{\mathsf{id}_{\mathrm{Cl}}}$. Note that the invariant must not be computed directly: it requires the client to know the provider's identifier, and vice versa. Hence, they jointly compute the invariant in three steps, which require an additional group element $Y_1 \in \mathbb{G}_1$. First, the provider computes a *pre-invariant* with randomness $r \in \mathbb{Z}_p$: $\mathsf{pre\_inv} = ({G_1}^r, \mathsf{Id}_{\mathrm{SP}} \cdot {Y_1}^r)$. The client then randomly chooses $s \in \mathbb{Z}_p$ to compute a *masked invariant*: $\mathsf{masked\_inv} = ({G_1}^s \cdot {Y_1}^{\mathsf{id}_{\mathrm{Cl}}}, {\mathsf{pre\_inv}_1}^s \cdot {\mathsf{pre\_inv}_2}^{\mathsf{id}_{\mathrm{Cl}}})$. Finally, the provider obtains the invariant from $\mathsf{masked\_inv}$: $\mathsf{inv} = \mathsf{masked\_inv}_2 \cdot {\mathsf{masked\_inv}_1}^{-r} = (\mathsf{Id}_{\mathrm{SP}})^{\mathsf{id}_{\mathrm{Cl}}}$. Note that the invariant is computed *after* the transaction, otherwise the provider would know whether she has already interacted with the client or not, which might introduce a bias in the provision of the service.

5 Reputation Protocol

Throughout the reputation protocol, users need cryptographic keys and identifiers. Specifically, the central authority $\mathcal{C}$ uses a structure-preserving signature key pair $(\mathsf{vk}_{\mathcal{C}}, \mathsf{sk}_{\mathcal{C}})$ to generate certificates on users' credentials. To enter the system, users register to this authority, which may require a computational or monetary cost to mitigate Sybil attacks. Note that this authority is required only for the registration of users, and possibly for the choice of accredited signers.

Clients have a structure-preserving signature key pair, consisting of a verification key $\mathsf{vk}_{\mathrm{Cl}}$ and a signing key $\mathsf{sk}_{\mathrm{Cl}}$. When clients enter the system, they register to the central authority $\mathcal{C}$ to get a random identifier $\mathsf{id}_{\mathrm{Cl}} \in \mathbb{Z}_p$, and a certificate $\mathsf{cert}_{\mathrm{Cl}}$ on $\mathsf{id}_{\mathrm{Cl}}$ and $\mathsf{vk}_{\mathrm{Cl}}$. Similarly, service providers have a structure-preserving signature key pair $(\mathsf{vk}_{\mathrm{SP}}, \mathsf{sk}_{\mathrm{SP}})$, and register to $\mathcal{C}$ to obtain a random identifier $\mathsf{Id}_{\mathrm{SP}} \in \mathbb{G}_1$, and a certificate $\mathsf{cert}_{\mathrm{SP}}$ on $\mathsf{Id}_{\mathrm{SP}}$ and $\mathsf{vk}_{\mathrm{SP}}$.

Accredited signers have a structure-preserving signature key pair $(\mathsf{vk}_{\mathrm{AS}}, \mathsf{sk}_{\mathrm{AS}})$ and a certificate $\mathsf{cert}_{\mathrm{AS}}$ on $\mathsf{vk}_{\mathrm{AS}}$. They use these keys to sign the reputation score of service providers at regular intervals, that we call *rounds*. We denote by σ_i the signature of the i-th accredited signer on the reputation score $\mathsf{rep}_{\mathrm{SP}}$ of the provider, for current round rnd, *i.e.* a signature on $\langle \mathsf{vk}_{\mathrm{SP}}, H(\mathsf{rep}_{\mathrm{SP}}, \mathsf{rnd}) \rangle$. In the following, n_{AS} represents the number of accredited signers. We assume that a majority t_{AS} of them are honest.

Share carriers possess two key pairs, namely a classical encryption key pair $(\mathsf{ek}_{\mathrm{SC}}, \mathsf{dk}_{\mathrm{SC}})$, and a classical signature key pair $(\mathsf{sk}_{\mathrm{SC}}, \mathsf{vk}_{\mathrm{SC}})$, used to encrypt received messages and sign sent messages. They also have a certificate $\mathsf{cert}_{\mathrm{SC}}$ on $\mathsf{ek}_{\mathrm{SC}}$ and $\mathsf{vk}_{\mathrm{SC}}$, issued by the central authority $\mathcal{C}$.

Both clients and providers compute by themselves their own pseudonyms. They renew them at each interaction. Pseudonyms $\mathsf{nym}_{\mathrm{Cl}}$ and $\mathsf{nym}_{\mathrm{SP}}$ are SXDH commitments to verification keys $\mathsf{vk}_{\mathrm{Cl}}$ and $\mathsf{vk}_{\mathrm{SP}}$. Similarly, both clients and service providers compute commitments $C_{\mathsf{id}_{\mathrm{Cl}}}$ and $C_{\mathsf{Id}_{\mathrm{SP}}}$ to their identifiers $\mathsf{id}_{\mathrm{Cl}}$ and $\mathsf{Id}_{\mathrm{SP}}$. Clients compute commitments $C_{\mathsf{cert}_{\mathrm{Cl}}}$ to their certificate, and NIZK proofs of their validity $\Pi_{\mathsf{cert}_{\mathrm{Cl}}}$. Similarly, service providers compute commitments $C_{\mathsf{cert}_{\mathrm{SP}}}$ and proofs $\Pi_{\mathsf{cert}_{\mathrm{Cl}}}$. Finally, service providers compute a pre-invariant $\mathsf{pre\_inv}$ from $\mathsf{Id}_{\mathrm{SP}}$ and a randomly chosen scalar $r_{\mathsf{pre\_inv}}$.

Due to space constraints, we defer the cryptographic proofs of the security of our protocol as well as figures detailing this protocol in a companion article [28].

5.1 Proof of the Reputation Score

When a client wishes to interact with a service provider, he sends a pseudonym $\mathsf{nym}_{\mathrm{Cl}}$ and a proof of its validity $C_{\mathsf{id}_{\mathrm{Cl}}}$, $C_{\mathsf{cert}_{\mathrm{Cl}}}$, and $\Pi_{\mathsf{cert}_{\mathrm{Cl}}}$ to the provider. Once the provider has verified this proof, she chooses a nonce s_{SC} and commits to it by computing $C_{\mathrm{SC}} = H(\mathsf{00} \| s_{\mathrm{SC}})$.[5] Then, the provider sends back her pseudonym, reputation, pre-invariant and respective proofs of validity, and committed nonce. That is, she sends $\mathsf{nym}_{\mathrm{SP}}$, $C_{\mathsf{Id}_{\mathrm{SP}}}$, $C_{\mathsf{cert}_{\mathrm{SP}}}$, $\Pi_{\mathsf{cert}_{\mathrm{SP}}}$, $\mathsf{rep}_{\mathrm{SP}}$, a proof of reputation Π_{rep},

[5] This concatenation guarantees that s_{SC} and r_{SC} are chosen independently.

$\mathsf{pre\_inv}$, a proof $\Pi_{\mathsf{pre\_inv}}$ of its computation while masking $\mathsf{Id}_{\mathrm{SP}}$ and $r_{\mathsf{pre\_inv}}$, and C_{SC}.

If the client is satisfied with the reputation of the provider, and if all the proofs are valid, the client computes the masked invariant $\mathsf{masked\_inv}$, chooses a nonce r_{SC}, computes a signature σ_{Cl} on $H(C_{\mathrm{SC}}, r_{\mathrm{SC}}, \mathsf{nym}_{\mathrm{SP}})$, and sends r_{SC} and σ_{Cl} to the provider. If σ_{Cl} is valid, the provider computes a signature on $H(s_{\mathrm{SC}}, r_{\mathrm{SC}}, \mathsf{nym}_{\mathrm{Cl}})$, and sends s_{SC} and σ_{SP} to the client. Note that the signatures guarantee that the client agreed to conduct a transaction with provider $\mathsf{nym}_{\mathrm{SP}}$, who uses the randomness hidden in C_{SC}, and that the provider agreed to conduct a transaction with client $\mathsf{nym}_{\mathrm{Cl}}$, who uses randomness r_{SC}. Once the client and the provider have exchanged their nonces, they choose the share carriers, using $(s_{\mathrm{SC}} \| r_{\mathrm{SC}} \| \mathsf{nym}_{\mathrm{Cl}} \| \mathsf{nym}_{\mathrm{SP}})$ as a seed. For that purpose, they iterate a hash function, *e.g.* SHA-256 [29], to randomly select n_{SC} share carriers among all service providers. In the remainder, this seed serves as an identifier of the transaction, and we note it $\mathsf{id}_{\mathsf{trans}}$.

During this step, the client sends one element in $\mathbb{Z}_p$, 86 in $\mathbb{G}_1$, and 74 in $\mathbb{G}_2$ to the provider, while the provider sends 3 element in $\mathbb{Z}_p$, $(74t_{\mathrm{AS}}+92)$ in $\mathbb{G}_1$, and $(66t_{\mathrm{AS}}+84)$ in $\mathbb{G}_2$. Once this step is over, besides being mutually authenticated, the provider has proven her reputation score to the client, each party is able to prove the implication of the other one in the interaction, and they finally have jointly and independently chosen the share carriers.

5.2 Sharing Ingredients of the Report

The client and the service provider now rely on the verifiable secret sharing scheme to guarantee the undeniability properties. The service provider shares her identifier $\mathsf{Id}_{\mathrm{SP}}$, that is, she chooses a polynomial Q of degree $t_{\mathrm{SC}} - 1$, with coefficients $\mathsf{Id}_{\mathrm{SP}}, A_1, \ldots, A_{t_{\mathrm{SC}}-1}$, where the A_j are randomly chosen in $\mathbb{G}_1$. The shares are the $\big(i, Q_i = Q(i)\big)$ for $1 \leqslant i \leqslant n_{\mathrm{SC}}$. To prove the sharing, the provider computes commitments C_{A_j} to the A_j, and NIZK proofs Π_{Q_i} that share Q_i was generated from $\mathsf{Id}_{\mathrm{SP}}$ and from the A_j for $1 \leqslant i \leqslant n_{\mathrm{SC}}$, while masking $\mathsf{Id}_{\mathrm{SP}}$ and the A_j. Note that $\mathsf{nym}_{\mathrm{SP}}$, $C_{\mathsf{Id}_{\mathrm{SP}}}$, $C_{\mathsf{cert}_{\mathrm{SP}}}$ and $\Pi_{\mathsf{cert}_{\mathrm{SP}}}$ have already proven the correctness of the secret, that is $\mathsf{Id}_{\mathrm{SP}}$. Finally, the provider sends the (C_{A_j}) to the client, and encrypts and sends $\mathsf{id}_{\mathsf{trans}}$, (i, Q_i), $C_{\mathsf{Id}_{\mathrm{SP}}}$, $(C_{A_j})_{1 \leqslant j < t_{\mathrm{SC}}}$, and Π_{Q_i} to the i-th share-carrier. If the received proof is valid, the share carriers send a confirmation to the client, that is $\mathsf{id}_{\mathsf{trans}}$, i, $C_{\mathsf{Id}_{\mathrm{SP}}}$, and (C_{A_j}), together with a signature. If these commitments are the same as the one received from the provider, the client accepts this confirmation: all the shares were generated from the same polynomial, which evaluates to the correct secret, $\mathsf{Id}_{\mathrm{SP}}$, in 0. Since the validity of the shares guarantees the undeniability properties, the client accepts the sharing once he has received $2t_{\mathrm{SC}} - 1$ valid shares. This requires for the provider to send $(2t_{\mathrm{SC}} - 2)$ elements in $\mathbb{G}_1$ to the client, and 4 in $\mathbb{Z}_p$, $(2t_{\mathrm{SC}} + 3)$ in $\mathbb{G}_1$, and 4 in $\mathbb{G}_2$ to each share carrier. Each share carrier sends 2 elements in $\mathbb{Z}_p$ and $2t_{\mathrm{SC}}$ in $\mathbb{G}_1$ to the provider.

In the meantime, the client shares his secret, that is the masked invariant $\mathsf{masked\_inv}$. Since $\mathsf{masked\_inv}$ consists of two elements, he must double the

sharing. That is, the client chooses two polynomial R_1, R_2 of degree $t_{\mathrm{SC}} - 1$ with coefficients $\mathsf{masked\_inv}_{\mathsf{k}}, B_{1,\mathsf{k}}, \ldots, B_{t_{\mathrm{SC}}-1,\mathsf{k}}$ for $\mathsf{k} \in \{1, 2\}$, and the shares are $\big(i, R_i = \big(R_1(i), R_2(i)\big)\big)$ for $1 \leqslant i \leqslant n_{\mathrm{SC}}$. To prove the sharing, the client computes commitments $C_{\mathsf{masked\_inv}}$ and $C_{B_{j,\mathsf{k}}}$ to $\mathsf{masked\_inv}$ and to the $B_{j,\mathsf{k}}$, and NIZK proofs Π_{R_i} that R_i was generated from $\mathsf{masked\_inv}$ and from the B_j for $1 \leqslant i \leqslant n_{\mathrm{SC}}$, while masking $\mathsf{masked\_inv}$ and the B_j. To prove the correctness of the secret, the client also computes a proof $\Pi_{C_{\mathsf{masked\_inv}}}$ guaranteeing the computation of $\mathsf{masked\_inv}$, while masking $\mathsf{masked\_inv}$, $\mathsf{id}_{\mathrm{Cl}}$, and the randomness used. Thus, the client sends $C_{\mathsf{masked\_inv}}$, $(C_{B_{j,\mathsf{k}}})$, and $\Pi_{C_{\mathsf{masked\_inv}}}$ to the provider, and encrypts and sends $\mathsf{id}_{\mathsf{trans}}$, (i, R_i), $C_{\mathsf{masked\_inv}}$, $(C_{B_{j,\mathsf{k}}})$, and Π_{R_i} to the i-th share carrier. As previously, the i-th share carrier sends a confirmation consisting of $\mathsf{id}_{\mathsf{trans}}$, i, $C_{\mathsf{masked\_inv}}$, $(C_{B_{j,\mathsf{k}}})$, and a signature to the provider if the share is valid. The provider accepts such a confirmation if the commitments are identical to the ones she received, and accepts the sharing as soon as she has received $2t_{\mathrm{SC}} - 1$ valid confirmations. Thus, the client sends one element in $\mathbb{Z}_p$, $(4t_{\mathrm{SC}} + 14)$ in $\mathbb{G}_1$ and 16 in $\mathbb{G}_2$ to the provider, and 2 in $\mathbb{Z}_p$, $(4t_{\mathrm{SC}} + 6)$ in $\mathbb{G}_1$ and 8 in $\mathbb{G}_2$ to each share carrier. Each share carrier sends 2 elements in $\mathbb{Z}_p$ and $4t_{\mathrm{SC}}$ in $\mathbb{G}_1$ to the provider. Once this step is over, the client is ensured that he will be able to obtain $\mathsf{Id}_{\mathrm{SP}}$ to issue the report. Similarly, the provider is guaranteed that she will be able to obtain a proof of transaction through the computation of $\mathsf{masked\_inv}$.Therefore, both parties can conduct their transaction.

5.3 Construction of the Reports

Once the transaction is over, the client can issue a rating and the provider can obtain a proof of transaction. Scenario A describes their interactions.

Scenario A – Nominal case. The client chooses a rating ρ and computes a signature $\sigma_{\rho,\mathrm{Cl}}$ on $H(\mathsf{id}_{\mathsf{trans}}, \rho)$ to prevent any modification on ρ, and a proof $\Pi_{\mathsf{masked\_inv}}$ of the computation of $\mathsf{masked\_inv}$, while masking $\mathsf{id}_{\mathrm{Cl}}$ and the randomness used. It is very important to note that the identity of the provider is preserved until the client issues and signs his rating, which fully preserves the objectivity of the rating. Once this is achieved, the provider can reveal her identity to the share carriers and even to the client. This allows the rating to be affected to the identity of the provider without allowing bad-mouthing attacks. Note also that by doing so, reputation scores reflect all the provider's interactions, not those conducted under a specific pseudonym. Since $\mathsf{masked\_inv}$ no longer needs to be hidden, $\Pi_{\mathsf{masked\_inv}}$ is a simpler proof than $\Pi_{C_{\mathsf{masked\_inv}}}$. The client sends message m_1 to the provider, with $m_1 = (\mathsf{id}_{\mathsf{trans}}, \rho, \mathsf{masked\_inv}, \Pi_{\mathsf{masked\_inv}}, \sigma_{\rho,\mathrm{Cl}})$. If both the proof and signature are valid, the provider computes the invariant inv from $\mathsf{masked\_inv}$ and $r_{\mathsf{pre\_inv}}$, and a signature $\sigma_{\rho,\mathrm{SP}}$ on $H(\mathsf{id}_{\mathsf{trans}}, \sigma_{\rho,\mathrm{Cl}})$. Note that since the provider reveals her identity, this signature is a structure-preserving signature, not an anonymous proxy signature. The provider then reveals her identifier, opens commitments $\mathsf{nym}_{\mathrm{SP}}$ and $C_{\mathsf{Id}_{\mathrm{SP}}}$, and reveals $r_{\mathsf{pre\_inv}}$. These proofs, denoted by Π_{SP}, guarantee both the computation of $\mathsf{pre\_inv}$ and that this provider is the one hidden behind $\mathsf{nym}_{\mathrm{SP}}$. The provider

Table 1. Components of the report in the three scenarii

	Scenario A	Scenario B	Scenario C
Provider	$\mathsf{Id}_{\mathrm{SP}}$, $\mathsf{vk}_{\mathrm{SP}}$, $\mathsf{cert}_{\mathrm{SP}}$, $\mathsf{nym}_{\mathrm{SP}}$, $C_{\mathsf{Id}_{\mathrm{SP}}}$, Π_{SP}	$\mathsf{Id}_{\mathrm{SP}}$, $\mathsf{vk}_{\mathrm{SP}}$, $\mathsf{cert}_{\mathrm{SP}}$, $\mathsf{nym}_{\mathrm{SP}}$, $C_{\mathsf{Id}_{\mathrm{SP}}}$, Π_{SP}	$C_{\mathsf{Id}_{\mathrm{SP}}}$, $\mathsf{nym}_{\mathrm{SP}}$, $\mathsf{PCert}_{\mathrm{SP}}$, $\mathsf{Id}_{\mathrm{SP}}$, $(C_{A_j})_j$, $\{i_j, Q_{i_j}, \Pi_{Q_{i_j}}\}_j$
Client	$C_{\mathsf{id}_{\mathrm{Cl}}}$, $\mathsf{nym}_{\mathrm{Cl}}$, $\mathsf{PCert}_{\mathrm{Cl}}$	$C_{\mathsf{id}_{\mathrm{Cl}}}$, $\mathsf{nym}_{\mathrm{Cl}}$, $\mathsf{PCert}_{\mathrm{Cl}}$	$C_{\mathsf{id}_{\mathrm{Cl}}}$, $\mathsf{nym}_{\mathrm{Cl}}$, $\mathsf{PCert}_{\mathrm{Cl}}$
Trans. id.	$\mathsf{id}_{\mathsf{trans}}, \sigma_{\mathrm{SP}}, \sigma_{\mathrm{Cl}}$	$\mathsf{id}_{\mathsf{trans}}, \sigma_{\mathrm{SP}}, \sigma_{\mathrm{Cl}}$	$\mathsf{id}_{\mathsf{trans}}, \sigma_{\mathrm{SP}}, \sigma_{\mathrm{Cl}}$
Invariant	$\mathsf{pre\_inv}$, $\mathsf{masked\_inv}$, inv, $r_{\mathsf{pre\_inv}}$, $\Pi_{\mathsf{masked\_inv}}$	$\mathsf{pre\_inv}$, $\mathsf{masked\_inv}$, inv, $r_{\mathsf{pre\_inv}}$, $(C_{B_{j,\mathsf{k}}})$, $\{i_j, R_{i_j}, \Pi_{R_{i_j}}\}_j$, $C_{\mathsf{masked\_inv}}$, $\Pi_{C_{\mathsf{masked\_inv}}}$	inv, Π_{inv}
Rating	ρ, $\sigma_{\rho,\mathrm{Cl}}$, $\sigma_{\rho,\mathrm{SP}}$		ρ, $\{\sigma_{\rho,\mathrm{SC}_{i_j}}\}_j$

sends message m_2 to the client, with $m_2 = (\mathsf{Id}_{\mathrm{SP}}, \mathsf{vk}_{\mathrm{SP}}, \mathsf{cert}_{\mathrm{SP}}, \mathsf{inv}, \Pi_{\mathrm{SP}}, \sigma_{\rho,\mathrm{SP}})$. The client verifies Π_{SP} and signature $\sigma_{\rho,\mathrm{SP}}$. Finally, both the client and the provider are able to issue the report by sending the elements given in the first column of Table 1 to the share carriers (where the first four lines represent the proof of transaction and the last one the rating together with the signatures of both parties). If all the signatures and proofs are valid, the report itself is considered valid by the share carriers. This scenario completes successfully if both parties are honest. If the client does not send message m_1 (resp. the provider does not send message m_2) then scenario B (resp. scenario C) is run. Finally, if neither the client nor the provider issue the report, then the transaction is not taken into account in the reputation score of the service provider. If this step proceeds correctly, the client sends 2 elements in $\mathbb{Z}_p$, 14 in $\mathbb{G}_1$, and 14 in $\mathbb{G}_2$ to the provider. Similarly, the provider sends 7 elements in $\mathbb{Z}_p$, 19 in $\mathbb{G}_1$, and 12 in $\mathbb{G}_2$. The report is composed of 11 elements in $\mathbb{Z}_p$, 143 in $\mathbb{G}_1$, and 116 in $\mathbb{G}_2$.

Scenario B – Dishonest client. If the provider does not receive message m_1 from the client, she queries the share carriers for their share by sending them $\mathsf{id}_{\mathsf{trans}}$. On their turn, they query the client to get his rating and, in absence of his answer, send their shares (i, R_i) and associated proofs Π_{R_i} to the provider. The provider is then able to reconstruct the masked invariant $\mathsf{masked\_inv}$ from t_{SC} valid received shares. From that point, the provider can compute inv from $\mathsf{masked\_inv}$ and $r_{\mathsf{pre\_inv}}$ and issue the report, which only contains the proof of transaction (*i.e.*, the elements in the second column of Table 1). During this step, the provider sends one element in $\mathbb{Z}_p$ to each share carrier, while each of them sends back to her one element in $\mathbb{Z}_p$, 6 in $\mathbb{G}_1$, and 8 in $\mathbb{G}_2$. The report is made of $(t_{\mathrm{SC}} + 10)$ elements in $\mathbb{Z}_p$, $(10t_{\mathrm{SC}} + 132)$ in $\mathbb{G}_1$, and $(8t_{\mathrm{SC}} + 108)$ in $\mathbb{G}_2$.

Scenario C – Dishonest provider. If the client does not receive message m_2 from the provider, he sends the masked invariant and his rating together with their associated proofs and signatures to the share carriers. That is, the client

sends $\mathsf{id}_{\mathsf{trans}}$, $\mathsf{nym}_{\mathrm{Cl}}$, $C_{\mathsf{id}_{\mathrm{Cl}}}$, $C_{\mathsf{cert}_{\mathrm{Cl}}}$, $\Pi_{\mathsf{cert}_{\mathrm{Cl}}}$, $\mathsf{nym}_{\mathrm{SP}}$, $C_{\mathsf{id}_{\mathrm{SP}}}$, $C_{\mathsf{cert}_{\mathrm{SP}}}$, $\Pi_{\mathsf{cert}_{\mathrm{SP}}}$, $\mathsf{pre\_inv}$, $\Pi_{\mathsf{pre\_inv}}$, $\mathsf{masked\_inv}$, $\Pi_{\mathsf{masked\_inv}}$, ρ, $\sigma_{\rho,\mathrm{Cl}}$. If all the proofs and signatures are valid, the share carriers forward them to the provider to give her the opportunity to reveal $\mathsf{Id}_{\mathrm{SP}}$ and the invariant. In absence of any response, the share carriers send their share (i, Q_i) and associated proof Π_{Q_i} to the client. Note that they also compute a signature $\sigma_{\rho,\mathrm{SC}_j}$ on $H(\mathsf{id}_{\mathsf{trans}}, \sigma_{\rho,\mathrm{Cl}})$ to validate the fact that the client has chosen his rating before knowing the provider's identity. Once the client has received t_{SC} valid shares, he reconstructs $\mathsf{Id}_{\mathrm{SP}}$, computes inv from $\mathsf{Id}_{\mathrm{SP}}$ and $\mathsf{id}_{\mathrm{Cl}}$, and computes a proof Π_{inv} of the computation of inv while masking $\mathsf{id}_{\mathrm{Cl}}$. Finally, the client issues the report by sending the elements presented on the third column of Table 1 to the share carriers. In this step, the client sends 4 elements in $\mathbb{Z}_p$, 202 in $\mathbb{G}_1$, and 178 in $\mathbb{G}_2$ to each share carrier. Each share carrier sends back one element in $\mathbb{Z}_p$, 3 in $\mathbb{G}_1$, and 4 in $\mathbb{G}_2$. Finally, the report is made of $(t_{\mathrm{SC}} + 4)$ elements in $\mathbb{Z}_p$, $(5t_{\mathrm{SC}} + 192)$ in $\mathbb{G}_1$, and $(4t_{\mathrm{SC}} + 164)$ in $\mathbb{G}_2$.

5.4 Computation of the Reputation Scores

At the end of round rnd, each share carrier gathers all the reports received since round $\mathsf{rnd} - 1$, and sends them to the accredited signers. This allows the accredited signers to update the reputation scores of all the service providers concerned by valid reports. Once accredited signers have checked the validity of a report, they only keep the identifier of the provider, the identifier of the transaction, the invariant inv, and the rating of the client, if any, and sign them. Note that if two (or more) reports have the same identifier of transaction and invariant, they keep a single one to avoid duplicates. Beyond handling negative ratings, the accredited signers know the rounds during which reports have been cast. Thus, as described in Section 2, any reputation score function can be used, *e.g.* to lower the influence of old ratings [14] or to limit the impact of ballot-stuffing attacks [22]. In addition, the accredited signers approximate the reputation score of providers to extend their anonymity set. Once the accredited signers have computed the reputation score of a provider, they compute a signature σ_i on $\langle \mathsf{vk}_{\mathrm{SP}}, H(\mathsf{rep}_{\mathrm{SP}}, \mathsf{rnd}) \rangle$ and send it to the provider. Service providers can use these signatures to prove their reputation to their clients during round $\mathsf{rnd} + 1$.

6 Performance Evaluation

We now evaluate our privacy-preserving reputation mechanism both in theoretical and practical ways. The former evaluation is achieved through an analysis of the performance of each building block, while the latter relies on its implementation on a platform made of heterogeneous computing nodes. The number of share carriers n_{SC} and the number of accredited signers n_{AS} are respectively equal to $n_{\mathrm{SC}} = 28$ and $n_{\mathrm{AS}} = 1$. This setting is sufficient to prevent the collusions of $\lceil n_{\mathrm{SC}}/3 \rceil - 1 = 9$ share carriers with probability 2^{-20} in a system comprising 10^8 service providers, including 5×10^6 malicious ones. This analysis, based on the hypergeometric distribution, appears in a companion paper [28].

Table 2. Size of exchanged messages for $n_{\mathrm{SC}} = 28$ and $n_{\mathrm{AS}} = 1$, in kibibytes

Phase	Cl $\leftrightarrow$ SP	Cl $\leftrightarrow$ SC_i	SP $\leftrightarrow$ SC_i	report
Proof of Reputation	22	0	0	—
Sharing	3.28	2.69	2.34	—
Scenario A	2.94	0	0	12.06
Scenario B	0	0	0.75	19.63
Scenario C	0	17.94	0	20.75

6.1 Theoretical Study

The correctness of our mechanism relies on the verification of NIZK proofs, which requires the computation of many pairings. To decrease the number of these operations, we adopt the technique proposed by Blazy *et al.* [30] which consists in verifying NIZKs by batches. We also ensure efficient pairing computations by relying on prime-order elliptic curves [31]. We consider elliptic curves in a subclass of the Barreto-Naehrig family. Thus, elements of $\mathbb{Z}_p$ and $\mathbb{G}_1$ (resp. $\mathbb{G}_2$) can be represented by 32 B (resp. 64 B). We use the computation costs given by Aranha *et al.* [31]. Namely, the four cores of a 3.0 GHz AMD Phenom II X4 940 processor – a top-level processor of 2010 – can compute 8 pairings in a millisecond, 16 exponentiations in $\mathbb{G}_2$, or 48 in $\mathbb{G}_1$. In the following, we study two metrics, namely (a) the size of messages exchanged between each entity, and (b) the time necessary for each entity to perform his computation. We now present and comment the main results obtained with these settings. Table 2 gives the size of messages (in KiB) exchanged between the different parties involved in the reputation mechanism, namely, between the client and the provider, the client and one share carrier, and the provider and one share carrier before the transaction takes place. Finally, it gives the size of the report sent to the accredited signer once the transaction is over.

These results are both satisfactory and reassuring. The largest messages correspond to the proof of reputation, which comprises the mutual authentication of the service provider and the client, and the proof by the provider of his reputation score. Nevertheless, this exchange requires only around 20 KiB. This is impressive compared to the mechanism proposed by Bethencourt *et al.* [5], where the proof of reputation requires 500 KiB per received rating. Table 2 also shows that share carriers only need 3 KiB when a transaction goes well, and less than 10 KiB in the worst case situation. This clearly shows that the design of a distributed trusted third party requires very little resources. The same comment applies for the accredited signers. The size of the report, that comprises all the proofs, requires no more than 20 KiB in the worst case. It is important to note that the only message that scales (linearly) as a function of the number of the accredited signers is the proof of reputation. Thus, even for larger sets of accredited signers, which typically do not grow to more than 20 entities, the communication cost remains acceptable. These results are very reassuring because they show that,

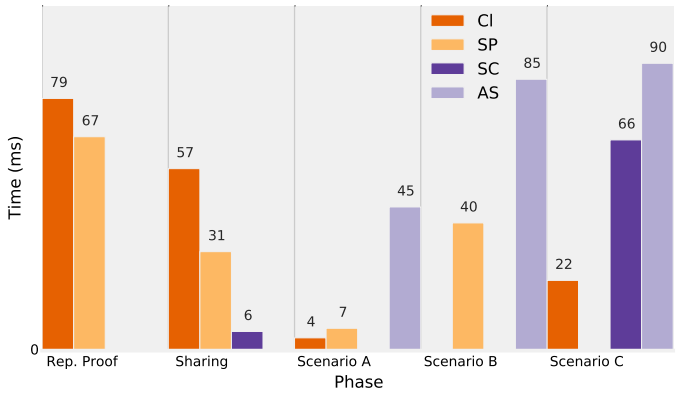

Fig. 1. Theoretical computation times (ms)

from a theoretical point of view, privacy-preserving reputation mechanisms are entirely viable. The next section will show that this holds in practice!

Figure 1 details the computation cost (in ms) of each phase of the reputation mechanism for each of the involved entities. Several remarks are in order. The main one is that computation times are very low. Indeed, each user needs no more than 200 ms for all their computations. In particular, each share carrier needs no more than 6 ms when both the client and the provider are honest. Even in the worst case, they need only 75 ms to perform their computations. Finally, the verification of a report requires between 45 ms and 90 ms. This clearly shows that participating to one of the two distributed trusted third parties computing entities costs little. Actually, the largest costs are due to scenarii B or C. We can minimize those costs by penalizing malicious users, *e.g.* by preventing them from interacting for a given period of time.

6.2 Implementing the Reputation Mechanism

We have implemented our reputation mechanism in Python 2.7 with the Charm framework [32]. This framework facilitates the implementation of complex cryptographic primitives, such as Groth-Sahai's NIZK proof system [25], and the combination of multiple primitives, *e.g.*, to build anonymous proxy signatures [26]. Furthermore, Charm provides the means to benchmark applications, both by giving their running time and by counting each elementary cryptographic operation. We also use Twisted, an event-driven networking engine, to handle communications between the different parties. Experiments have been conducted on heterogeneous entities, namely, a virtual machine running on a Dell Latitude E6430 laptop with a 2.60 GHz Core i7-3720 QM processor, and cheap Raspberry Pi model B machines with the Raspbian operating system.

Figure 2 presents the results of the conducted experiments. It shows the mean and standard deviation of the computation times of each user for every step of

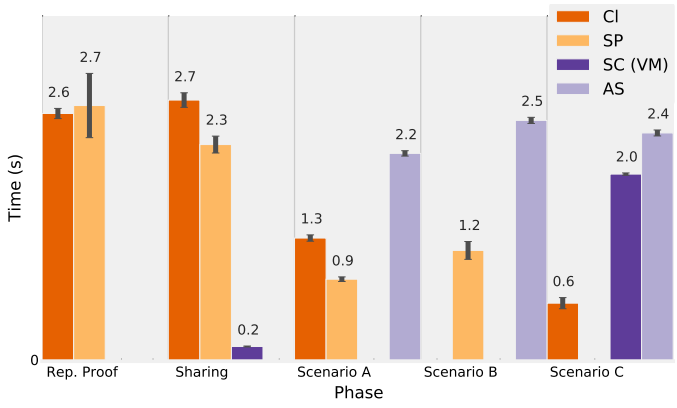

Fig. 2. Practical computation times (s)

the interaction, namely, the proof of reputation, the sharing, and the issuing of the report in every scenario. Note that the "SC" columns correspond to the computation times of one share carrier running on the virtual machine, and that the "AS" columns relate to the verification of one report by an accredited signer.

Clearly, the computation times are higher than the one obtained in theory, which can easily be explained. First, Aranha *et al.* carefully select a Barreto-Naehrig curve, and optimize the computation of pairings using Assembly and C code on this specific curve [31]. In our case, we rely on the MNT-159 curve proposed by Charm, which is a Python framework wrapping around Lynn's `pbc` library.[6] Furthermore, the theoretical number of operations per second assumes that they are all ran in parallel, which is not the case in our experiments. Finally, all the users except one share carrier were run on a single virtual machine. This does not slow down the phases where users run computations sequentially, *e.g.* the proof of reputation or the construction of the report in Scenario A, but it does slow down the concurrent ones, *e.g.* the sharing of the secrets.

Even with those limitations, we observe that our mechanism allows clients to interact with providers, and to run all the preparation in no more than 5 s. Issuing the report may take longer, but the most important point is that clients can rapidly verify the reputation of a provider and get involved in the transaction. Similarly, the pre-transaction and the post-transaction phases respectively require no more than 5 s and 1 s which clearly allows the provider to interact with many clients simultaneously. Note that share carriers can even be run on cheap Raspberry Pi machines. In that case, sharing the secrets requires no more than 4.7 s, while issuing the rating in presence of malicious clients needs no more than 59 s. Such cheap machines increase the waiting time of both clients and providers, but this delay remains acceptable (less than 15 s), compared for example to the time required to buy items on any e-commerce web sites. Finally, running clients on Raspberry Pi requires about 75 s for conducting the reputation

[6] http://crypto.stanford.edu/pbc/

proof and 115 s for the sharing. That is, clients need about 3 min before being able to conduct a transaction, which is clearly reasonable to engage in (possibly) financial transactions.

7 Conclusion

In this article, we have presented a privacy-preserving distributed reputation mechanism. Beyond being non-monotonic, this mechanism reveals to be fully usable even with cheap on-board computers. This is a very encouraging result as it shows that privacy does not impede utility and accuracy. This has been achieved by combining distributed algorithms and cryptographic schemes. Our mechanism is independent of the reputation model, that is, our system can integrate any reputation model [14], preferably one using both positive and negative ratings.

As future works, we plan to study more deeply an off-line version of the secret sharing, which requires the share carriers only in Scenarii B and C, and to improve the report verification when the service provider does not want to collaborate.

References

1. Dellarocas, C.: Immunizing online reputation reporting systems against unfair ratings and discriminatory behavior. In: ACM Conference on Electronic Commerce (EC), USA (2000)
2. European Parliament and Council of the European Union: Directive 95/46/EC. Official Journal of the European Communities (1995)
3. Narayanan, A., Shmatikov, V.: Robust de-anonymization of large sparse datasets. In: IEEE Symposium on Security and Privacy, Oakland,USA (2008)
4. Steinbrecher, S.: Enhancing multilateral security in and by reputation systems. In: The Future of Identity in the Information Society. (2008)
5. Bethencourt, J., Shi, E., Song, D.: Signatures of reputation. In: Financial Cryptography and Data Security (FC), Canary Islands (2010)
6. Pavlov, E., Rosenschein, J.S., Topol, Z.: Supporting privacy in decentralized additive reputation systems. In: International Conference on Trust Management. (2004)
7. Hasan, O., Brunie, L., Bertino, E.: Preserving privacy of feedback providers in decentralized reputation systems. Computers & Security (2012)
8. Kerschbaum, F.: A verifiable, centralized, coercion-free reputation system. In: Workshop on Privacy in the Electronic Society (WPES), Chicago,USA (2009)
9. Clauß, S., Schiffner, S., Kerschbaum, F.: *k*-anonymous reputation. In: Symposium on Information, Computer and Communications Security (ASIACCS). (2013)
10. Androulaki, E., Choi, S.G., Bellovin, S.M., Malkin, T.: Reputation systems for anonymous networks. In: Privacy Enhancing Technologies (PETS), Belgium (2008)
11. Baumeister, R.F., Bratslavsky, E., Finkenauer, C., Vohs, K.D.: Bad is stronger than good. Review of General Psychology (2001)
12. Lajoie-Mazenc, P., Anceaume, E., Guette, G., Sirvent, T., Viet Triem Tong, V.: Extending signatures of reputation. In: Privacy and Identity. IFIP AICT 421 (2014) 165–176

13. Resnick, P., Zeckhauser, R.: Trust among strangers in Internet transactions: Empirical analysis of eBay's reputation system. The Economics of the Internet and E-Commerce (2002)
14. Jøsang, A., Ismail, R.: The beta reputation system. In: Bled Electronic Commerce Conference, Bled, Slovenia (2002)
15. Yu, B., Singh, M.P.: Distributed reputation management for electronic commerce. Computational Intelligence (2002)
16. Kamvar, S.D., Schlosser, M.T., Garcia-Molina, H.: The EigenTrust algorithm for reputation management in P2P networks. In: International World Wide Web Conference (WWW). (2003)
17. Anceaume, E., Ravoaja, A.: Incentive-based robust reputation mechanism for P2P services. In: International Conference on Principles of Distributed Systems (OPODIS), Bordeaux, France, Springer Berlin Heidelberg (2006)
18. Hasan, O., Brunie, L., Bertino, E., Shang, N.: A decentralized privacy preserving reputation protocol for the malicious adversarial model. IEEE Transactions on Information Forensics and Security (2013)
19. Jøsang, A., Ismail, R., Boyd, C.: A survey of trust and reputation systems for online service provision. Decision Support Systems (2007)
20. Marti, S., Garcia-Molina, H.: Taxonomy of trust: Categorizing P2P reputation systems. Computer Networks (2006)
21. Pfitzmann, A., Hansen, M.: A terminology for talking about privacy by data minimization: Anonymity, unlinkability, undetectability, unobservability, pseudonymity, and identity management (2010)
22. Whitby, A., Jøsang, A., Indulska, J.: Filtering out unfair ratings in bayesian reputation systems. In: International Workshop on Trust in Agent Societies. (2004)
23. Dingledine, R., Mathewson, N., Syverson, P.: Tor: The second-generation onion router. In: USENIX Security Symposium. (2004)
24. Feldman, P.: A practical scheme for non-interactive verifiable secret sharing. In: Foundations of Computer Science (FOCS), USA (1987)
25. Groth, J., Sahai, A.: Efficient non-interactive proof systems for bilinear groups. In: Eurocrypt, Istanbul, Turkey, Springer Berlin Heidelberg (2008)
26. Fuchsbauer, G., Pointcheval, D.: Anonymous proxy signatures. In: Security and Cryptography for Networks (SCN). (2008)
27. Abe, M., Fuchsbauer, G., Groth, J., Haralambiev, K., Ohkubo, M.: Structure-preserving signatures and commitments to group elements. In: Advances in Cryptology—CRYPTO. (2010)
28. Lajoie-Mazenc, P., Anceaume, E., Guette, G., Sirvent, T., Viet Triem Tong, V.: Efficient distributed privacy-preserving reputation mechanism handling non-monotonic ratings. Technical report, https://hal.archives-ouvertes.fr/hal-01104837 (2015)
29. National Institute of Standards and Technology: Secure hash standard (2012)
30. Blazy, O., Fuchsbauer, G., Izabachène, M., Jambert, A., Sibert, H., Vergnaud, D.: Batch groth-sahai. In: Applied Cryptography and Network Security (ACNS), Beijing, China, Springer Berlin Heidelberg (2010)
31. Aranha, D.F., Karabina, K., Longa, P., Gebotys, C.H., López, J.: Faster explicit formulas for computing pairings over ordinary curves. In: Eurocrypt. (2011)
32. Akinyele, J.A., Garman, C., Miers, I., Pagano, M.W., Rushanan, M., Green, M., Rubin, A.D.: Charm: a framework for rapidly prototyping cryptosystems. Journal of Cryptographic Engineering **3** (2013) 111–128