Street Rep: A Privacy-Preserving Reputation Aggregation System

Christophe Hauser[1], Shirin Nilizadeh[2], Yan Shoshitaishvili[3], Ni Trieu[3], Srivatsan Ravi[4], Christopher Kruegel[5], and Giovanni Vigna[5]

[1] Dartmouth College
[2] University of Texas at Arlington
[3] Arizona State University
[4] University of Southern California
[5] University of California Santa Barbara

Abstract. Over the last decade, online reputation has become a central aspect of our digital lives. Most online services and communities assign a reputation score to users, based on feedback from other users about various criteria such as how reliable, helpful, or knowledgeable a person is. While many online services compute reputation based on the same set of such criteria, users currently do not have the ability to use their reputation scores *across* services. As a result, users face trouble establishing themselves on new services or trusting each other on services that do not support reputation tracking. Existing systems that aggregate reputation scores, unfortunately, provide no guarantee in terms of user privacy, and their use makes user accounts *linkable*. Such a lack of privacy may result in embarrassment, or worse, place users in danger.
In this paper, we present *Street Rep*, a practical system for aggregating user reputation scores in a privacy-preserving manner. *Street Rep* makes it possible for users to provide their aggregated scores over multiple services *without* revealing their respective identities on each service. We discuss our novel approach for tamper-proof privacy preserving score aggregation from multiple sources by combining existing techniques such as blind signatures, homomorphic signatures and private information retrieval. We discuss its practicality and resiliency against different types of attacks. We also built a prototype implementation of *Street Rep*. Our evaluation demonstrates that *Street Rep* (a) performs efficiently and (b) practically scales to a large user base.

1 Introduction

However, designing such a unified reputation system for aggregating users' reputation scores arises serious privacy implications. Even though users might wish to share reputation across different platforms, they may not want their different accounts (even different accounts on the same services, as in the case of Reddit "throwaway" accounts) to be linkable. For example, someone could have two Reddit accounts, one for general participation and another for participating in a forum of *e.g.,* abuse survivors, and not want her two worlds to collide. Likewise,

someone trying to find something on Craigslist might not want strangers on the Internet to be able to link their Craigslist post to their Airbnb account and recover their address. In fact, considering the sensitivity of the links between these accounts, it is reasonable for users of the system to require it to preserve the unlinkability of their accounts even from the entity running the system itself. Thus, any sort of centralized reputation system must be strongly *privacy-preserving*. Additionally, a reputation system must be resistant to *tampering* with or *cherry-picking* of the scores. In the context of the privacy preserving requirement, these characteristics are non-trivial to achieve.

We propose *Street Rep*, a novel design for a privacy-preserving centralized reputation system that facilitates the linking of reputation between different accounts (whether on the same or different platform) while preserving the unlinkability of the accounts themselves and being resistant to reputation tampering and cherry-picking. *Street Rep* allows its users to compute their own aggregated scores, for all or a subset of their *identities*, in a tamper-proof manner, without revealing the underlying identities themselves. By identities, we refer to the set of individual accounts that a user might have on different online services, which the user would like to include in the computation of their aggregated score. The resulting aggregated score is analogous to a credit score in the sense that it provides an "arbitrary" measure of trust. In the United States, a credit score is a number generated by a mathematical algorithm using information in a person's credit report, obtained from many different sources. Similarly, the aggregated reputation score in *Street Rep* is computed using a person's reputation ratings based on her online activities. The key difference is that, in *Street Rep*, the central server that collects the individual reputation scores for a user's accounts does not learn any information about the real identity of users, or which accounts belong to the same user. This further holds for any querier (*i.e.,* other users querying a given user for her score). Users of *Street Rep* aggregate their reputation ratings themselves by following a cryptographic protocol. While this *self-computation* is a key aspect to the anonymity and integrity properties ensured by *Street Rep*, delegating the reputation aggregation to users involves several challenges, as users may attempt to forge a higher score than their own by using malicious techniques in order to tamper with the computation. For example, they may attempt to omit their bad ratings, or use expired ratings with higher scores. *Street Rep* addresses the problem of tamper-proof privacy preserving score aggregation by combining existing techniques such as blind signatures, homomorphic signatures and Private Information Retrieval (PIR) in a novel way.

Our contributions are the following. (i) We designed a protocol allowing for the computation of an aggregated score, corresponding to multiple services, without disclosing individual scores or identities. (ii) We built a novel reputation aggregation system preventing any party from linking individual users' identities used in the computation, based on the aforementioned protocol. (iii) We implemented *Street Rep* as a prototype, and demonstrated that it performs efficiently and practically scales to a large user base.

To the best of our knowledge, we are not aware of any privacy-preserving reputation aggregation system that combines *anonymity*, *account unlinkability*, *integrity and unforgeability of reputation scores*, *score freshness* (preventing users from using old reputation scores) in a practical and scalable protocol as constructed in this paper.

2 System overview

In this section, we will provide an overview of *Street Rep*, before delving deeper throughout the rest of the paper.

Example use case As a motivating example, consider the following use case. An individual, Alex, has accounts on several different online services. She might have a Slashdot account for technical discussions, an eBay account used for shopping, and a Reddit account with which she participates in sensitive discussions (for example, on the `/r/survivorsofabuse` *sub-reddit*; a community focused on helping abuse survivors). All of these activities generate reputation: Alex might get moderated on Slashdot, voted on Reddit, and reviewed on eBay. Due to the sensitivity of the involvement on Reddit, however, Alex may wish to keep her Reddit account unaffiliated with her other accounts.

The reputation information accumulated by Alex on these services represents, in some sense, the reputation of "Alex the Person", rather than just various accounts that she controls. However, due to the desire for anonymity, she cannot utilize a traditional centralized reputation system to "unify" this reputation, as she would have to publicly link the Slashdot, eBay, and Reddit accounts, losing anonymity on the latter site. Using *Street Rep*, Alex can pool the reputation of these accounts while maintaining the Reddit account's anonymity, allowing for reputation ratings built up on those accounts to form the reputation of "Alex the Person".

Street Rep would be used in several steps involving Alex, the service providers (Reddit, Slashdot, and eBay), the central server (that manages the reputations in a privacy-preserving way), and the querier who will eventually query Alex's combined rating.

High-level description From a high-level perspective, *Street Rep* involves the following steps and operations. Alex first *commits* to provide scores from a given set of services to *Street Rep*. She initially chooses which services to incorporate, as part of a public *commitment set*. Each value in this set represents a different account that she owns. She then *blinds* these values using a cryptographic operation based on Chaum's blind signatures [5], which produces so-called *blind commitment values* that do not reveal any information about the original values (accounts) from the commitment set. After this, Alex registers the blind commitment values with each service provider (one blind commitment value per service provider). The service providers (SPs) update the central server (CS) with the reputation values associated with these blind commitments. The CS essentially acts as a database of entries of the form $\epsilon :< i, r >$ where i is an identity, and

r is a rating[6]. The central server generates signatures on combination of each of her scores with its corresponding blind commitment value. Next, Alex privately queries CS for the signatures on her scores without revealing her identities using private information retrieval mechanisms. Then she privately computes her aggregated score *herself* [7], and provides the result to the querier. This step is both tamper proof and privacy perserving, and relies on a novel insight involving blind signature transformations with respect to the (multiplicative) homomorphic properties of RSA. This aspect is described in detail in § 3.3. After this process, the querier will have obtained an aggregated reputation score for Alex, including her reputation from Reddit, eBay, and Slashdot. Neither the querier, the service providers, nor the central reputation server will be able to link Alex's Reddit account to the other two accounts. Moreover, the querier is able to verify integrity of an aggregated reputation score, making sure that Alex has not modified or cherry picked her reputation ratings.

Threat model We assume that SPs and the CS are *honest-but-curious*, *i.e.,* they follow the protocol honestly and run the protocol exactly as specified (no deviations, malicious or otherwise), but may try to learn as much as possible about the users and their accounts. This is a standard assumption in the literature, and in our particular case, the underlying intuition is that SPs and the CS have no particular interest in tampering with user scores. However, we assume that users may be malicious, and therefore are not trusted with the integrity of their aggregated reputation scores.

In summary, we assume that: SPs and the CS *do not* provide incorrect reputation ratings for individual accounts; SPs, queriers, and the CS may collude with each other as an attempt to de-anonymize users, or link their identities together; Users may try to forge arbitrary ratings, share individual ratings with other users, or omit their bad scores and cherry-pick their best scores, as an attempt to increase their own aggregated score. Users *do not* try to increase their reputation ratings by colluding with the CS, or with the SPs; The mapping between an *individual* account and its associated reputation score, or rating, is public[8].

Security and privacy requirements Here, we list the security and privacy requirements of a privacy-preserving reputation aggregation system.

- **Anonymity**: the underlying identity, or *real* identify of accounts is not revealed to the SPs nor to the CS. They only know the *apparent* identity associated with each account (*e.g.,* nicknames). Clearly, *Street Rep* does not provide anonymity for users whose identifiable information is explicitly or implicitly attached to their accounts.
- **Account unlinkability**: accounts of the same user cannot be linked together: none of the involved parties learns which accounts' reputation ratings

[6] In the remainder of this paper, we interchangeably use the terms *rating* and *reputation score*

[7] This process is automated, and is performed on the client-side.

[8] Users are responsible for using nicknames or non-identifying pseudonyms on their public profiles, *e.g.,* on Reddit.

Table 1: Reputation Aggregation Schemes.

Scheme	System and Threat model	Anonymity	Account Unlinkability	Integrity & Unforgeability	Score freshness
Street Rep	SPs push reputation ratings to CS, users obtain aggregated reputation score from CS ; Assumes that SPs and the CS are honest-but-curious.	yes	yes	yes	yes
SOR [4]	Only supports monotonic aggregation/measures of reputation; Requires a trusted registration authority	yes	no	no	no
DECO [19]	Users access websites via TLS to prove provenance; Secure against both malicious users and servers	yes	yes	no	Not applicable
Prio [8]	Designs for a traditional secure aggregation; requires multiple non-colluding servers, and at least one honest	yes	(for clients)	no	Not applicable

are used in the computation of an aggregated score. Users solely know their own set of accounts, and this information remains private during the whole process.

- **Integrity and unforgeability of reputation scores**: users cannot report manipulated (increased) individual ratings; the ratings remain identical to what is published by SPs in their websites. They also cannot forge any parameter in the computation of their own score, nor use another user's scores.
- **Score freshness**: the overall aggregated reputation score is fresh and the user cannot use her old and higher reputation ratings in the computation.

Related work and its limitations In contrast with existing work in the literature, our approach meets *all* of these requirements. In order to do so, *Street Rep* presents an efficient and practical scheme inspired by existing cryptographic techniques from the literature. It builds on a novel combination of blind signatures, homomorphic encryption, and private information retrieval. More specifically, our approach leverages the multiplicative homomorphic properties of RSA along with Chaum's blind signatures in order to build a system featuring the aforementioned security and privacy requirements. To the best of our knowledge, existing schemes proposed in the literature only address a subset of the security and privacy requirements presented in our current approach:

- Existing approaches based on anonymous credentials or secure aggregation do not provide all privacy and security requirements (e.g, unlikability, score freshness) in a way that would be usable in the context of our work.
- Current schemes based on zero knowledge proofs involve a considerably higher performance impact than the scheme we propose.

We choose the most relevant work and compare it with Street Rep in Table 1.

Bethencourt et al. [4] proposed *Signatures of Reputation (SOR)* as a new cryptographic framework that makes it possible to have a messaging system where anonymous users post unlinkable messages while the votes on their posts get aggregated. These aggregated votes are shown along with the posts as the reputations of their authors. In *Street Rep*, however, the reputation ratings are generated by users of several different websites and then they are aggregated. The SOR only supports *monotonic* aggregation of reputation where additional feedback *cannot* decrease a user's reputation. However, in *Street Rep*, there is no constraint on reputation scores and they may decrease or increase. In addition,

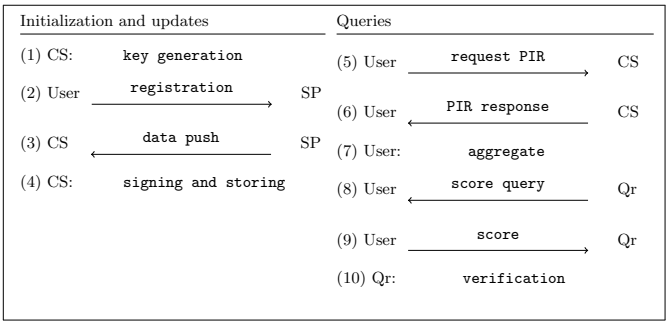


Fig. 1: Steps in *Street Rep*

in *Street Rep*, we provide mechanisms for preventing users from omitting or replacing their bad scores to forge a good reputation.

It might be possible to apply some techniques used in anonymous credential system to provide account unlinkability. For example, Zhang *et al.* [19] recently introduce DECO, a decentralized oracle for liberating web data. DECO system consists of 3 parties: prover, verifier, and server. The goal is to allow the (malicious) prover to prove the verifier that a piece of data came from a particular website (the server) and prove statements about such data while keeping the data itself secret. Indeed, DECO can be used to support reputation aggregation. Concretely, one can consider DECO's prover as a user in Street Rep, DECO's servers as our service providers (SP), and DECO's verifier as our centralized server (CS). The Street Rep's user now needs to prove CS that he has a reputation score from the respective i^{th} service provider. They can simply do it by invoking DECO. However, their system does not provide all privacy and security requirements for a reputation aggregation system. For example, a user can still cherry-pick some scores and provide the credentials only for higher reputation scores.

Using secret-shared non-interactive proofs, Prio [8] is able to prevent malicious users who can follow any arbitrary polynomial-time strategy to deviate from the scheme. One can use secure aggregation to aggregate reputation scores of an individual from different service providers. However, this solution is clearly vulnerable to user's privacy (e.g. anonymity) since the centralized server has to know the real/underlying identify of account to determine the reputation scores from the same user.

3 System design

The core component of *Street Rep* is a protocol allowing users to compute and share aggregated scores in a privacy-preserving manner.

Online service providers, or SPs, provide a service involving a reputation score, such as discussion forums, or e-commerce services. We define the set of all SPs as $\mathcal{S}p = \{sp_1, sp_2, ..., sp_k\}$. **Users** of *Street Rep* are identities who gain

reputation by participating in online activities. $\mathcal{U}$ is the set of all users involved in the system, where: $\mathcal{U} = \{u_1, u_2, ..., u_n\}$. **Identities**. Each user has a variable number of identities (or accounts) spread across multiple SPs. The set of all the identities appearing in these services is defined as $\mathcal{I} = \{i_1, i_2, ..., i_m\}$. There can be a one-to-many mapping between a user and her set of identities. **The central server**, or CS, regularly obtains the reputation ratings for all identities on all services. It then signs and stores these ratings. Later, upon requests, it provides these signatures to users through private information retrieval. **Queriers** are third-party clients (such as a seller, buyer, renter, driver or a recruiter) who directly contact and request *Street Rep*' users for their updated overall reputation score.

As shown in Figure 1, players interact with each other using a protocol consisting of ten steps: (1) initialization, (2) registration, (3) pushing data to the central server, (4) signing and storing, (5, 6) privately retrieving data from the central server, (7) signature aggregation, (8) querying for a user's reputation score, (9) sending the reputation and its signature to the querier, and (10) verification by the querier. The remainder of this section describes each step in more detail.

3.1 Initialization

The first phase of *Street Rep* consists of its initialization, that is, setting up cryptographic keys, and registering user accounts. This corresponds to steps (1) and (2) as described above.

Key generation At *initialization* time, the CS follows the RSA key generation phase and generates two pairs of keys, (pk_1, sk_1) and (pk_2, sk_2), each consisting of a public key and a secret key respectively.

It chooses two pairs of large prime numbers (p, q) and (p', q'), and computes $n = p \cdot q$, $n' = p' \cdot q'$, $\phi(n) = (p-1)(q-1)$ and $\phi(n') = (p'-1)(q'-1)$, where $\phi(n)$ is Euler's totient function, and $n \neq n'$ and $\phi(n) \neq \phi(n')$. It then chooses two encryption keys e and e' and calculates d and d' such that $e \cdot d \equiv 1 \pmod{\phi(n)}$ and $e' \cdot d' \equiv 1 \pmod{\phi(n')}$. The CS finally publishes $pk_1 = (e, n)$ and $pk_2 = (e', n')$ as its public keys and keeps $sk_1 = (d, p, q)$ and $sk_2 = (d', p', q')$ secretly.

In the remainder of this paper, we denote by: $enc(e, m) = m^e \pmod{n}$, the encryption of message m using the public key e, and its decryption using the private key d, such that: $dec(d, enc(e, m)) = m$. Similarly, we denote by $sig(d, m)$ and $sigverif(e, m)$ the signature of message m using the private key d, and its verification using the public key e, respectively.

Registration Figure 2 illustrates the steps for a user u with m accounts to follow and register in the *Street Rep* system. They include *User commitment*, *Hiding*, and *Registration request*. We denote `user_SPs` as a set of service providers

User u

1 : **foreach** (i, sp_i) **in** (`user_accounts`, `user_SPs`) **do**
2 : $c_i =$ `request_commitment_value()`
3 : $r_i =$ `random()`
4 : $b_i =$ `blind(`c_i`,` r_i`)`
5 : `register(`$sp_i, < i, b_i >$`)`
6 : **done**
7 : $pro_u = \{(c_i, sp_{c_i})\}$
8 : `publish_profile(`pro_u`)`

Fig. 2: Algorithm for user registration

on which u has some accounts and `user_accounts` as a set of accounts for the user.

User commitment. Each user of the system initially commits to always using the same defined set of accounts when computing her aggregated score. This allows for aggregated scores to be consistent over time, and it also prevents users from cherry-picking high scores while leaving low scores out of the computation. In order to commit to a given set of accounts, users associate a *unique* identifier to each of their underlying identities. This unique identifier, which we term *commitment value* is, in itself, public. However, the mapping between identities and commitment values remains private, and only known by the user.

In order to ensure that each commitment value is unique, users obtain each value by querying the CS, which maintains a database of allocated commitment values (this query is represented as `request_commitment_value()` in the algorithm shown in Figure 2). As a result, each user u, with m accounts, commits to a *commitment set* $\mathcal{C}_u$ as follows:

$$\mathcal{C}_u = \{c_1, \cdots, c_m\}$$

The user creates a public profile, pro_u, and stores it in the CS. This profile is a list of pairs, where each pair is composed of an identifier (corresponding to a service provider), and a commitment value.

$$pro_u = \{(c_1, sp_1), \cdots, (c_m, sp_m)\}$$

Therefore, anyone including the queriers is able to know which services host the accounts of a given user[9]. This information can be used by queriers to ask for the aggregation of scores from a subset of the user's accounts. For example, a querier may only request the aggregation of scores in the category of finance and marketing. More information regarding *specific queries* is provided in § 3.5.

[9] However, the specific accounts at play remain private, and it is therefore impossible for an attacker to know that *e.g.,* a given user participated in the `/r/survivorsofabuse` *sub-reddit.*

Hiding. Along with each commitment value c_i, users compute a *blind token* b_i as follows:

$$b_i = c_i \cdot {r_i}^e \pmod{n}$$

where r_i is a large random value that is kept secret by the user, and (e, n) is the public key of the CS, as part of the pk_1 pair. Along with r_i, users also compute ${r_i}^{-1}$ such that:

$$r_i \cdot {r_i}^{-1} \equiv 1 \pmod{n}$$

In Figure 2, we call this function `blind(`c_i`,` r_i`):` $\mathcal{I} \rightarrow \mathcal{B}$, that returns the blind commitment value associated with a user account. Blind tokens essentially embed a commitment value, without revealing it, and are signed later by the CS to produce a *blind signature*. These allow users to interact with other parties without revealing the link between their underlying identities, while being able to prove that they comply with their commitment (this aspect is described in further detail in § 3.3). We denote by $\mathcal{B}$ the set of all blind commitment values associated with all identities.

Registration request. Users register accounts of their choice for use in *Street Rep* by sending a registration request to their respective SPs, sp_i, as follows:

$$reg_i =< i, b_i >$$

Each SP is responsible for authenticating its own users over secure channels. Once a service provider knows the blind token associated with an identity, it considers this account as being *registered* for use in *Street Rep*, and it proceeds to push its associated scores into the CS at each round.

3.2 Updates

While the system is running, user accounts are constantly changing due to online user activity, affecting the user scores on the SPs' side. In order to keep up to date with the latest scores, *Street Rep* requires SPs to update scores on a regular basis. This is achieved through the concept of *rounds*, where SPs push data related to each of their registered user accounts, and the CS signs and stores it.

During this process, SPs are considered trustworthy with respect to the integrity of the data, *i.e.,* SPs have no interest in tampering with users' scores.

Pushing data SPs interact with the CS by pushing fresh scores on a regular basis. Different service providers may use different scales for ratings (*e.g.,* one to five stars, or a percentage). In *Street Rep*, the scores are all normalized to the same scale before being pushed. Each SP is responsible for implementing its own `normalize()` function, which normalizes scores of its own scale to Street Rep's scale in which scores are represented in the discrete integer scale $[1, \mu]$.

During this normalization process, SPs add some noise to the ratings, *e.g.,* scores are *rounded* to the closest integer value, where the loss of accuracy is justified by a gain in privacy. Moreover, SPs can apply other sophisticated differential privacy techniques [10]. We consider these improvements for future

work. In § 4.3, we argue that when μ is small (e.g., 5), it is impractical to uniquely identify users based solely on their individual reputation ratings and their aggregated scores.

At each round k, a new time stamp t_k is generated, and a *storage request* is sent to the CS. A storage request is performed by sending to the CS a vector $sto_k =< i, t_k, \nu_i, b_i >$, where ν_i is the normalized score associated with the identity i and b_i is a *blind* user commitment. The corresponding algorithm is described in Figure 3, in which we introduce the function `score:` $\mathcal{I} \rightarrow \mathcal{S}$ that returns the score value associated with a user identity.

SP

At time t_k
foreach i **in** `user_accounts`, **do**
  $\nu_i =$ `normalize(score(i))`
  $m = (i, t_k, \nu_i, b_i)$
  `send`(CS, m)
done

Fig. 3: Algorithm for storage requests.

The communication between SPs and CS, and between SPs and users operates on a secure channel, and SPs are trusted to provide correct scores.

Signing and storing As it is shown in Figure 4, the central server stores information on service providers' storage requests. After receiving a storage request, the CS creates a new entry in its database, of the form $< i, \sigma, \sigma' >$, along with a time stamp t, such that:

$$\sigma = sig(d, \nu_i \cdot b_i \cdot t) = (\nu_i \cdot t \cdot c_i \cdot {r_i}^e)^d \pmod{n}$$

and:

$$\sigma' = sig(d', \nu_i) = {\nu_i}^{d'} \pmod{n'}$$

where i is the index of the entry. Each entry contains the signatures corresponding to an account. The values d and d' are the private keys of sk_1 and sk_2 (the two key pairs initially generated by CS in §3.1), and the values n and n' are part of the public keys, pk_1 and pk_2 respectively.

It should be emphasized that, since RSA is homomorphic with respect to multiplication, σ can also be expressed as

$$\sigma = sig(d, \nu_i) \cdot sig(d, b_i) \cdot sig(d, t) \pmod{n}$$

and, thus, by signing b_i, the CS generates a *blind signature* over the blind message b_i.

CS

foreach (i, t_k, ν_i, b_i)**in** `receive`(sto_k) **do**
  $\sigma_i = sig((d, \nu_i \cdot t \cdot b_i))$
  $\sigma'_i = sig(d', \nu_i)$
  `store`$[i] = (\sigma_i, \sigma'_i)$
done

Fig. 4: Algorithm for signing and storing.

Both σ and σ' are then used in the aggregation and verification steps, as described in the next sections. As a result, σ is the signature of the product of a reputation score, its associated blind commitment value, and a time stamp. It is used by users when aggregating scores.

While σ' is the signature of the reputation score alone, the use of signatures, σ and σ', has a tamper-proof role, as described in § 4, they both together ensure the integrity of overall aggregated reputation score that will be reported by the user.

3.3 Queries

At each round k, users retrieve their latest individual reputation ratings (and their signatures) from the CS, and aggregate them together. This operation guarantees that users respect their initial commitments (otherwise, the resulting signatures are invalid, as discussed in §4. The results include aggregated reputation ratings and their signatures, which later can be provided to queriers upon a request.

Privately retrieving data First, introduced by Chor et al. [6], Private Information Retrieval (PIR) is a protocol that allows a user to query a database for an item without revealing which item is being retrieved.

The CS database leverages PIR primitives, each including r blocks of b bits in size. Users of *Street Rep* know the exact database indexes of the blocks where their accounts are located.

In *Street Rep*, we use a lattice-based CPIR scheme called XPIR [1], in which one PIR server is employed, in contrast to IT-PIR schemes where the data is split among multiple servers. It has been shown that XPIR can process a wide range of databases in a few seconds, even for $100Gb$ databases [1].

In *Street Rep*, users do *privately* retrieve the signatures related to each of their reputation ratings. Since each request is based on PIR, the CS is unable to gather information about the content being retrieved, and is thus unable to infer any link between individual user accounts during this process. For each of their accounts, users retrieve signatures σ and σ' from the CS, as shown in Figure 5.

User

foreach i **in**$\{i_1, \ldots, i_m\}$ **do**
  `request_pir`(i)
  $(\sigma_i, \sigma_i') =$ `pir_response()`
done

Fig. 5: Private information retrieval in Street Rep

Most PIR schemes, including XPIR, do not ensure database confidentiality. In *Street Rep*, malicious users may attempt to leverage this to obtain information belonging to other user accounts, without the CS being aware of it. To this end, a dishonest user may try to usurp the signatures of other users' reputation ratings. However, we show in § A.2 that this attack vector is not applicable.

User aggregation After the CS has signed the scores for individual identities, the user aggregates these signatures, and generates an aggregated signature for her aggregated score. This aggregated signature ensures that the score is not forged by the user, and that she has not cherry-picked a subset of her accounts (*i.e.,* those exposing the best scores) to compute the overall score.

To aggregate her signatures, a user first unblinds them. Recall from § 3.1 that values for r_i and r_i^{-1} are kept secret by the user. The signatures retrieved by the user from the CS are of the form:

$$\sigma_i = (\nu_i \cdot t \cdot c_i \cdot r_i^e)^d \pmod{n}$$

As ${r_i}^{ed} = r_i \pmod{n}$, this is equivalent to:

$$\sigma_i = (\nu_i \cdot t \cdot c_i)^d \cdot r_i \pmod{n}$$

From there, since $r_i \cdot {r_i}^{-1} \equiv 1 \pmod{n}$, and since r_i and ${r_i}^{-1}$ are kept secret, the user is the only player that has the ability to perform the following operation:

$$\sigma_i \cdot {r_i}^{-1} = (\nu_i \cdot t \cdot c_i)^d \pmod{n}$$

This operation allows users to *unblind* the commitment values embedded into each signature σ_i. In Figure 6, this function is called `unblind`. In order to aggregate scores, users calculate the product of all of their m unblinded signatures as follows:

$$p = \prod_{i=1}^{m} \sigma_i \cdot {r_i}^{-1} \pmod{n}$$

which is equivalent to:

$$p = \prod_{i=1}^{m} (\nu_i \cdot t \cdot c_i)^d \pmod{n}$$

Users also compute the product of their reputation scores signed with the private key sk_2 and represented by σ', as follows:

$$p' = \prod_{i=1}^{m} \sigma'_i \pmod{n'}$$

User

foreach σ_i **in** `pir_responses()` **do**

$\hat{\sigma}_i =$ `unblind`(σ_i)

done

$$p = \prod_{i=1}^{m} \hat{\sigma}_i \pmod{n}$$

$$p' = \prod_{i=1}^{m} \sigma'_i \pmod{n'}$$

Fig. 6: Algorithm for aggregating signatures.

At any time during round k, a querier may ask a user to provide an aggregated score. This query can be sent by any medium, and can be automated, for example, in person, by email or automatically through a trusted service or application providing online presence (this point is outside of the scope of this current work). At the time of query, the user may be offline or online. As a response, when available, the user sends ν, p and p' to the querier, where $\nu = \prod_{i=1}^{m} \nu_i$

Verification A querier interested in a user's overall reputation score, sends a request to her using the function `request_user()`. At this point, the querier received v, p and p' from the user. By using the CS's public key, the querier verifies both signatures using the `verify()` function, which computes

$$p^e \pmod{n} = \prod_{i=1}^{m} (\nu_i \cdot t \cdot c_i) = \prod_{i=1}^{m} \nu_i \cdot \prod_{i=1}^{m} t \cdot \prod_{i=1}^{m} c_i \pmod{n},$$

and:

$$p'^{e'} \pmod{n'} = \prod_{i=1}^{m} \nu_i \pmod{n'}.$$

If both signatures are accepted by the verification function, the querier can ensure that the score is the aggregation score of all the user's accounts, and that the user has not forged any signature. In § 4, we elaborate these properties in more detail.

Querier

```
request_user()
(ν, p, p′) = receive_user()
if verify(ν, p, p′, t, c1, ··· , cm) then
  _accept_
else
  _reject_
done
```

Fig. 7: Algorithm for verifying signatures.

Aggregated reputation score As mentioned previously, the commitment set for a user u, C_u, is public in her profile. Therefore, the querier can trivially compute $m = |C_u|$. Also, the user provides ν to the querier, thus, she can infer the final reputation score by computing the geometric mean of the aggregated score as follows:

$$f = \nu^{(1/m)}$$

This geometric mean indicates the central tendency of a user's reputation scores and shows a single "figure of merit" ; a good indicator of one's reputation over multiple services. A querier can use this final reputation score as a metric to evaluate the trustfulness or worthiness of a person compared to others.

3.4 Commitment updates

Over time some users may register into new websites while leaving a few others. *Street Rep* supports the functionality for users to add new accounts. They need to register on these new websites and update their profiles. Then, they can use their new reputation ratings to compute their aggregated reputation score.

However, *Street Rep* currently does not allow users to remove accounts from their profiles. This intentional limitation prevents users to "cheat" by removing accounts as they reach lower scores. As a results, users' bad reputation ratings are always used in calculating the overall reputation score.

3.5 Specific queries

In some scenarios, queriers may see more value in the aggregation of scores from a specific subset of services on which the user has an account. For example, a recruiter may be more interested in one's reputation on websites that are more related to job search. Recall that users publish their profiles, containing a list of services on which they have accounts (along with their committed values). This allows queriers to generate specific queries targeting a subset of accounts, which we term *cluster*. Possible impacts of specific queries on user's privacy are discussed in Section 6.

4 Security analysis

We present hereafter an analysis of the security properties of *Street Rep*, demonstrate how these are practical under realistic assumptions, and demonstrate the impracticality of subsequent vectors of attack.

4.1 Fundamental properties

By design, the RSA cryptosystem is prone to *multiplicative homomorphism*. *Street Rep* relies on these homomorphic properties in order to perform arithmetic operations over cryptographic signatures. If performed naively, such operations are naturally subject to *signature forgery*. In fact, one could argue that, in the general case, such a scheme is by definition insecure; RSA is only homomorphic when used *without padding*, and as such:

1. Small input values (cleartext m) or encryption exponents (e) resulting in $m^e < n$ are by definition not subject to modulo reduction and can therefore be easily decrypted by an attacker.
2. A consequence of small encryption exponents is that attacks (based on the Chinese remainder theorem) exist when a number of recipients of a ciphertext share the same exponent [13,7].
3. It is not semantically secure [11], that is, ciphertexts are subject to plaintext chosen attacks. (This is however not specific to RSA, as any trapdoor function is existentially subject to such attacks [12]).

For these reasons, general applications of RSA always rely on secure padding schemes, such as OAEP defined in the PKCS#1 standard [14]. One of the effects of using such padding schemes is that of rendering the resulting cryptosystem non-homomorphic.

In contrast with general approaches, *Street Rep* leverages the homomorphic properties of RSA in the context of a well-defined message space, where such potential vectors of attack are mitigated by the following points:

- The input space in *Street Rep* is both large and of a *fixed* size: values of an input cleartext m depend on well defined *multiplicative expressions of large integer numbers* (that is, t, ν and b from § 3). Therefore, all operations involved as part of our protocol are subject to modulo reduction, and weaknesses *1)* and *2)* stated above do not apply in this context.
- Each multiplicative expression in *Street Rep* includes a blind commitment value, itself generated from a large input space of *pseudorandom values*. A consequence of this is that the message space is sparse, which has the effect of making plaintext chosen attacks impractical [12]. This mitigates weakness *2)* stated above.
- One of the fundamental properties of RSA is that of claw-freeness. This property renders *adaptive plaintext chosen attacks* impractical. We will come back to this point in § A.2. This mitigates weakness *3)* stated above.

In summary, per the fundamental properties of RSA and given the aforementioned properties of size and sparsity of the message space intrinsic to our scheme, the previous general classes of attacks are impractical in the context of

Street Rep. The remainder of this section presents i) a more detailed discussion of the attack surface implied by the homomorphic properties of RSA, and how our restrictive assumptions render such attacks impractical within the scope of our application; and ii) a detailed discussion of specific points of importance with respect to the security of our protocol.

4.2 Anonymity

The CS does not infer the real identities of accounts because users retrieve their information from the CS's database through PIR. The SPs do not obtain additional information about users of *Street Rep*. Since commitment values are blinded by users, the SPs are not able to detect if an identity in a website belongs to a specific user. The only way for an adversary to be successful is to unblind the commitment values, which is equivalent to the problem of inverting RSA. Obviously, SPs might have already collected some data about their users. However, real identities of their users can remain private if users have not revealed them in their profiles/ accounts.

4.3 Account unlinkability

Account unlinkability is an important property, since the probability of deanonymizing users, that is, revealing their real identity, increases as an attacker recovers the link between individual user accounts. The following section discusses how *Street Rep* provides unlinkability. More specifically, we discuss the impracticality for an attacker to *link* multiple individual accounts together. We assume that users are responsible for using secure and anonymous channels during communications with the service providers (*e.g.,* users may need to rely on Tor to avoid cooperating service providers to link identities by matching IP addresses or perform intersection attacks based on the churn rates of their users [18]).

General level of unlinkability The probability to obtain a random set of k identities corresponding to a given user u with k identities is $pr = \frac{1}{C_m^k}$, where $m = |\mathcal{I}|$. This probability corresponds to the most general applicable attack to *Street Rep*, and is very low. Furthermore, increasing k reduces this probability (as it increases C_m^k).

We refer to $\mathcal{L} = 1 - pr$ by *level of unlinkability*, which empirically corresponds to the difficulty for an attacker to detect the mapping between a given user and her identities. $\mathcal{L}$ corresponds to the maximum level of unlinkability that *Street Rep* can provide for a user with k identities. Here, we show that our protocol does not further degrade this level of unlinkability.

Linking identities The following demonstrates the difficulty for an attacker to link users' accounts, at each relevant step of the protocol.

- **Registration phase:** during the registration phase, users communicate their blind tokens to the service providers. First, it is not practical for an attacker to recover the initial commitment values from the blinded values generated by RSA without both knowing the private key of sp_1 and the modular multiplicative inverse of r_i for identity i, introduced in § 3. Second, a different random number r is used in generating every blind token, thus blind tokens are random, independent of users and accounts.
- **Signing phase**: during this phase, the CS gathers a blind token of the form $b_i = c_i \cdot {r_i}^e \pmod{n}$ (where c_i is a unique commitment value from the user's public profile, and r_i is a secret random number known by the user only) and signs it using its private key d. Since b was blinded using the CS's public key e, this signing operation is equivalent to a decryption operation, and the CS is able to learn the value of $c_i \cdot r \pmod{n}$ in the process. However, since r is both secret and very large, the CS cannot infer anything about the value of c_i.
- **Retrieving data**: The use of PIR by users when retrieving signatures from the CS ensures that no information is revealed about the retrieved content, and thus no link can be made between accounts or signature entries during this phase.
- **Verification phase**: By obtaining aggregated signatures of users, queriers do not gather additional information about user identities, other than their aggregated score.

Linking scores During the verification phase of *Street Rep*, the querier learns the product of scores, or ν. On a system where users each have k identities, with scores normalized within the scale $[1, \mu]$, the minimum and maximum values for ν are 1 and μ^k, respectively. Some values between these two numbers occur with a higher probability than others. For example, prime numbers larger than μ can never occur, while values with lots of small prime factors occur more often. As a result, for example, score products with large prime factors are more identifying than those with lots of small prime factors.

Therefore, the success of this attack depends on the number k of accounts per user, and on the size μ of the space used to represent scores. Empirically, when using realistic values for k and μ, many users share the same aggregated score. These users with the same product score create an "anonymity set." This anonymity set provides plausible deniability for its users and it is very difficult for an adversary to be able to uniquely de-anonymize a user. For instance, as a simple illustrative example, let us consider a system with 10,000 users using a scale from 1 to $\mu = 5$ in order to aggregate the scores of $k = 5$ accounts each. If we first assume a uniform distribution of users ratings, then each unique score product is shared by 384 unique users, and this scales in a linear fashion (*i.e.,* on a system with 1 million users, each unique product is shared by 38400 users, and so on). In practice, this uniformity assumption is unlikely to hold, and related work has shown that the overall distribution of user rating roughly falls into a normal bell-shaped distribution [15], which (in comparison with a uniformly

distributed dataset) affects the probability of very low and very high scores. We argue, however, that this is problematic only if the scale of valid scores is large enough that some score values do actually fall under a specific probability threshold (e.g. 1 out of 1000 users). In other terms, the fact that the querier learns the product ν in this case does not reveal the public identity of the user being queried, except when large values of ν are used, or when the number of users in the system is low.

Finally, note that score freshness cannot be manipulated: a dishonest user may try to use an old reputation score and its signature to increase her overall reputation score. However, the signatures created by the CS include a timestamp and by using an old signature, the overall signature will be rejected by the verifier (querier).

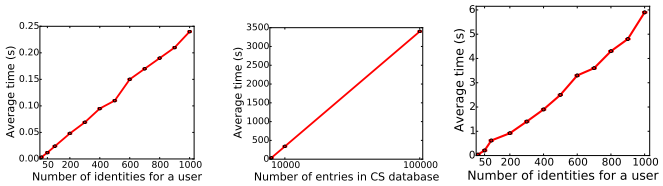
5 Evaluation

We performed the following experiments in order to evaluate the performance of *Street Rep* in terms of latency. For each party, *i.e.,* users, CS, SPs and queriers, we measured the overhead of computing the cryptographic functions and retrieving data from database using Private Information Retrieval (PIR). We have developed a prototype of *Street Rep* in Python. Our prototype leverages XPIR [2] for PIR capabilities, as well as PyCrypto [16] as a basis for the implementation of our cryptographic primitives.

In our experiments, we assumed that 1000 SPs are available and that users can only have one account per SP.Users are assigned a random number of accounts, up to 1000 accounts per user. The experiments are run on a regular PC featuring an Intel Core i7 X980 @ 3.33GHz, and 8 GB of RAM.

Overhead of cryptographic functions In order to register, a user calculates her blind commitment values (as described in §3.1), and sends them to corresponding SPs. Figure 8(a) shows the average latency of computing the blind commitments for a user with k accounts. Each point in this figure is the average latency for 100 users with a specific number of accounts. It shows that, even with 1000 accounts per user, the average overhead on the system is about 0.24 seconds (std_err=0.004), and with less than 10 accounts per user, the latency decreases to 0.003 seconds.

Figure 8(b) shows the overhead of the cryptographic operations performed by CS to compute two signatures for each account, where the total number of accounts is 1000, 10,000 and 100,000 respectively. Each point on the plot shows the results for 10 databases with the same size. For a database with 100,000 accounts, the average latency is about 3,400 seconds, that is, a little less than an hour (std_err=15). Figure 8(b) indicates a linear relationship between the size of database and the latency. Thus, if the total number of accounts is 1 million then it takes about 9.4 hours to compute the signatures. Note that the signing and storing phase can be done in parallel by multiple processes. Also, CS only signs users' data once at every round of the protocol. Depending on the system's requirements, scores may get updated every day/week/month.



(a) User registration (b) Signing the ratings by the CS (c) Signature aggregation by users

Fig. 8: Overhead of cryptographic operations: at user registration phase, on average, a user with 1000 accounts only spends about 0.2 seconds to compute blind signatures. The CS computes two signatures for each of 100,000 accounts in less than an hour. Users aggregate their (2* 1000) rating signatures in about 6 seconds. Finally, queriers verify aggregated signatures in 0.0005 seconds.

Figure 8(c) shows the overhead of cryptographic operations for a user to aggregate her reputation ratings. The results are the average latency for 100 users with the same number of accounts. For a user with 1000 accounts, the average overhead for unblinding and aggregating her signatures is about 5.9 seconds (std_err=0.1). The latency decreases to less than 0.04 seconds when the user has less than 10 accounts.

We also performed experiments to measure the time needed for a querier to verify the overall reputation score. On average it is 0.0005 seconds (std_err=4.41E-5). This latency is independent of number of accounts for a user.

Overhead of PIR Figure 9(a) shows the average time needed for users with k accounts to query and retrieve their data from the PIR server. In this experiment, the CS database includes data (*i.e.,* the two signatures) for total of 100,000 accounts. Figure 9(a) shows that users with 10 and 200 accounts respectively on average spend about 0.02 seconds (std_err=0.0002) and 0.57 seconds (std_err=0.04) to retrieve their data from the CS. Recall that if an user has m accounts, he/she needs to make m PIR requests to the centralized server CS. By using multi-query PIR [3], we expect that the performance of Street Rep can achieve up to 40× faster over processing queries one at a time.

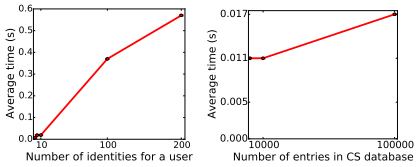


Fig. 9: PIR Overhead: A user privately retrieves her 200 accounts from the CS database in about 0.6 seconds. The size of database has a very small impact (about 0.006 seconds) on the retrieval time.

In Figure 9(b), we set $k = 10$ to be a constant value and vary the database size from 1000 to 100,000. Every data point here indicates the average results of 10 different databases with the same size. This shows that the database size only slightly impacts the latency of PIR operations. In particular, users with 10 accounts retrieve their data from the CS with 1000 and 100000 elements in 0.011 seconds (std_err=$7.7e-05$) and 0.017 seconds (std_err=0.0002) respectively.

6 Discussion

In this section, we discuss about a few properties and limitations of our approach.
Overhead on the SPs: Street Rep imposes little overhead on the SPs. They are only required to normalize the scores before sending them to the CS for updates. Integrating this normalization process as part of the API provided to the SPs is trivial. SPs, in return, are expected to push updates on a regular interval, which is standard in many service platforms (*e.g.,* commonly automated with schedulers like Unix Cron).
Aggregation function: In *Street Rep*, the aggregated signatures are computed using the geometrical mean over reputation ratings. For future work, we will investigate the possibility of computing complex functions.
De-activation of accounts: As previously described in § 3.4, users in *Street Rep* are not allowed to remove any of their accounts from their profiles. In some situations, however, users may genuinely and legitimately not be interested in using a particular online service anymore. A simple solution may reside in the ability for users to move some of their accounts to an "inactive" stash, while notifying queriers that the corresponding commitment values should be excluded from the computation.
Anonymous credentials: One question that might arise is: *"Why not use anonymous credentials?"* With anonymous credentials, users obtain a distinct credential for each account, and credentials are provided to queriers individually. This makes it trivial for an adversary (especially the SPs) to identify a user and link her accounts just from the observation of her scores. In our system, no one, including the SP or CS, should be able to link *any* account to a user's identity. This link must remain private at all times. Furthermore, anonymous credentials rely on zero-knowledge proof protocols, requiring the user and the querier be online and participate in the protocol at the same time. Our approach with Street-Rep avoids this limitation.
Concluding remarks The privacy-preserving functionality that *Street Rep* provides is not available from any previously-existing model and opens the door for the creation of a privacy-preserving Reputation Bank, as proposed by futurists and technology thinkers [17].

References

1. Aguilar-Melchor, C., Barrier, J., Fousse, L., Killijian, M.O.: Xpir: Private information retrieval for everyone. In: Proceedings on Privacy Enhancing Technologies. vol. 2016, pp. 155–174 (2016)

2. Aguilar-Melchor, C., Barrier, J., Fousse, L., Killijian, M.O.: Xpir: Private information retrieval for everyone. https://github.com/XPIR-team/XPIR (2016)
3. Angel, S., Chen, H., Laine, K., Setty, S.: Pir with compressed queries and amortized query processing. In: 2018 IEEE Symposium on Security and Privacy (SP). pp. 962–979 (2018). https://doi.org/10.1109/SP.2018.00062
4. Bethencourt, J., Shi, E., Song, D.: Signatures of reputation. In: Financial Cryptography and Data Security, pp. 400–407. Springer (2010)
5. Chaum, D.: Blind signature system. In: Advances in cryptology. pp. 153–153. Springer (1984)
6. Chor, B., Kushilevitz, E., Goldreich, O., Sudan, M.: Private information retrieval. Journal of the ACM (JACM) **45**(6), 965–981 (1998)
7. Coppersmith, D.: Small solutions to polynomial equations, and low exponent rsa vulnerabilities. Journal of Cryptology **10**(4), 233–260 (Sep 1997). https://doi.org/10.1007/s001459900030, https://doi.org/10.1007/s001459900030
8. Corrigan-Gibbs, H., Boneh, D.: Prio: Private, robust, and scalable computation of aggregate statistics. In: Proceedings of the 14th USENIX Conference on Networked Systems Design and Implementation. p. 259–282. NSDI'17, USENIX Association, USA (2017)
9. Dodis, Y., Reyzin, L.: On the power of claw-free permutations. In: SCN. vol. 2, pp. 55–73. Springer (2002)
10. Dwork, C.: Differential privacy. In: Automata, languages and programming, pp. 1–12. Springer (2006)
11. Goldwasser, S., Micali, S.: Probabilistic encryption & how to play mental poker keeping secret all partial information. In: Proceedings of the Fourteenth Annual ACM Symposium on Theory of Computing. pp. 365–377. STOC '82, ACM, New York, NY, USA (1982). https://doi.org/10.1145/800070.802212, http://doi.acm.org/10.1145/800070.802212
12. Goldwasser, S., Micali, S., Rivest, R.L.: A" paradoxical" solution to the signature problem. In: Foundations of Computer Science, 1984. 25th Annual Symposium on. pp. 441–448. IEEE (1984)
13. Hastad, J.: N Using RSA with Low Exponent in a Public Key Network, pp. 403–408. Springer Berlin Heidelberg, Berlin, Heidelberg (1986). https://doi.org/10.1007/3-540-39799-X$_2$9, https://doi.org/10.1007/3-540-39799-X_29
14. Jonsson, J., Moriarty, K., Kaliski, B., Rusch, A.: Pkcs# 1: Rsa cryptography specifications version 2.2 (2016)
15. Lin, Z., Li, D., Janamanchi, B., Huang, W.: Reputation distribution and consumer-to-consumer online auction market structure: an exploratory study. Decision Support Systems **41**(2), 435–448 (2006)
16. Litzenberger, D.: Python cryptography toolkit. https://github.com/dlitz/pycrypto (2016)
17. Stross, C.: Accelerando. Ace (2005)
18. Wolinsky, D.I., Syta, E., Ford, B.: Hang with your buddies to resist intersection attacks. In: Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security. pp. 1153–1166. ACM, New York, NY, USA (2013)
19. Zhang, F., Maram, D., Malvai, H., Goldfeder, S., Juels, A.: Deco: Liberating web data using decentralized oracles for tls. In: Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security. p. 1919–1938. CCS '20, Association for Computing Machinery, New York,

NY, USA (2020). https://doi.org/10.1145/3372297.3417239, https://doi.org/10.1145/3372297.3417239

A Integrity and unforgeability

In this section, we discuss the impracticality for a user to increase her own reputation score by *cherry-picking*, *forging* or *usurping* signatures of reputation scores.

A.1 Cherry picking

The following demonstrates the impracticality for users to cherry pick commitment values. A dishonest user, u, may try to only pick the signatures related to her highest reputation scores, and discard signatures related to lower scores. Formally, the user attempts to use a *subset* $\hat{\mathcal{I}}$ exhibiting her k best scores. We denote by $\hat{\mathcal{V}}$ the set of these scores, and by $\hat{C}$ the corresponding commitment values. During the computation of the aggregated signature, the user attempts the following signature aggregation:

$$\hat{\sigma} = sig(d, \prod_{i=1}^{k} \hat{\nu}_i \cdot \hat{c}_i \cdot t)$$

where $\hat{\nu}_i \in \hat{\mathcal{V}}, \hat{c}_i \in \hat{C}$ and t is a valid fresh time stamp.

However, this aggregated signature would be rejected by the querier during the verification phase, because the product of the cherry-picked scores does not match the product of the user's public commitment values: $\prod_{i=1}^{k} \hat{c}_i \neq \prod_{i=1}^{k} c_i$. It should also be emphasized that, while users may modify their public commitment set over time, *Street Rep* only lets them include additional accounts, but existing ones cannot be removed. Thus, the product of commitment values cannot be adjusted to match a subset of cherry-picked accounts.

A.2 Preventing Usurping of Signatures

The following demonstrates the inability for a malicious user to usurp signatures by replacing or mixing existing signatures.

In an attempt to increase their own score, users may also try to *usurp* other users' signatures, that is, claim those as their own and use them during the computation of their aggregated scores. Since users are all able to query the CS for any entry, they may attempt to replace or aggregate some chosen signatures to forge a signature for a higher reputation score. There are two possible scenarios described below.

Replacing signatures Recall that the link between user commitment values and identities is hidden by the fact that users employ *blind tokens* to interact with the system. Thus, during the aggregation step of the protocol, users *unblind* part of the message using the (secret) value r^{-1} as described in § 3. Signatures stored in the CS are of the form:

$$\sigma_{cs} = sig(d, \nu_i \cdot t \cdot b_i)$$

where b_i is a blind token. However, these signatures, once unblinded by the user, are of the form:

$$\sigma_u = sig(d, \nu_i \cdot t \cdot c_i)$$

where c_i is a (unblinded) commitment value. The latter is used by users during the aggregation phase, but in fact, there is no way for the querier to know if a given signature (or more precisely, a product of signatures that they are getting from the user) contains blinded or unblinded values. As a result, dishonest users may attempt to swap one of their unblinded signatures, σ_u (that exhibits a low score) with a random signature $\hat{\sigma}$ obtained from the CS (that possibly exhibits a higher score). Since users cannot unblind signatures for which they do not know the associated secret value r^{-1}, they may attempt to *skip* the unblinding phase for such signatures. This may remain unnoticed by the querier if the user can find a signature $\hat{\sigma}$ along with a score $\hat{\nu}$ such that:

$$\hat{\sigma} = sig(d, \nu \cdot t \cdot b_i) = sig(d, \hat{\nu} \cdot t \cdot c_i)$$

and that the forged score $\hat{\nu}$ fits in the scale $[1, \mu]$ and is better than the original (low) score ν: $\nu \leq \hat{\nu} \leq \mu$. And also, by definition: $1 \leq \hat{\nu} = \frac{\nu * b_i}{c_i} \leq \mu$.

However, the probability for an attacker to find such constraints is *negligible*, because the range of possible values for any random $\hat{\nu}$ that the user may pick from the CS is very large in comparison to $[1, \mu]$: the domain of commitment values is $[0, 2^{32} - 1]$ while the domain of blind commitment values is $[0, 2^{2048} - 1]$. Thus, the range of $\hat{\nu}$ is $[0, 2^{2016} \cdot \mu]$ and the probability that $\hat{\nu}$ hits the space of scores is $\frac{1}{2^{2016}}$, which is negligible. Therefore, the probability that a randomly picked signature from the CS yields a valid score while matching a valid commitment value is negligible.

Mixing signatures. The adversary may generalize the previous attack and replace some of her signatures with several signatures from the CS. For instance, if the dishonest user has k identities, she may try to replace her signatures with m signatures where $k \leq m$, such that the forged score $\hat{\nu}$ satisfies:

$$1 \leq \hat{\nu} = \frac{\prod_{i=1}^{m} \nu_i \cdot t \cdot b_i}{\prod_{j=1}^{k} \nu_j \cdot t \cdot c_j} \leq \mu^k \quad \text{and} \quad \hat{\nu} > \prod_{j=1}^{k} \nu_j.$$

If such $\hat{\nu}$ exists, then the user can claim that her aggregated score is equal to $\hat{\nu}$, and forge a signature:

$$\hat{\sigma} = sig(d, \prod_{i=1}^{m} \nu_i \cdot t \cdot b_i) \quad \text{as it equals to} \quad sig(d, \hat{\nu} \cdot \prod_{j=1}^{k} t \cdot c_j).$$

However, the probability for an attacker to find such signatures in practice is negligible, for the same reasons as previously: The probability that $\hat{\nu}$ hits the space of scores is even smaller than that in case 1. The maximum range of $\hat{\nu}$

depends on m and k and it is roughly in the order of $\frac{2^{2048*m}}{2^{32*k}}$. Thus the probability that $\hat{\nu}$ hits the space of scores is $\frac{1}{2^{2048*m-32*k}}$. This probability is very small, and even if $m == k$, it is $\frac{1}{2^{2016*k}}$. Therefore, the probability that a user can use a signature as of hers is negligible.

Note that if $k > m$, then there is no benefits for the user in forging these signatures, since it diminishes the overall score (by adding one more term in the computation of the mean, see § 3.3).

Multiplying random numbers In the aggregation phase, a dishonest user u with k identities may multiply the aggregated signatures, p and p', by random numbers x_1 and x_2 to forge signatures, $\hat{p}$ and $\hat{p'}$, in order to obtain a higher overall score $\hat{\nu}$ such that:

$$\hat{\nu} = \prod_{j=1}^{k} \nu_j * x_2.$$

To succeed, one needs to find x and y such that:

$$x^e \pmod{n} \equiv y^{e'} \pmod{n'} \equiv z$$

This attack is impractical since, as demonstrated by Dodis *et al.* [9], the claw-freeness property of RSA renders adaptive chosen plaintext attacks extremely difficult. In this particular case, finding a claw (x, z) implies $y = x.z^e(modn)$, which implies inverting RSA on a random input y.