A New Approach to Modelling Centralised Reputation Systems

Lydia Garms and Elizabeth A. Quaglia

Information Security Group, Royal Holloway University of London
{Lydia.Garms.2015, Elizabeth.Quaglia}@rhul.ac.uk

Abstract. A reputation system assigns a user or item a reputation value which can be used to evaluate trustworthiness. Blömer, Juhnke and Kolb in 2015, and Kaafarani, Katsumata and Solomon in 2018, gave formal models for *centralised* reputation systems, which rely on a central server and are widely used by service providers such as AirBnB, Uber and Amazon. In these models, reputation values are given to items, instead of users. We advocate a need for shift in how reputation systems are modelled, whereby reputation values are given to users, instead of items, and each user has unlinkable items that other users can give feedback on, contributing to their reputation value. This setting is not captured by the previous models, and we argue it captures more realistically the functionality and security requirements of a reputation system. We provide definitions for this new model, and give a construction from standard primitives, proving it satisfies these security requirements. We show that there is a low efficiency cost for this new functionality.

1 Introduction

Reputation has always played a fundamental role in how we exchange products and services. While traditionally we have been used to trusting the reputation of established brands or companies, we are now facing a new challenge in the online world: determining the trustworthiness of a wide variety of possible exchanges. Whether we are selecting a restaurant, buying a product or getting a taxi, we are increasingly relying on scores and ratings to make our choice. For example, on Amazon, which in 2015 had over 2 million third party sellers worldwide [1], each seller is given a rating out of 5. Also Uber, with over 40 million monthly active users [2], allows drivers and passengers to rate each other.

A *reputation system* formalises this process of rating a user or service by associating with them a value representing their trustworthiness. A reputation is then built as the value gets updated over time, as a consequence of user interactions and service exchanges. Obviously, to form a reputation value for a specific user or service, their behaviour across interactions needs to be linked, but this may have privacy implications. For instance, a user could be deanonymised by linking all their interactions together in a profiling attack.

Given this, a cryptographic treatment of reputation systems has been considered necessary, and several models have been proposed in the literature so far [3,9,19]. Reputation Systems can be generally categorised into *distributed* or

centralised systems. Distributed systems [24] have no central server and use local reputation values, i.e., reputation values created by users on other users. For example, a user may generate a reputation value based on feedback from querying other users, and their own interactions. This means a user does not have a unique reputation value, but many other users hold their own reputation value for them. For example privacy preserving decentralised reputation systems [26] are designed to maintain anonymity when answering queries from other nodes.

Centralised systems, on the other hand, have a central server that manages the network, performing tasks such as controlling communication between users, receiving feedback and evaluating reputation values. In this paper, we will focus on centralised systems since the reputation systems used by most service providers such as Airbnb, Uber and Amazon are of this type. A variety of centralised reputation system models and instantiations have been proposed in the literature, as we shall see in Section 2. While their applications and the used techniques vary greatly amongst them, all of the models have in common that reputations are assigned to each *item* or service, the object of the reputation, rather than each *user*, the provider of the service. To understand the limitations of this, let us consider the case of online shopping: in such a scenario, existing reputation systems would typically allocate a reputation to each product sold (item), and not to each seller (user), based on all their sold products.

In this paper, we advocate the need for a shift in how reputation systems are modelled, and we propose a new model for reputation systems in which a reputation value is given to each user, based on all their user behaviour or items. This is crucial in ensuring that a user's previous behaviour will contribute to their current reputation, instead of having separate reputations for each service provided. Clearly, if items belonging to a user could be linked together, the model which has been used so far could be transformed into our new one, by collating the reviews for each item belonging to a user to form a reputation value. However, if the user wishes to make their items unlinkable for privacy reasons, then this becomes more challenging.

1.1 Motivation and contribution

Our contribution is to propose a new model for reputation systems so that reputation values are given to users instead of items, whilst guaranteeing that the user's behaviour is unlinkable, and that the central server does not have to be involved during every transaction. This means that users can have multiple unlinkable items, whilst a reputation value still reflects their entire behaviour. Therefore users can have the benefits of privacy, whilst still being held accountable for the previous behaviour.

A car pooling app is an example of the reputation systems we are modelling. A user may not want their trips (or items) to be linked together, as their movements could be tracked. However, a user's reputation should be based on their previous trips, so others can judge their reliability. In this context, reputations based on each journey are not useful, as they cannot be used for future journeys. This is why it is important to give reputation values to users instead of items.

The first challenge when developing such reputation systems, is to provide a mechanism for generating reputation values, whilst ensuring items cannot be linked by user. We model this with a ReceiveFB algorithm run by the Central Server (CS), the central server, which takes feedback, and links it to other feedback on items with the same author, updating their reputation. We define the security requirement, *Unlinkability of User Behaviour*, which defines the unlinkability of items by the same user achievable while reputations can still be updated using ReceiveFB. Our approach as described so far gives rise to a possible attack in which a user produces a valid item which will not contribute to this user's reputation, or will even unfairly affect another user ReceiveFB. We introduce the *Traceability* security requirement to mitigate against this attack. These security requirements are reminiscent of those for group signature schemes in [4].

The second challenge is to determine the reputation of a specific user, whose items are unlinkable. A naive solution could be for the user to simply attach their reputation to an item, but the user could lie about their reputation. To avoid this, we introduce the PostItem algorithm, with which the user posts their item, and proves they were given a reputation at a particular time, using a token generated by the CS. We further introduce the security requirement of *Unforgeability of Reputation* to ensure the user cannot lie about their reputation.

Finally, the standard security requirements of a centralised reputation system [9,19], namely Anonymity of Feedback, Soundness of Reputation and Non–frameability, still need to hold, and we adapt these naturally to our new model.

A reputation system satisfying our security requirements can be built using two standard primitives: Group Signature Schemes [17] for posting items, under the condition that they can be modified so that users can prove their reputation at a particular time with a token generated by the CS, and Direct Anonymous Attestation (DAA) [13] for sending feedback. In this paper we present a concrete construction using a modified version of the group signature scheme given in [18], which ensures the Unlinkability of User Behaviour, Traceability and Unforgeability of Reputation requirements described above. Our modification to [18], similarly to in [25], allows users to prove their reputation at a particular time. Our construction also makes use of the DAA scheme given in [16], which ensures anonymity of feedback, whilst multiple feedback on the same item can be detected, ensuring Soundness of Reputation. This is due to the user controlled linkability property of the DAA scheme, where only signatures with the same basenames can be linked, and we use this by setting the basename to be the item feedback is given on.

2 Related Work

While there exists an abundant literature on centralised reputation systems [27,3,29,25,20,8] the most relevant work to this paper are [9] and [19], due to their focus on formal models of reputation systems.

The model proposed in [9] is inspired by the security model for dynamic group signatures [6], and the authors provide an extra linkability feature to detect users

giving multiple feedback on the same subject. In [19], the security requirements for this model are improved by giving more power to the adversary, (for example, in the public linkability security requirement the key issuer is no longer assumed to be honest), and introducing the requirement that an adversary cannot give feedback that will link to another user, invalidating their feedback. In [19] the model is also made fully dynamic [12], ie users can join or leave the scheme at any time, and a lattice-based instantiation satisfying this model is provided. To reduce complexity, our model is in the static setting, however the model and constuction could be converted to the dynamic case using [7], and due to the dynamic setting of our building blocks.

Crucially, in both [9] and [19] reputations are assigned to each *item*, the subject of feedback, not each *user*. By contrast, we propose a new model for reputation systems in which a reputation value is given to each user, based on all their user behaviour or items. This ensures a user's reputation reflects their entire past behaviour and so ensures they are accountable for their previous actions, modelling more accurately how such systems truly operate.

3 Defining a Reputation System

We define a reputation system, Π, as consisting of the following probabilistic polynomial time algorithms: Setup, AllocateReputation, PostItem, CheckItem, SendFB, VerifyFB, LinkFB, ReceiveFB. We illustrate our model in Figure 1.

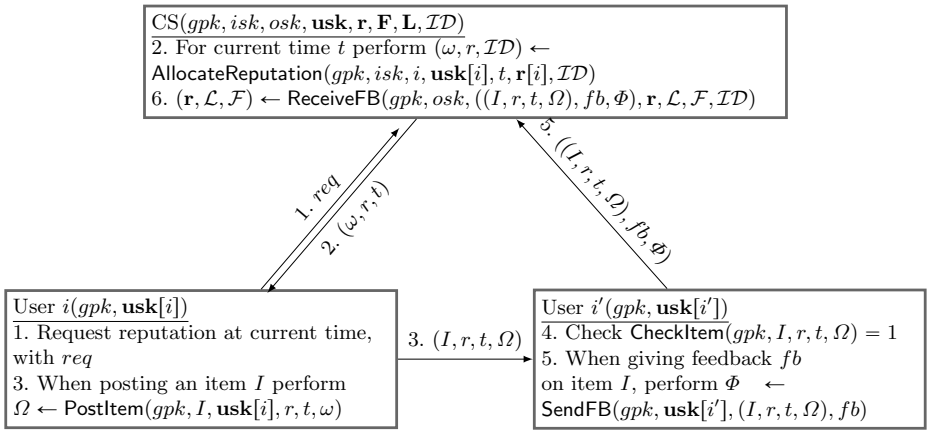

Fig. 1. Diagram modelling how entities interact in a centralised reputation system.

SendFB, VerifyFB, and LinkFB are equivalent to the Sign, Verify, Link algorithms in [9] and [19]. The additional algorithms, which we introduce in this paper, represent the key features of our new approach.

The entities involved are a set of users $\mathcal{U}$ and a Central Server (CS). The Central Server has two secret keys, isk and osk. The issuing secret key isk is

necessary for allowing users to join the system and allocating them tokens to prove their reputation, whereas the opening secret key, osk, is necessary for forming reputations from feedback. For simplicity we give the CS both secret keys, but to reduce the power of one entity the role of the CS could be distributed.

The CS begins by running Setup. Users post items[1], which are the subject of feedback, while proving their reputation at a certain time using PostItem. After a request from a user, the CS runs AllocateReputation, which outputs tokens to allow a user to prove their current reputation at a specific time in PostItem. And other users verify that an item is a valid output of PostItem by running CheckItem. This ensures the item was authored by an enrolled user, the reputation alongside the item is correct for the given time, and the CS can use feedback on the item to form reputations. SendFB is run by a user when giving feedback on an item, and its output is sent to the CS. ReceiveFB is run by the Central Server when receiving the output of SendFB from a user. The CS updates their stored feedback and reputations, based on this. VerifyFB and LinkFB are used by ReceiveFB to check the feedback is valid and that there is no feedback by the same user on this item, otherwise ReceiveFB will abort.

In the car pooling example, whenever a driver wishes to update their reputation, they request the CS run AllocateReputation to obtain a token for their reputation. They are incentivised to do this by the fact the reputation is displayed alongside the time it was allocated. When they wish to give a ride, they use their most recent token to post an item with PostItem, which can be verified by passengers with CheckItem. The passenger can then pay using some anonymous payment system. After the ride, their passenger can then give feedback on this item to the CS using SendFB. The CS uses ReceiveFB to update their lists of feedback, and reputations for each user, if the feedback is valid.

Before describing in detail our new model, we provide, for ease of reading, an overview of our notation.

- $\mathcal{R}$: The set of all possible reputation values.
- $\hat{r}$: The initial reputation of every user at the system's setup.
- $\mathcal{U}$: The set of all users in the scheme.
- Aggr : A function that takes as input the new feedback fb, the user who's reputation is being updated i, the list of feedback already received $\mathcal{F}$, and the most recent reputation r, and outputs the new reputation r'.
- $\mathbf{r}$: For the user $i \in \mathcal{U}$, $\mathbf{r}[i]$ is the user i's reputation held by the CS.
- $\mathcal{L}$: A list of feedback that will contain entries in the form of a 6-tuple $((I, r, t, \Omega), fb, \Phi)$, where (fb, Φ) is feedback/ proof pair, given on item I with reputation r, and time t, with the proof Ω. $\mathcal{L}$ is used by the CS to keep track of all feedback given, so that multiple feedback on the same item can be detected in ReceiveFB.
- $\mathcal{F}$: A list of feedback that will contain entries of the form (i, fb) where fb is feedback given to user i. $\mathcal{F}$ is used by the CS to keep track of all feedback given on user i to form reputations in ReceiveFB.

[1] A simple example of an item could be a product being sold.

$\mathcal{ID}$: A list of identities for all users, this list will allow the CS to store information on users whilst running AllocateReputation for use in ReceiveFB.

We next formally define a centralised reputation system Π, consisting of the following probabilistic polynomial time algorithms: Setup, AllocateReputation, PostItem, CheckItem, SendFB, VerifyFB, LinkFB, ReceiveFB.

- $\mathsf{Setup}(k, \mathcal{R}, \hat{r}, \mathcal{U}, \mathsf{Aggr})$ takes as input: a security parameter k, a set $\mathcal{R}$ of reputation values, $\hat{r} \in \mathcal{R}$, the initial reputation, a set of users $\mathcal{U}$, and the aggregation algorithm Aggr. The CS computes a public key gpk, the issuing secret key isk, which is used to issue new user secret keys, and in AllocateReputation, and the opening secret key osk, which is used in ReceiveFB to trace the author of an item to form reputations. The CS computes a secret key for each user, $\mathbf{usk} = \{\mathbf{usk}[i] : i \in \mathcal{U}\}$, and $\mathbf{r}$, the reputation for all users held by the CS, where $\forall i \in \mathcal{U}, \mathbf{r}[i] = \hat{r}$. The CS creates empty lists $\mathcal{L}, \mathcal{F}, \mathcal{ID}$ which are described above. It outputs $(gpk, isk, osk, \mathbf{usk}, \mathbf{r}, \mathcal{L}, \mathcal{F}, \mathcal{ID})$.
- $\mathsf{AllocateReputation}(gpk, isk, i, \mathbf{usk}[i], t, \mathbf{r}[i], \mathcal{ID})$ takes as input the public key gpk, the issuing secret key isk, user i's secret key $\mathbf{usk}[i]$, the current time t, the current reputation of user i held by the CS $\mathbf{r}[i]$, and the list of identities for users $\mathcal{ID}$. It updates the list of identities $\mathcal{ID}$, and outputs $(\omega, \mathbf{r}[i], \mathcal{ID})$, where ω allows user i to prove they have reputation $\mathbf{r}[i]$.
- $\mathsf{PostItem}(gpk, I, \mathbf{usk}[i], r, t, \omega)$ takes as input the public key gpk, an item I, user i's secret key $\mathbf{usk}[i]$, the last reputation r, time t and token ω received from the CS (r is not necessarily the reputation $\mathbf{r}[i]$ held by the CS). It outputs Ω, which proves the author is enrolled and has reputation r at time t, and is used in ReceiveFB to form a reputation for i.
- $\mathsf{CheckItem}(gpk, I, r, t, \Omega)$ takes as input the public key gpk, an item I, a reputation r, a time t and Ω. It outputs 1 if Ω is a valid output of PostItem, given (I, r, t), and 0 otherwise.
- $\mathsf{SendFB}(gpk, \mathbf{usk}[i], (I, r, t, \Omega), fb)$ takes as input the public key gpk, user i's secret key $\mathbf{usk}[i]$, the subject of their feedback, (I, r, t, Ω), and the feedback fb. It outputs Φ which is sent to the CS, to prove the author of Φ is enrolled, and also for the detection of multiple feedback.
- $\mathsf{VerifyFB}(gpk, (I, r, t, \Omega), fb, \Phi)$ takes as input the public key gpk, an item (I, r, t, Ω), and feedback/ proof pair on this item (fb, Φ). It outputs 1 if Φ is a valid output of SendFB, and 0 otherwise.
- $\mathsf{LinkFB}(gpk, (I, r, t, \Omega), fb_0, \Phi_0, fb_1, \Phi_1)$ takes as input the public key gpk, an item (I, r, t, Ω), and two feedback/ proof pairs on this item, (fb_0, Φ_0), (fb_1, Φ_1). It outputs 1 if Φ_0 and Φ_1 were generated by the same user with the same input of (I, r, t, Ω), and 0 otherwise.
- $\mathsf{ReceiveFB}(gpk, osk, ((I, r, t, \Omega), fb, \Phi), \mathbf{r}, \mathcal{L}, \mathcal{F}, \mathcal{ID})$ takes as input the public key gpk, the opening secret key osk, a feedback/ proof pair (fb, Φ) on item (I, r, t, Ω), the current reputations $\mathbf{r}$ held by the CS, the lists of feedback so far $\mathcal{L}$ and $\mathcal{F}$, and the list of user identities $\mathcal{ID}$. If Φ is not valid, or the LinkFB algorithm finds multiple feedbacks in $\mathcal{L}$ then it outputs $\perp$. Otherwise, it uses the aggregation algorithm Aggr, and the list $\mathcal{F}$, to update $\mathbf{r}$, $\mathcal{L}$ and $\mathcal{F}$ to take into account the new feedback. It outputs $(\mathbf{r}, \mathcal{L}, \mathcal{F})$.

4 Security Requirements

As discussed earlier, we consider reputation systems satisfying the following requirements: Correctness, Unforgeability of Reputation, Traceability, Unlinkability of User Behaviour, Soundness of Reputation, Anonymity of Feedback, and Non–frameability. We begin with an informal discussion explaining the necessity for our security requirements and then follow up with formal definitions for the three security requirements original to this work.

We propose Unforgeability of Reputation, a new requirement that ensures a user cannot prove that they have a reputation for a certain time, which differs from the one they were allocated by the CS in AllocateReputation. This is necessary because when an item is unlinkable, the author's reputation cannot be determined. Therefore the reputation must be included alongside the item. This requirement ensures that the sender has not lied about their reputation.

Here we introduce Unlinkability of User Behaviour, which formalises our definition of unlinkable user behaviour, given that ReceiveFB can still form reputations, as well as Traceability, which ensures that all items generated by an adversary can be traced back to them when computing their reputation. This is necessary because, due to the Unlinkability of User Behaviour requirement, an attacker could attempt to subvert ReceiveFB. These requirements are reminiscent of the Full-Anonymity and Full-Traceability requirements [4] for group signature schemes, and have been adapted for reputation systems.

Soundness of Reputation ensures an adversary cannot give multiple feedback on the same item, undermining the integrity of reputation values. Anonymity of Feedback ensures that feedback cannot be traced to the user's identity and is unlinkable. We have adapted these two requirements from [9] to fit our notation[2]. Non–frameability, adapted from [19], ensures that an adversary cannot forge feedback that links to another user's feedback, so this feedback is unfairly disregarded. The Traceability requirement from [19] is not carried over, as we believe opening of feedback would add unnecessary complexity to the model.

We highlight that the issuing secret key is used by the Central Server for joining users to the scheme, and therefore for the Traceability and Soundness of Reputation requirements the adversary cannot corrupt the isk as otherwise they could cheat by creating unregistered users. The opening secret key is used by the Central Server to trace items, so that reputations can be updated with new feedback. Therefore in the Unlinkability of User Behaviour requirement the adversary cannot corrupt the osk as otherwise they could trace signatures. This means the CS could be split into two separate entities with different secret keys.

In Figure 2, we provide the oracles used in our security requirements: `USK`, `POSTITEM`, `SENDFB`, `RECEIVEFB` and `ALLOCATEREP`. `USK` allows the adversary to obtain users' secret keys. `POSTITEM` allows the adversary to obtain valid items of a user, without their secret key. `SENDFB` allows the adversary to obtain valid feedbacks of a user, without their secret key, storing outputs in the sets $\mathcal{G}_i$, for

[2] Soundness of Reputation is comparable to Public Linkability and Anonymity of Feedback is comparable to Anonymity.

use in the Non–frameability requirement. RECEIVEFB allows the adversary to discover the output of ReceiveFB, without the opening secret key, osk. ALLOCATEREP allows the adversary to obtain outputs of the AllocateReputation algorithm, without the issuing secret key, isk.

$\underline{\texttt{USK}(\text{i}):}$

$\mathcal{C} \leftarrow \mathcal{C} \cup \{i\}; \textbf{return usk}[i]$

$\underline{\texttt{POSTITEM}(I, i, r, t, \omega):}$

$\textbf{return } \mathsf{PostItem}(gpk, I, \textbf{usk}[i], r, t, \omega)$

$\underline{\texttt{SENDFB}(i, fb, (I, r, t, \Omega)):}$

$\Phi \leftarrow \mathsf{SendFB}(gpk, \textbf{usk}[i], (I, r, t, \Omega), fb), \mathcal{G}_i \leftarrow \mathcal{G}_i \cup \{((I, r, t, \Omega), fb, \Phi)\} \quad \textbf{return } \Phi$

$\underline{\texttt{RECEIVEFB}((I, r, t, \Omega), fb, \Phi, \mathcal{ID}):}$

$\textbf{return } (\mathbf{r}, \mathcal{L}, \mathcal{F}) \leftarrow \mathsf{ReceiveFB}(gpk, osk, ((I, r, t, \Omega), fb, \Phi), \mathbf{r}, \mathcal{L}, \mathcal{F}, \mathcal{ID})$

$\underline{\texttt{ALLOCATEREP}(i, t, r):}$

$\textbf{return } \mathsf{AllocateReputation}(gpk, isk, i, \textbf{usk}[i], t, r, \mathcal{ID})$

Fig. 2. Oracles used in our Security Requirements

We next formally define our requirements. The full correctness conditions as well Soundness of Reputation Values, Anonymity of Feedback, and Non-Frameability are given in the full version [21] of this paper due to their similarities to existing work.

Correctness: There are five conditions for correctness. Condition 1 ensures that if AllocateReputation and PostItem are computed honestly then CheckItem will output 1. Condition 2 ensures that if SendFB is computed honestly then VerifyFB will output 1. Condition 3 ensures the LinkFB algorithm will output 1, with input valid outputs of SendFB on the same item of (I, r, t, Ω), using the same user secret key. Condition 4 ensures if an item and feedback were generated honestly in PostItem and SendFB, then ReceiveFB updates $\mathbf{r}, \mathcal{L}, \mathcal{F}$ correctly. Condition 5 ensures that ReceiveFB fails, if the feedback input is not valid according to VerifyFB, or links to other feedback in $\mathcal{L}$ according to LinkFB.

We first present our new security requirements, which are necessary as reputation values are assigned to users instead of their individual unlinkable items.

Unforgeability of Reputation: A user can only prove that they have reputation r at time t, if this was allocated to them by the CS in AllocateReputation. In the context of car pooling, this security requirement means that a driver cannot lie about their reputation when requesting a passenger.

In our security game in Figure 3, the adversary is given the opening secret key osk, the list of user identities $\mathcal{ID}$, the USK, POSTITEM, ALLOCATEREP oracles, but not isk, as they could run AllocateReputation. The adversary wins if they output a valid item, for reputation r, time t, tracing to a corrupted user i in ReceiveFB, without querying (i, r, t) to the ALLOCATEREP oracle, or it does not trace to any user.

Experiment: $\mathsf{Exp}^{\text{anon-ub}}_{\mathcal{A},\Pi}(k, \mathcal{R}, \hat{r}, \mathcal{U}, \mathsf{Aggr})$

$b \leftarrow\$ \{0,1\}$; $\quad (gpk, isk, osk, \mathbf{usk}, \mathbf{r}, \mathcal{L}, \mathcal{F}, \mathcal{ID}) \leftarrow\$ \mathsf{Setup}(k, \mathcal{R}, \hat{r}, \mathcal{U}, \mathsf{Aggr})$

$(\mathrm{St}, i_0, i_1, I, r, t, \mathcal{ID}) \leftarrow\$ \mathcal{A}^{\text{RECEIVEFB}}(\mathsf{choose}, gpk, isk, \mathbf{usk}, \mathbf{r}, \mathcal{L}, \mathcal{F}, \mathcal{ID})$

$\forall \tilde{b} \in \{0,1\} \quad (\omega_{\tilde{b}}, \mathcal{ID}) \leftarrow \mathsf{AllocateReputation}(gpk, isk, i, \mathbf{usk}[i_{\tilde{b}}], t, r, \mathcal{ID})$

$\Omega \leftarrow \mathsf{PostItem}(gpk, I, \mathbf{usk}[i_b], r, t, \omega_b)$

$d \leftarrow\$ \mathcal{A}^{\text{RECEIVEFB}}(\mathsf{guess}, St, \Omega, \mathcal{ID})$; $\quad d' \leftarrow\$ 0,1$

if $((I, r, t, \Omega), \cdot)$ queried to the RECEIVEFB oracle **return** $d \leftarrow d'$

if $d = b$ **return** 1; **else return** 0

Experiment: $\mathsf{Exp}^{\text{trace}}_{\mathcal{A},\Pi}(k, \mathcal{R}, \hat{r}, \mathcal{U}, \mathsf{Aggr})$

$(gpk, isk, osk, \mathbf{usk}, \mathcal{ID}) \leftarrow\$ \mathsf{Setup}(k, \mathcal{R}, \hat{r}, \mathcal{U}, \mathsf{Aggr}); C \leftarrow \varnothing$

$(I, r, t, \Omega, fb, \Phi, \mathbf{r}, \mathcal{L}, \mathcal{F}) \leftarrow\$ \mathcal{A}^{\text{USK,POSTITEM,SENDFB, ALLOCATEREP}}(gpk, osk, \mathcal{ID})$

if $\mathsf{CheckItem}(gpk, I, r, t, \Omega) = 0$ or $\mathsf{VerifyFB}(gpk, (I, r, t, \Omega), fb, \Phi) = 0$ **return** 0

if $\exists((I, r, t, \Omega), fb', \Phi') \in \mathcal{L}$ with $\mathsf{LinkFB}(gpk, (I, r, t, \Omega), fb, \Phi, fb', \Phi') = 1$ **return** 0

$\forall i \in \mathcal{U}, \mathcal{ID} \leftarrow \mathsf{AllocateReputation}(gpk, isk, i, \mathbf{usk}[i], t, r, \mathcal{ID})$

if $\perp \leftarrow \mathsf{ReceiveFB}(gpk, osk, ((I, r, t, \Omega), fb, \Phi), \mathbf{r}, \mathcal{L}, \mathcal{F}, \mathcal{ID})$ **return** 1

else $(\mathbf{r}^*, \mathcal{L}^*, \mathcal{F}^*) \leftarrow \mathsf{ReceiveFB}(gpk, osk, ((I, r, t, \Omega), fb, \Phi), \mathbf{r}, \mathcal{L}, \mathcal{F}, \mathcal{ID})$

if $\mathcal{L}^* \neq ((I, r, t, \Omega), fb, \Phi) \cup \mathcal{L}$ **return** 1

if $\mathcal{F}^* = (i', fb) \cup \mathcal{F}$ for some $i' \in \mathcal{U}$ $\quad i^* \leftarrow i'$ **else return** 1

if $\mathbf{r}^*[i^*] \neq \mathsf{Aggr}(fb, i^*, \mathcal{F}, \mathbf{r}[i^*])$, or $\exists \hat{i} \in \mathcal{U} \backslash \{i^*\}$ such that $\mathbf{r}^*[\hat{i}] \neq \mathbf{r}[\hat{i}]$ **return** 1

if $i^* \notin \mathcal{C}$ and $(I, i^*, r, t, \cdot)$ was not queried to the POSTITEM oracle **return** 1

else **return** 0

Experiment: $\mathsf{Exp}^{unforge-rep}_{\mathcal{A},\Pi}(k, \mathcal{R}, \hat{r}, \mathcal{U}, \mathsf{Aggr})$

$(gpk, isk, osk, \mathbf{usk}, \mathbf{r}, \mathcal{L}, \mathcal{F}, \mathcal{ID}) \leftarrow\$ \mathsf{Setup}(k, \mathcal{R}, \hat{r}, \mathcal{U}, \mathsf{Aggr})$

$(I, r, t, \Omega) \leftarrow\$ \mathcal{A}^{\text{USK,POSTITEM,ALLOCATEREP}}(gpk, osk, \mathcal{ID})$

if Ω returned by POSTITEM or **if** $\mathsf{CheckItem}(gpk, I, r, t, \Omega) = 0$ **return** 0

$j \leftarrow\$ \mathcal{U}, fb \leftarrow 0, \Phi \leftarrow \mathsf{SendFB}(gpk, \mathbf{usk}[j], (I, r, t, \Omega), fb)$

$(\mathbf{r}^*, \mathcal{L}^*, \mathcal{F}^*) \leftarrow \mathsf{ReceiveFB}(gpk, osk, ((I, r, t, \Omega), fb, \Phi), \mathbf{r}, \mathcal{L}, \mathcal{F}, \mathcal{ID})$

if $\mathcal{F}^* \backslash \mathcal{F} = \{i', fb\}$ for some $i' \in \mathcal{U}$ $\quad i^* \leftarrow i'$ **else return** 1

if $\mathcal{A}$ queried (i^*, t, r) to ALLOCATEREP oracle and $i^* \in \mathcal{C}$ **return** 0 **else return** 1

Fig. 3. Experiments capturing our Unlinkability of User Behaviour, Traceability and Unforgeability of Reputation security requirements

A reputation system Π satisfies Unforgeability of Reputation if for all polynomial time adversaries $\mathcal{A}$, all sets $\mathcal{R}$ and $\mathcal{U}$ such that $|\mathcal{R}|$ and $|\mathcal{U}|$ are polynomially bounded in k, all $\hat{r} \in \mathcal{R}$, all Aggr functions, there exists a negligible function in k, $negl$, such that: $Pr[\mathsf{Exp}^{unforge-rep}_{\mathcal{A},\Pi}(k, \mathcal{R}, \hat{r}, \mathcal{U}, \mathsf{Aggr}) = 1] \leqslant negl$.

Traceability of Users: This security requirement ensures that any valid item an adversary produces will contribute towards their own reputation in Re-

ceiveFB. This also guarantees unforgeability. In the context of car pooling, this security requirement means that feedback on a driver's rides will always affect their own reputation and not another's.

In our security game in Figure 3, the adversary is given the opening secret key osk, the list of user identities $\mathcal{ID}$, the `USK` oracle to corrupt users, and the `POSTITEM`, `SENDFB`, `ALLOCATEREP` oracles for uncorrupted user, but not isk, because they could cheat by generating the secret key of a new user. They must output a valid item and feedback, and $\mathbf{r}, \mathcal{L}, \mathcal{F}$, such that the feedback does not link to any in $\mathcal{L}$. If ReceiveFB fails, does not correctly update $\mathbf{r}, \mathcal{L}, \mathcal{F}$, or updates the reputation of a non corrupted user, then the adversary wins.

A reputation system Π satisfies Traceability if for all polynomial time adversaries $\mathcal{A}$, all sets $\mathcal{R}$ and $\mathcal{U}$ such that $|\mathcal{R}|$ and $|\mathcal{U}|$ are polynomially bounded in k, all $\hat{r} \in \mathcal{R}$, all Aggr functions, there exists a negligible funtion in k, $negl$, such that: $Pr[\mathsf{Exp}^{trace}_{\mathcal{A},\Pi}(k, \mathcal{R}, \hat{r}, \mathcal{U}, \mathsf{Aggr}) = 1] \leqslant negl$.

Unlinkability of User Behaviour: This requirement ensures other users cannot link together the items authored by a particular user, while the CS can link items to form reputation values based on a user's entire behaviour. In the context of car pooling, this security requirement means that all rides a driver/ user undertakes are unlinkable, so their movements cannot be tracked.

In our security game, given in Figure 3, the adversary is given all user secret keys, the issuing secret key isk, $\mathbf{r}$, $\mathcal{L}$, $\mathcal{F}$, and $\mathcal{ID}$, but not the opening secret key osk, because otherwise they could run ReceiveFB, and then check which user's reputation changes. They are given the `RECEIVEFB` oracle, but its use is restricted so that the challenge signature cannot be queried, to avoid the attack above. This attack would not be practical in the real world, as reputations will be updated at intervals so that multiple users' reputations will change at once. Future work could consider specific Aggr algorithms that would allow this security requirement to be strengthened. In our work, to ensure our model is generic, we define security for all possible Aggr functions.

The adversary chooses an item I, a reputation r and a time t, an updated list of identities $\mathcal{ID}$, and two users i_0, i_1, they are then given Ω and must decide whether it was authored by i_0 or i_1.

A reputation system Π satisfies Unlinkability of User Behaviour if for all polynomial time adversaries $\mathcal{A}$, all sets $\mathcal{R}$ and $\mathcal{U}$ such that $|\mathcal{R}|$ and $|\mathcal{U}|$ are polynomially bounded in k, and all $\hat{r} \in \mathcal{R}$, all Aggr functions, there exists a negligible funtion in k, $negl$, such that: $Pr[\mathsf{Exp}^{anon-ub}_{\mathcal{A},\Pi}(k, \mathcal{R}, \hat{r}, \mathcal{U}, \mathsf{Aggr}) = 1] - 1/2 \leqslant negl$.

We now give an overview of the existing security requirements.

Soundness of Reputation Values: Users who are not enrolled should not be able to give feedback. Reputation values should be based on only one piece of feedback per item per user. In the context of car pooling, this security requirement would mitigate against an attack where a passenger repeatedly gives feedback on one ride, unfairly negatively influencing the driver's reputation.

In the security game, adapted from [9], given in the full version [21], the adversary is able to corrupt users with the `USK` oracle, and is given the opening

secret key osk, but not the issuing key isk, as they could use this to cheat by generating a secret key for a new user. They can use the SENDFB, ALLOCATEREP and POSTITEM oracles for uncorrupted users. The adversary outputs a list of feedback on the same item. They win if they can output more valid unlinkable feedback than the number of corrupted users, without using the SENDFB oracle.

Anonymity of Feedback: Anonymity of Feedback captures the anonymity of feedback senders against the Central Server, and up to all but two colluding users. Unfortunately it is not possible for a reputation system to have anonymity against all colluding users, whilst still satisfying Soundness of Reputation. This is because an adversary could discover whether a user i authored some feedback $((I, r, t, \Omega), fb, \Phi)$ by running $\Phi' \leftarrow \mathsf{SendFB}(gpk, \mathbf{usk}[i], (I, r, t, \Omega), fb')$, then running $\mathsf{LinkFB}(gpk, (I, r, t, \Omega), fb, \Phi, fb', \Phi')$. If this outputs 1, then $((I, r, t, \Omega), fb, \Phi)$ must be authored by i. In the context of car pooling, this security requirement means that provided passengers never give multiple feedback on the same ride, their feedback will be unlinkable.

In the security game, adapted from [9], and given in the full version [21], the adversary is given isk, osk, and must choose two users i_0 and i_1, an item (I, r, t, Ω), and feedback fb. They then must decide which of these users authored the Φ returned to them. The adversary can corrupt users with USK, and use SENDFB, POSTITEM and ALLOCATEREP for uncorrupted users. We do not allow the adversary to query i_0 or i_1 to the USK oracle, or to query SENDFB with either i_0 or i_1 and (I, r, t, Ω), so that they cannot perform the above attack.

Non–frameability: This requirement, adapted from [19], ensures that an adversary, who has corrupted the Central Server and all users, cannot forge feedback that links to feedback of another user, meaning ReceiveFB detects multiple feedback by this user, and unfairly outputs $\perp$. In the context of car pooling, this security requirement means that a passenger cannot feedback on their own ride, linking to the driver involved, invalidating any feedback they give.

In the security game, given in the full version [21], the adversary is given isk, osk and can corrupt users using the USK oracle, and use the POSTITEM, SENDFB, ALLOCATEREP oracles for uncorrupted users. To win, they must output valid feedback not output by the SENDFB oracle, which links to feedback output by the SENDFB oracle, authored by an uncorrupted user.

5 A Centralised Reputation System with Unlinkable User Behaviour

We now give a construction for Π, a reputation system as defined in Section 3, satisfying the security requirements we defined in Section 4. Our construction makes use of two existing primitives: a Group Signature scheme [17], and Direct Anonymous Attestation (DAA) [13].

More specifically, we modify the group signature scheme XS [18], in XS*, similarly to what was done in [25,20], for posting items in PostItem, CheckItem, and AllocateReputation. The XS scheme satisfies Unlinkability of User Behaviour, whilst still allowing reputations to be formed in ReceiveFB, using the opening

key ensuring Traceability. Furthermore our modification allows a user to prove they were allocated a reputation at a certain time by AllocateReputation.

We then adopt the DAA scheme in [16] for the feedback component of the reputation system in SendFB, VerifyFB, LinkFB. This perfectly fits our requirements, because of the user controlled linkability of the DAA scheme. Signatures are signed with respect to a basename, and are linkable only when they have the same author and basename. Therefore in the context of reputation systems, by setting the basename to be the subject of the feedback, multiple feedback on the same item can be detected, whilst still ensuring Anonymity of Feedback.

5.1 Binding Reputation to the XS Group Signature Scheme

Group Signatures [17] prove a user is a member of a group without revealing their identity, except to those with an opening key. Security requirements were defined for static groups [4], partially dynamic groups [6], and fully dynamic groups [12]. The XS scheme [18] satisfies the security requirements for partially dynamic groups [6], of Anonymity, Traceability and Non-Frameability, under the q-SDH [10] assumption and in the random oracle model [5].

q-Strong Diffie Hellman Assumption (q-SDH) There are two versions of the q-Strong Diffie Hellman Assumption. The first version, given by Boneh and Boyen in [10], is defined in a type-1 or type-2 pairing setting. We use their second version of that definition that supports type-3 pairings and was stated in the journal version of their paper [11].

Given $(g_1, g_1^{\chi}, g_1^{(\chi)^2}, ..., g_1^{(\chi)^q}, g_2, g_2^{\chi})$ such that $g_1 \in \mathbb{G}_1, g_2 \in \mathbb{G}_2$, output $(g_1^{\frac{1}{\chi+x}}, x) \in \mathbb{G}_1 \times \mathbb{Z}_p \backslash \{-\chi\}$.

We present XS*, a modification of the XS scheme [18], to allow users to prove their reputation in PostItem. In this modification, we introduce an additional algorithm XSUpdate*, used in AllocateReputation, which outputs a token allowing a user to update their secret key, depending on their reputation r at time t. PostItem uses XSSign* to sign as in the original group signature scheme, but with this updated secret key as input. CheckItem uses XSVerify*, which is modified so that it takes (r, t) as input, and only outputs 1 if the secret key used to generate this signature has been updated correctly with (r, t).

The XS* signature scheme consists of the algorithms given in Figure 4, and the group public parameters gpp_1 chosen as follows. Let $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_3$ be multiplicative cyclic groups with large prime order p, with $|p| = k$, and with generators G_1 and G_2 respectively. Let $\hat{t} : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_3$, be a bilinear map. The q-SDH assumption must hold in $(\mathbb{G}_1, \mathbb{G}_2)$. Select two hash functions: $\mathcal{H}_1 : \{0,1\}^* \rightarrow \mathbb{G}_1$ and $\mathcal{H}_2 : \{0,1\}^* \rightarrow \mathbb{Z}_p^*$. The group public parameters for XS* are: $gpp_1 = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_3, p, \hat{t}, G_1, G_2, \mathcal{H}_1, \mathcal{H}_2)$.

5.2 Direct Anonymous Attestation

Direct Anonymous Attestation (DAA) [13] allows users to prove they are members of a group. There is no opening key but there is user controlled linkability,

XSKeyGen*(gpp_1)

$\xi_1, \xi_2 \leftarrow\$ \mathbb{Z}_p; K \leftarrow\$ \mathbb{G}_1 \backslash \{1_{\mathbb{G}_1}\}, H \leftarrow K^{\xi_1}, G \leftarrow K^{\xi_2}, \gamma \leftarrow\$ \mathbb{Z}_p, W \leftarrow G_2^{\gamma}$
return $gpk_1 = (G_1, K, H, G, G_2, W), isk_1 = \gamma, osk = (\xi_1, \xi_2)$

XSJoin*(isk_1, i, gpk_1)

$x, y \leftarrow\$ \mathbb{Z}_p; \quad Z \leftarrow (G_1 H^y)^{\frac{1}{\gamma + x}}$
return $\mathbf{usk}_1[i] = (Z, x, y)$

XSUpdate*($t, r, isk_1, (Z, x, y), gpk_1$)

$Q \leftarrow \mathcal{H}_1(r, t)$
return $(\omega \leftarrow Q^{\frac{1}{\gamma + x}}, Z \cdot \omega)$

XSSign*($I, (\tilde{Z}, x, y), gpk_1, r, t$)

$\rho_1, \rho_2 \leftarrow\$ \mathbb{Z}_p, T_1 \leftarrow K^{\rho_1}, T_2 \leftarrow \tilde{Z} H^{\rho_1}, T_3 \leftarrow K^{\rho_2}, T_4 \leftarrow \tilde{Z} G^{\rho_2}, r_{\rho_1}, r_{\rho_2}, r_x, r_z \leftarrow\$ \mathbb{Z}_p$
$R_1 \leftarrow K^{r\rho_1}, R_3 \leftarrow K^{r\rho_2}, R_4 \leftarrow H^{r\rho_1} G^{-r\rho_2}, R_2 \leftarrow \hat{t}(T_2, G_2)^{r_x} \hat{t}(H, W)^{-r\rho_1} \hat{t}(H, G_2)^{-r_z}$
$c \leftarrow \mathcal{H}_2(I, T_1, T_2, T_3, T_4, R_1, R_2, R_3, R_4, r, t), z \leftarrow x\rho_1 + y, s_{\rho_1} = r_{\rho_1} + c\rho_1, s_{\rho_2} = r_{\rho_2} + c\rho_2$
$s_x = r_x + cx, s_z = r_z + cz$ **return** $\Omega = (T_1, T_2, T_3, T_4, c, s_{\rho_1}, s_{\rho_2}, s_x, s_z)$

XSVerify*(I, r, t, Ω, gpk_1)

Let $\Omega = (T_1, T_2, T_3, T_4, c, s_{\rho_1}, s_{\rho_2}, s_x, s_z), \tilde{G}_1 = G_1 \cdot \mathcal{H}_1(r, t)$
$\tilde{R}_1 \leftarrow K^{s\rho_1} T_1^{-c}, \tilde{R}_3 \leftarrow K^{s\rho_2} T_3^{-c}, \tilde{R}_4 = H^{s\rho_1} G^{-s\rho_2} T_4^{c} T_2^{-c}$
$\tilde{R}_2 = \hat{t}(T_2, G_2)^{s_x} \hat{t}(H, W)^{-s\rho_1} \hat{t}(H, G_2)^{-s_z} \left(\frac{\hat{t}(T_2, W)}{\hat{t}(\tilde{G}_1, G_2)} \right)^{c}$
if $c = \mathcal{H}_2(I, T_1, T_2, T_3, T_4, \tilde{R}_1, \tilde{R}_2, \tilde{R}_3, \tilde{R}_4, r, t)$ **return** 1 **else return** 0

XSOpen*($I, r, t, \Omega, osk, gpk_1$)

if XSVerify*$(I, r, t, \Omega, gpk_1) = 0$ **return** $\perp$ **return** $\tilde{Z} \leftarrow T_2 T_1^{-\xi_1}$

Fig. 4. The algorithms of XS*, our modification to [18]

as defined at the beginning of this section. In DAA, a signer consists of two separate entities: a trusted TPM and a host with higher computational power. A DAA scheme is secure if it is indistinguishable from the ideal functionality, given in [14]. The CDL scheme [16], is proved secure, assuming the LSRW [23], Discrete Logarithm (DL), and DDH assumptions. We use the CDL DAA scheme in particular, because as shown in Table 1 of [15], it has the lowest estimated running time for signing out of the schemes proved secure under the more recent models. We prioritise efficiency of signing over verification, because in reputation systems verification is performed by a server with more computational power.

The CDL scheme, with the TPM and host merged, consists of the algorithms in Figure 5, and the group public parameters gpp_2. Let $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_3$ be multiplicative cyclic groups with large prime order p, with $|p| = k$, and with generators G_1 and G_2. Let $\hat{t} : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_3$, be a bilinear map. The DDH and DL problem must be hard in $\mathbb{G}_1$, and the bilinear LRSW [23] problem must be hard in $(\mathbb{G}_1, \mathbb{G}_2)$. Select two hash functions: $\mathcal{H}_1 : \{0,1\}^* \rightarrow \mathbb{G}_1$, $\mathcal{H}_2 : \{0,1\}^* \rightarrow \mathbb{Z}_p^*$. The group public parameters for CDL are: $gpp_2 = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_3, p, \hat{t}, G_1, G_2, \mathcal{H}_1, \mathcal{H}_2)$.

CDLKeyGen(gpp_2)

$\alpha, \beta \leftarrow\$ \mathbb{Z}_p^*, X \leftarrow {G_2}^{\alpha} \in \mathbb{G}_2, Y = {G_2}^{\beta} \in \mathbb{G}_2$
return $gpk_2 = ((X, Y),\ isk_2 = (\alpha, \beta))$

CDLJoin(i, isk_2, gpk_2)

$f \leftarrow\$ \mathbb{Z}_p; F \leftarrow {G_1}^{f}, r \leftarrow\$ \mathbb{Z}_p$
$A \leftarrow {G_1}^{r}, B \leftarrow A^{\beta}; C \leftarrow (A^{\alpha} F^{r\alpha\beta}), D \leftarrow (F)^{r\beta}$
$cre \leftarrow (A, B, C, D)$ **return** $\mathbf{usk}_2[i] \leftarrow (f, cre)$

CDLSign($msg, fb, (f, (A, B, C, D)), gpk_2$)

$a \leftarrow\$ \mathbb{Z}_p, A' \leftarrow A^a, B' \leftarrow B^a, C' \leftarrow C^a, D' \leftarrow D^a, z \leftarrow\$ \mathbb{Z}_p, J \leftarrow \mathcal{H}_1(msg)^f, M \leftarrow {B'}^z, N \leftarrow \mathcal{H}_1(msg)^z$
$c \leftarrow \mathcal{H}_2(B'||D'||J||M||N||msg||fb), s \leftarrow z - c \cdot f \pmod p$ **return** $\Phi = (A', B', C', D', J, c, s)$

CDLVerify(msg, fb, Φ, gpk_2)

Let $\Phi = (A', B', C', D', J, c, s), \tilde{M} \leftarrow {B'}^s {D'}^c, \tilde{N} \leftarrow \mathcal{H}_1(msg)^s J^c$
if $c \neq \mathcal{H}_2(B'||D'||J||\tilde{M}||\tilde{N}||msg||fb)$ or $A' = 1$ **return** 0
if $\hat{t}(A', Y) \neq \hat{t}(B', G_2)$ or $\hat{t}(A'D', X) \neq \hat{t}(C', G_2)$ **return** 0 **else return** 1

CDLLink($msg, (fb_0, \Phi_0), (fb_1, \Phi_1), gpk_2$)

if $\exists \iota \in \{0, 1\}$ with CDLVerify$(msg_\iota, fb_\iota, \Phi_\iota, gpk_2) = 0$ **return** 0
For $\iota \in \{0, 1\}$, let $\Phi_\iota = ({A'}_\iota, {B'}_\iota, {C'}_\iota, {D'}_\iota, J_\iota, c_\iota, s_\iota)$
if $J_0 = J_1$ **return** 1 **else return** 0

Fig. 5. The algorithms of CDL [16]

5.3 Our Construction

In Figure 6 we give our construction for a reputation system Π, as defined in Section 3, derived from the XS* scheme and the CDL scheme. We prove in the full paper that this satisfies the security requirements from Section 4.

6 Evaluation of our Construction

We first analyse the security of our construction, and then evaluate the efficiency. We prove Theorems 1–6 and correctness in the full version of this paper [21].

In the proof of Theorem 1, we show that if an adversary $\mathcal{A}$ can break the Unforgeability of Reputation experiment for our construction then we can build an adversary $\mathcal{A}'$ that breaks the q-SDH assumption. $\mathcal{A}'$ uses the q-SDH instance input to simulate $gpk, osk,$ **usk** for $\mathcal{A}$ so that they are identically distributed to in the experiment, but γ in the isk is also the secret value in the problem instance. $\mathcal{A}'$ also simulates responses the `AllocateRep` oracle to $\mathcal{A}$ using the problem instance and by programming the random oracle. $\mathcal{A}'$ then uses the signature (I, r, t, Σ) output by $\mathcal{A}$ to output a valid solution to the q-SDH problem.

Theorem 1 (Unforgeability of Reputation). *Assuming the random oracle model, and the q-SDH assumption, our reputation system Π satisfies Unforgeability of Reputation.*

The proofs of the following Theorems 2 and 3, are similar to the proofs of Traceability/ Non-Frameability and Anonymity for the XS scheme [18]. We have

$\mathsf{Setup}(k, \mathcal{R}, \hat{r}, \mathcal{U}, \mathsf{Aggr})$

Generate (gpp_1, gpp_2) as above , $(gpk_1, isk_1, osk) \leftarrow \text{XSKeyGen*}(gpp_1), (gpk_2, isk_2) \leftarrow \text{CDLKeyGen}(gpp_2)$
$gpk \leftarrow (gpk_1, gpk_2), isk \leftarrow (isk_1, isk_2)$
$\forall i \in \mathcal{U} \quad \mathbf{usk}_1[i] \leftarrow \text{XSJoin*}(isk_1, i, gpk_1), \mathbf{usk}_2[i] \leftarrow \text{CDLJoin}(i, isk_2, gpk_2), \mathbf{usk}[i] \leftarrow (\mathbf{usk}_1[i], \mathbf{usk}_2[i])$
$\mathcal{L} \leftarrow \emptyset, \mathcal{F} \leftarrow \emptyset, \mathcal{ID} \leftarrow \emptyset, \forall i \in \mathcal{U}, \mathbf{r}[i] \leftarrow \hat{r} \quad \mathbf{return}\ (gpk, isk, osk, \mathbf{usk}, \mathbf{r}, \mathcal{L}, \mathcal{F}, \mathcal{ID})$

$\mathsf{AllocateReputation}(gpk, isk, i, \mathbf{usk}[i], t, \mathbf{r}[i], \mathcal{ID})$

$(\omega, \tilde{Z}) \leftarrow \text{XSUpdate*}(t, \mathbf{r}[i], \mathbf{usk}_1[i], isk_1, gpk_1)$
$\mathcal{ID} \leftarrow \mathcal{ID} \cup (i, \mathbf{r}[i], t, \tilde{Z}) \quad \mathbf{return}\ (\omega, \mathbf{r}[i], \mathcal{ID})$

$\mathsf{PostItem}(gpk, I, (Z, x, y), r, t, \omega)$

$\mathbf{if}\ \hat{t}(\omega, WG_2^x) \neq \hat{t}(\mathcal{H}_1(r, t), G_2) \quad \mathbf{return}\ \perp$
$\tilde{Z} \leftarrow Z \cdot \omega, u\tilde{s}k \leftarrow (\tilde{Z}, x, y)$
$\mathbf{return}\ \Omega \leftarrow \text{XSSign*}(I, u\tilde{s}k, gpk_1, r, t)$

$\mathsf{CheckItem}(gpk, I, r, t, \Omega)$

$\mathbf{return}\ b \leftarrow \text{XSVerify*}(I, r, t, \Omega, gpk_1)$

$\mathsf{SendFB}(gpk, \mathbf{usk}[i], (I, r, t, \Omega), fb)$

$\mathbf{return}\ \Phi \leftarrow \text{CDLSign}((I, r, t, \Omega), fb, \mathbf{usk}_2[i], gpk_2)$

$\mathsf{VerifyFB}(gpk, (I, r, t, \Omega), fb, \Phi)$

$\mathbf{return}\ b \leftarrow \text{CDLVerify}((I, r, t, \Omega), fb, \Phi, gpk_2)$

$\mathsf{LinkFB}(gpk, (I, r, t, \Omega), fb_0, \Phi_0, fb_1, \Phi_1)$

$\mathbf{return}\ b \leftarrow \text{CDLLink}((I, r, t, \Omega), (fb_0, \Phi_0), (fb_1, \Phi_1), gpk_2)$

$\mathsf{ReceiveFB}(gpk, osk, ((I, r, t, \Omega), fb, \Phi), \mathbf{r}, \mathcal{L}, \mathcal{F}, \mathcal{ID})$

$\mathbf{if}\ \mathsf{VerifyFB}(gpk, (I, r, t, \Omega), fb, \Phi) = 0 \quad \mathbf{return}\ \perp$
$\mathbf{if}\ \exists (I, r, t, \Omega), fb', \Phi') \in \mathcal{L}$ s.t. $\mathsf{LinkFB}(gpk, (I, r, t, \Omega), fb, \Phi, fb', \Phi') = 1 \quad \mathbf{return}\ \perp$
$\tilde{Z} \leftarrow \text{XSOpen*}(I, r, t, \Omega, osk, gpk_1)$, Find $(i, r, t, \tilde{Z}) \in \mathcal{ID}$, otherwise $\mathbf{return}\ \perp$
$\mathbf{r}[i] \leftarrow \mathsf{Aggr}(fb, i, \mathcal{F}, \mathbf{r}[i]), \mathcal{L} \leftarrow ((I, r, t, \Omega), fb, \Phi) \cup \mathcal{L}, \mathcal{F} \leftarrow (i, fb) \cup \mathcal{F} \quad \mathbf{return}\ (\mathbf{r}, \mathcal{L})$

Fig. 6. Our Reputation System, Π

adapted these proofs due to the modification in XS*, and as our model is static (users do not join or leave after the scheme begins). The proofs of Theorems 4, 5, 6 are similar to the simulation based proof of security of CDL [16]. It is clear due to the similarity of the security requirements for DAA schemes [16] and the security requirements of Soundness of Reputation, Anonymity of Feedback and Non-Frameability, that a reputation system that uses the CDL scheme will satisfy these requirements.

Theorem 2 (Traceability). *Assuming the random oracle model, and the joc q-SDH assumption, our reputation system Π satisfies Traceability.*

Theorem 3 (Unlinkability of User Behaviour). *Assuming the random oracle model, and the q-sdh assumption, our reputation system Π satisfies Unlinkability of User Behaviour.*

Theorem 4 (Soundness of Reputation). *Assuming the bilinear LRSW problem is hard in $(\mathbb{G}_1, \mathbb{G}_2)$, and the random oracle model, our reputation system Π satisfies Soundness of Reputation.*

Theorem 5 (Anonymity of Feedback). *Assuming the DDH assumption in $\mathbb{G}_1$, and the random oracle model, our reputation system Π satisfies Anonymity of Feedback.*

Theorem 6 (Non–frameability). *Assuming the DL assumption in $\mathbb{G}_1$, and the random oracle model, our reputation system Π satisfies Non–frameability.*

6.1 Efficiency

Computational cost. We focus on PostItem, CheckItem, and SendFB, because these are performed by users with less computational power. We note that ReceiveFeedback only needs to check all feedback for the same item, to ensure Soundness of Reputation, not all feedback. SendFB requires 7 exponentiations in $\mathbb{G}_1$, and 2 hash computations which is a low computational cost.

There is an extra cost, compared to [9,19], required to achieve Unlinkability of user behaviour, in PostItem and CheckItem. We note that in [9,19] whenever a user posts an item they must receive a new secret key from the managing authority, which is not required by our reputation system. Assuming pre–computation, PostItem requires 8 exponentiations in $\mathbb{G}_1$, 3 exponentiations in $\mathbb{G}_3$, and 1 hash computation. CheckItem requires 2 computations of $\hat{t}$, 10 exponentiations in $\mathbb{G}_1$, 2 exponentiations in $\mathbb{G}_2$, 2 exponentiations in $\mathbb{G}_3$, 2 hash computations.

Communication Overhead. Using updated parameters for curves that give 128 bit security [28] and point compression, the XS* signature Ω has length 432 bytes and the CDL signature Φ has length 336 bytes. Therefore the communication overhead when sending feedback with our construction is 768 bytes, compared to 624 bytes in [9] using the same curves. This is a relatively small increase given the additional security of Unlinkability for user behaviour achieved. Our communication overhead compares well to [19] where signatures have length $\mathcal{O}(k \log(n))$, as shown in [22], compared to our signatures of length $\mathcal{O}(k)$.

6.2 Conventional Attacks on Reputation Systems

There are several attacks outside the scope of this work such as: On-Off attacks, where adversaries behave honestly/dishonestly alternatively, Whitewashing attacks, where adversaries leave and rejoin to shed a bad reputation, Sybil attacks, where users give dishonest feedback, and Self Rating attacks, where adversaries positively rate a large number of their own items.

Sybil attacks are partly mitigated by the Soundness of Reputation requirement. A solution for Whitewashing and Sybil attacks could be to make joining a scheme expensive. Self Rating attacks could be mitigated by making all users give the feedback "*" on their own items that could be used to link to self ratings, or be punished by the CS. The Central Server can also punish authors of items that do not represent a valid transaction.

7 Conclusion

We have introduced and formally defined a new security model for centralised reputation systems, where user behaviour is and unlinkable. This represents a shift from previous models which aims at more accurately capturing the real-world requirements of reputation systems, used by many on a daily basis.

We have provided a concrete construction which satisfies the new security requirements with a low additional efficiency cost. As a next step, we are considering the extension of our model to allow for dynamic join of users, similarly to

[7], as well as a concrete implementation of the system to be used, for instance, by a car-pooling app.

References

1. Amazons third-party sellers ship record-breaking 2 billion items in 2014, but merchant numbers stay flat. `https://techcrunch.com/2015/01/05/amazon-third-party-sellers-2014/`. [Online; accessed 1st-April-2019].
2. Travis kalanick says uber has 40 million monthly active riders. `https://techcrunch.com/2016/10/19/travis-kalanick-says-uber-has-40-million-monthly-active-riders/`. [Online; accessed 1st-April-2019].
3. Elli Androulaki, Seung Geol Choi, Steven M. Bellovin, and Tal Malkin. Reputation systems for anonymous networks. In *International Symposium on Privacy Enhancing Technologies*, pages 202–218. Springer, 2008.
4. Mihir Bellare, Daniele Micciancio, and Bogdan Warinschi. Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general assumptions. In Eli Biham, editor, *EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 614–629, Warsaw, Poland, May 4–8, 2003. Springer, Heidelberg, Germany.
5. Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In V. Ashby, editor, *ACM CCS 93*, pages 62–73, Fairfax, Virginia, USA, November 3–5, 1993. ACM Press.
6. Mihir Bellare, Haixia Shi, and Chong Zhang. Foundations of group signatures: The case of dynamic groups. In *Cryptographers Track at the RSA Conference*, pages 136–153. Springer, 2005.
7. Mihir Bellare, Haixia Shi, and Chong Zhang. Foundations of group signatures: The case of dynamic groups. In Alfred Menezes, editor, *CT-RSA 2005*, volume 3376 of *LNCS*, pages 136–153, San Francisco, CA, USA, February 14–18, 2005. Springer, Heidelberg, Germany.
8. John Bethencourt, Elaine Shi, and Dawn Song. Signatures of reputation. In Radu Sion, editor, *FC 2010*, volume 6052 of *LNCS*, pages 400–407, Tenerife, Canary Islands, Spain, January 25–28, 2010. Springer, Heidelberg, Germany.
9. Johannes Blömer, Jakob Juhnke, and Christina Kolb. Anonymous and publicly linkable reputation systems. In Rainer Böhme and Tatsuaki Okamoto, editors, *FC 2015*, volume 8975 of *LNCS*, pages 478–488, San Juan, Puerto Rico, January 26–30, 2015. Springer, Heidelberg, Germany.
10. Dan Boneh and Xavier Boyen. Short signatures without random oracles. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 56–73, Interlaken, Switzerland, May 2–6, 2004. Springer, Heidelberg, Germany.
11. Dan Boneh and Xavier Boyen. Short signatures without random oracles and the SDH assumption in bilinear groups. *Journal of Cryptology*, 21(2):149–177, April 2008.
12. Jonathan Bootle, Andrea Cerulli, Pyrros Chaidos, Essam Ghadafi, and Jens Groth. Foundations of fully dynamic group signatures. In Mark Manulis, Ahmad-Reza Sadeghi, and Steve Schneider, editors, *ACNS 16*, volume 9696 of *LNCS*, pages 117–136, Guildford, UK, June 19–22, 2016. Springer, Heidelberg, Germany.
13. Ernest F. Brickell, Jan Camenisch, and Liqun Chen. Direct anonymous attestation. In Vijayalakshmi Atluri, Birgit Pfitzmann, and Patrick McDaniel, editors, *ACM CCS 04*, pages 132–145, Washington D.C., USA, October 25–29, 2004. ACM Press.

14. Jan Camenisch, Liqun Chen, Manu Drijvers, Anja Lehmann, David Novick, and Rainer Urian. One TPM to bind them all: Fixing TPM 2.0 for provably secure anonymous attestation. In *2017 IEEE Symposium on Security and Privacy, SP*, pages 901–920. IEEE, 2017.
15. Jan Camenisch, Manu Drijvers, and Anja Lehmann. Anonymous attestation using the strong diffie hellman assumption revisited. In *International Conference on Trust and Trustworthy Computing*, pages 1–20. Springer, 2016.
16. Jan Camenisch, Manu Drijvers, and Anja Lehmann. Universally composable direct anonymous attestation. In Chen-Mou Cheng, Kai-Min Chung, Giuseppe Persiano, and Bo-Yin Yang, editors, *PKC 2016, Part II*, volume 9615 of *LNCS*, pages 234–264, Taipei, Taiwan, March 6–9, 2016. Springer, Heidelberg, Germany.
17. David Chaum and Eugène van Heyst. Group signatures. In Donald W. Davies, editor, *EUROCRYPT'91*, volume 547 of *LNCS*, pages 257–265, Brighton, UK, April 8–11, 1991. Springer, Heidelberg, Germany.
18. Cécile Delerablée and David Pointcheval. Dynamic fully anonymous short group signatures. In *Progress in Cryptology-VIETCRYPT 2006*, pages 193–210. Springer, 2006.
19. Ali El Kaafarani, Shuichi Katsumata, and Ravital Solomon. Anonymous reputation systems achieving full dynamicity from lattices. In *Twenty-Second International Conference on Financial Cryptography and Data Security*, forthcoming.
20. Lydia Garms, Keith Martin, and Siaw-Lynn Ng. Reputation schemes for pervasive social networks with anonymity. In *Proceedings of the fifteenth International Conference on Privacy, Security and Trust (PST 2017), IEEE*, 2017.
21. Lydia Garms and Elizabeth A. Quaglia. A new approach to modelling centralised reputation systems. Cryptology ePrint Archive, Report 2019/453, 2019. `https://eprint.iacr.org/2019/453`.
22. San Ling, Khoa Nguyen, Huaxiong Wang, and Yanhong Xu. Lattice-based group signatures: Achieving full dynamicity with ease. In *International Conference on Applied Cryptography and Network Security*, pages 293–312. Springer, 2017.
23. Anna Lysyanskaya, Ronald L Rivest, Amit Sahai, and Stefan Wolf. Pseudonym systems. In *International Workshop on Selected Areas in Cryptography*, pages 184–199. Springer, 1999.
24. Félix Gómez Mármol and Gregorio Martínez Pérez. Security threats scenarios in trust and reputation models for distributed systems. *Computers & Security*, 28(7):545–556, 2009.
25. Siaw-Lynn Ng, Keith Martin, Liqun Chen, and Qin Li. Private reputation retrieval in public - a privacy-aware announcement scheme for vanets. *IET Information Security, DOI: 10.1049/iet-ifs.2014.0316*, 2016.
26. Elan Pavlov, Jeffrey S Rosenschein, and Zvi Topol. Supporting privacy in decentralized additive reputation systems. In *International Conference on Trust Management*, pages 108–119. Springer, 2004.
27. Ronald Petrlic, Sascha Lutters, and Christoph Sorge. Privacy-preserving reputation management. In *Proceedings of the 29th Annual ACM Symposium on Applied Computing*, SAC '14, pages 1712–1718, New York, NY, USA, 2014. ACM.
28. Michael Scott. Pairing implementation revisited. Cryptology ePrint Archive, Report 2019/077, 2019. `https://eprint.iacr.org/2019/077`.
29. Ennan Zhai, David Isaac Wolinsky, Ruichuan Chen, Ewa Syta, Chao Teng, and Bryan Ford. Anonrep: towards tracking-resistant anonymous reputation. In *13th USENIX Symposium on Networked Systems Design and Implementation (NSDI 16)*, pages 583–596. USENIX Association, 2016.