This is the updated full version of the paper that appeared in FC'15 with the same title.

Anonymous and Publicly Linkable Reputation Systems

Johannes Blömer * Jakob Juhnke † Christina Kolb *

March 13, 2015

University of Paderborn, Germany
{bloemer,jakob.juhnke,christina.kolb}@uni-paderborn.de

Abstract

We consider reputation systems where users are allowed to rate products that they purchased previously. To obtain trustworthy reputations, they are allowed to rate these products only once. As long as they do so, the users stay anonymous. Everybody is able to detect users deviating from the rate-products-only-once policy and the anonymity of such dishonest users can be revoked by a system manager. In this paper we present formal models for such reputation systems and their security. Based on group signatures we design an efficient reputation system that meets all our requirements.

Keywords: Reputation, trust, group signatures, anonymity, linkability, verifier-local revocation, traceability, strong-exculpability

1 Introduction

Reputation systems are an increasingly popular tool to give providers and customers valuable information about previous transactions. To provide trustworthy, reliable, and honest ratings there is a need for anonymous reputation systems that also guarantee that customers rate products only once. To further increase trust in the system, everyone - even outsiders - should be able to verify the validity of ratings. In this paper, we propose models for secure and anonymous reputation systems and give an efficient construction of such a system.
Some of the properties for reputation systems stated above have been studied in the context of group signatures, as defined in [4] for the static and in [5] for the dynamic case. However, the concept of group signatures does not meet all the requirements for reputation systems. In

*This author was partially supported by the German Research Foundation (DFG) within the Collaborative Research Centre On-The-Fly Computing (SFB 901).
†This author was supported by the International Graduate School "Dynamic Intelligent Systems".

particular, reputation systems do not consist of a single group of users. Rather one can think of reputation systems as a family of group signature schemes - one for each product.
Moreover, we may have providers with several products. Hence, when looking at security and anonymity group signature schemes for different products can not be considered in isolation. Finally, known constructions of group signatures do not provide all properties that we need for a secure and anonymous reputation system and do not provide them simultaneously.

Our Contribution. We define models for secure and anonymous reputation systems and give a first construction of such a system based on group signature schemes. We use the terms rating and message synonymously. Our construction provides anonymity, traceability, strong-exculpability, verifier-local revocation, and public linkability. Anonymity means that signatures of honest users are indistinguishable. Traceability means that it is impossible for any set of colluding users to create ratings that can not be traced back to a user of the system. Strong-exculpability means that nobody can produce signatures on behalf of honest users. A system has verifier-local revocation, if revocation messages only have to be sent to signature verifiers, but not to individual signers. Public linkability requires that anyone can decide whether or not two ratings for the same product were created by the same user, i.e. no secret key is required to link messages. Note that public linkability implies that users can only stay anonymous as long as they rate products just once. As a remark, it is well known how to realize the described properties in the context of group signatures, although not necessarily simultaneously.
Our construction of a reputation system is based on the group signature scheme by Boneh, Boyen, and Shacham [6] (BBS) and the dynamic version of the scheme presented by Delerablée and Pointcheval [11]. These schemes already give us anonymity, traceability, and strong-exculpability. To achieve verifier-local revocation we modify a technique by [26]. With the same technique we achieve public linkability. Note that anonymity of group signatures does not imply anonymity in our reputation system. This is due to the fact that providers control the groups corresponding to several products. Hence, they may combine information for different groups to violate anonymity. To prevent this, we need a system manager that contributes a trustworthy component to each group public key. In Section 3 we present a formal model for reputation systems. The security of our system can be shown in the random oracle model and is based on the Decision Linear Assumption and the q-SDH Assumption in bilinear groups. Figure 1 illustrates informally the architecture of our reputation system.

Related Work. Reputation systems are a popular research topic in economics and computer science, see for example [2, 10, 12, 13, 19, 20]. Although privacy, i.e. anonymity and security, i.e. unforgeability, have been identified as key properties of reputation systems, no generally accepted privacy and security definitions for reputation systems have emerged. Definitions of anonymity based on differential privacy have been proposed in [10, 12, 28]. These are restricted to special reputation functions. In [2, 21, 25] cryptography has been proposed as a methodology to achieve anonymity in reputation systems, albeit without providing detailed definitions. In contrast to this, (anonymous) group signatures have been well studied in cryptography and formal security models exist. Important techniques to design group signature schemes were first described by Ateniese et al. [3]. For the case of static groups formal definitions of security were first given by Bellare, Micciancio and Warinschi [4], for dynamic groups by Bellare, Shi and Zhang [5]. Both works provide frameworks to construct group signature schemes. One

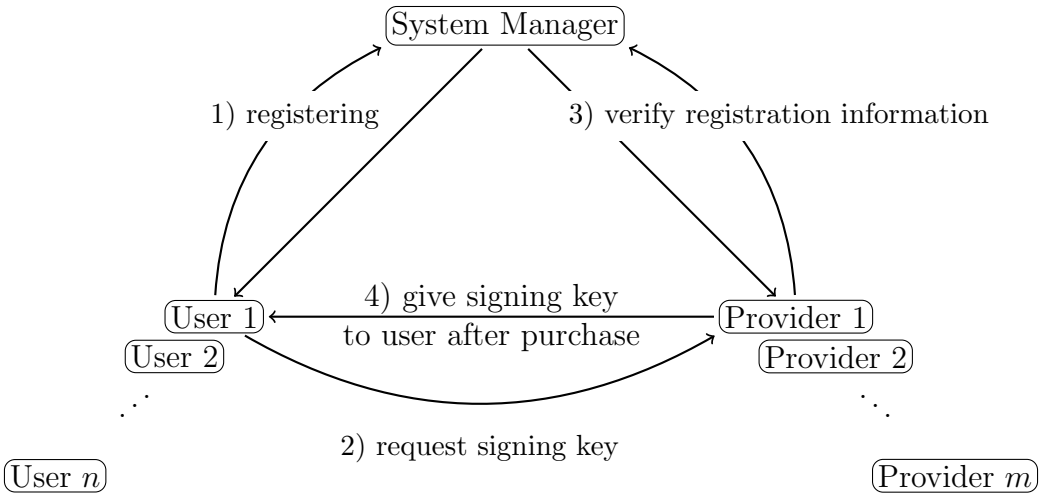

Figure 1: Informal architecture of our reputation system.

of the most efficient static schemes is that of Boneh, Boyen and Shacham [6]. Schemes with verifier-local revocation include [8, 26], linkable, though not publicly linkable, group signature schemes include [18, 15, 24]. In the context of ring signatures different definitions of linkability have been considered before, for example in [16, 9, 29, 23]. Our definition of public linkability is based on the definition given in [16].

2 Notation

Throughout this paper we will use the following notations. If $\mathcal{S}$ is a set, then $|\mathcal{S}|$ denotes its size. An empty string is denoted by ε. We distinguish three assignment operators. If x is a variable and y is an expression, then $x := y$ denotes the assignment of the value of y to the variable x. If $\mathcal{S}$ is a finite set, then we write $x \xleftarrow{\$} \mathcal{S}$ to indicate that an element x of $\mathcal{S}$ is picked uniformly at random. If $\mathcal{A}$ is an (probabilistic) algorithm running on inputs $y_1, y_2, \ldots$, then $x \longleftarrow \mathcal{A}(y_1, y_2, \ldots)$ denotes the operation of assigning the output of $\mathcal{A}$ to the variable x. The set of all possible outputs of an algorithm $\mathcal{A}$ with input $y_1, y_2, \ldots$ we denote by $[\mathcal{A}(y_1, y_2, \ldots)]$. If $\mathcal{A}$ and $\mathcal{B}$ are interactive algorithms, then the set of all possible outputs of the interactive algorithms $\mathcal{A}$ and $\mathcal{B}$ we denote by $[\mathcal{A}(y_1, y_2, \ldots)] \times [\mathcal{B}(z_1, z_2, \ldots)]$. Running an algorithm $\mathcal{A}$ with inputs $y_1, y_2, \ldots$ and access to the oracles $\mathcal{O}_1, \mathcal{O}_2, \ldots$ we denote by $\mathcal{A}(y_1, y_2, \ldots : \mathcal{O}_1, \mathcal{O}_2, \ldots)$.

3 A Model for Reputation Systems

In this section we provide a model for reputation systems. This model is based on the model for dynamic group signature schemes by Bellare, Shi, and Zhang [5]. Therefore, we will use the same notation for the authorities, algorithms and security properties as Bellare, Shi, and Zhang.

3.1 Algorithms

A reputation system consists of one authority called the group manager, a set of authorities called the key issuers, and a set of users. The group manager is assumed to be honest, provides the group manager's public key *gmpk* and is able to trace group members. Every key issuer provides *items* with corresponding item-based public keys $ipk[item]$, which will be used by the group members to rate/vote a specific item. Users have unique identities $i \in \mathbb{N}$ and may become group members by registering at the group manager.
The specification of a reputation system is a tuple $\mathcal{RS} = (\text{KeyGen}_{GM}, \text{KeyGen}_{KI}, \text{KeyGen}_{U}, \text{Register}_{GM}, \text{Register}_{U}, \text{Join}, \text{Issue}, \text{Sign}, \text{Verify}, \text{Open}, \text{Link}, \text{Revoke})$ of polynomial-time algorithms. Their functionality is described as follows.
$\mathbf{KeyGen_{GM}()}$: This randomized algorithm is run in the setup phase by the group manager to create the public key *gmpk* and the secret key *gmsk*. The secret key *gmsk* contains elements which allow tracing of group members and the creation of revocation tokens.
$\mathbf{KeyGen_{KI}(item)}$: This randomized algorithm is run by a key issuer for every *item* he provides. For the given *item* this algorithm creates an *item*-based public key $ipk[item]$ and a corresponding *item*-based secret key $isk[item]$. The tuple $(item, ipk[item])$ is added to the *ItemList*.
$\mathbf{KeyGen_{U}(i)}$: This randomized algorithm is run to create the user's public and secret key pair $(upk[i], usk[i])$. The user's public key $upk[i]$ is used during the registration to the group, the corresponding secret key $usk[i]$ is used to create signatures.
$\mathbf{Register_{GM}(St_{GM}, M_{GM}), Register_{U}(St_{U}, M_{U})}$: These randomized interactive algorithms are run by the group manager and a user $i \in \mathbb{N}$, who wants to become a group member. During this protocol the user's public and secret key pair $(upk[i], usk[i])$ is chosen by using the KeyGen_U algorithm. If the group manager accepts, the tuple $(i, upk[i])$ is added to the registration table *reg*. The input parameters of the algorithms are some state information and a message, which was received from the communicating partner. It is assumed that the user starts the interaction.
$\mathbf{Join(St_{U}, M_{U}), Issue(St_{KI}, M_{KI})}$: These randomized interactive algorithms are run by a user $i \in \mathbb{N}$ and a key issuer. The input parameters of the algorithms are some state information and a message, which was received from the communicating partner. It is assumed that the user starts the interaction. The first message of the user must contain his public key $upk[i]$, an *item*, and his identity i. If Issue accepts, the key issuer sends a personal signing key for the given *item* $gsk[i, item]$ to the user and saves the tuple $(upk[i], gsk[i, item])$ in the identification list IL_{item} for the specified *item*.
$\mathbf{Sign(item, gmpk, ipk[item], gsk[i, item], usk[i], M)}$: This randomized algorithm is run by a user to create a signature for the specified *item*. Given an *item*, the group manager's public key *gmpk*, an *item*-based public key $ipk[item]$, the signing key for the given *item* of user i $gsk[i, item]$, the secret key of user i $usk[i]$, and a message M, Sign computes and outputs a signature σ on M under the given keys.
$\mathbf{Verify(item, gmpk, ipk[item], \mathcal{RL}, M, \sigma)}$: This deterministic algorithm can be run by any user, even by an outsider, having access to the public *ItemList*, the group manager's public key *gmpk*, the revocation list $\mathcal{RL}$, a message M and a candidate signature σ for M, to obtain a bit v. We say that σ is a *valid* signature of M with respect to the given keys, iff the bit v is 1.
$\mathbf{Open(gmpk, gmsk, M, \sigma)}$: This deterministic algorithm is run by the group manager to *open* signatures. Given the group manager's public key *gmpk*, the group manager's secret key *gmsk*, a message M and a signature σ, output the identity of the signer or `failure`.

Link($item, gmpk, ipk[item], (M', \sigma'), (M'', \sigma'')$): This deterministic algorithm can be run by any user, even by an outsider, having access to the public *ItemList*, the group manager's public key *gmpk* and two message-signature pairs $(M', \sigma'), (M'', \sigma'')$, to obtain a bit ℓ. If Link outputs $\ell = 1$, we call σ' and σ'' *publicly linkable* signatures.
Revoke($gmpk, gmsk, i$): This deterministic algorithm is run by the group manager to revoke signers in case of misuse. Given the group manager's public key *gmpk*, the group manager's secret key *gmsk* and the identity of the group member to revoke, compute the revocation token $grt[i]$ and add it to the public revocation list $\mathcal{RL}$.

Figure 2 illustrates the interaction of the described parties and the algorithms involved. It is not hard to see that the number of key issuers is not important in this model: a single key issuer has the same capabilities as a colluding set of key issuers. Therefore, in all formal definitions we will only consider the case that the number of key issuers is 1. Additionally, we assume that the signing keys from the key issuer given to a user are publicly verifiable, i.e. the correctness of keys can be checked using only public parameters.

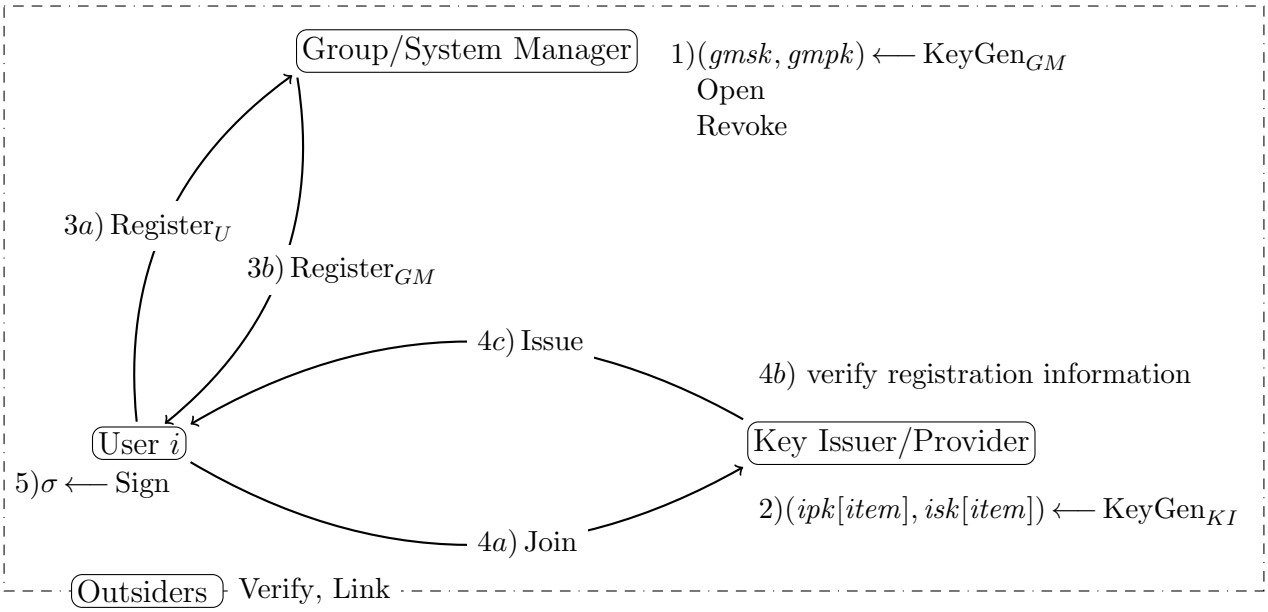

Figure 2: Interaction of the parties within a reputation system.

Correctness: A reputation system must satisfy the following correctness requirements:
For all $i \in \mathbb{N}$, all $item \in \{0,1\}^*$, all $(gmpk,\ gmsk) \in [\text{KeyGen}_{GM}]$, all $(ipk[item],\ isk[item]) \in [\text{KeyGen}_{KI}]$, all $(upk[i],\ usk[i]) \in [\text{Register}_{GM}] \times [\text{Register}_U]$, all $gsk[i, item] \in [\text{Join}] \times [\text{Issue}]$, and all $grt[i] \in [\text{Revoke}]$:

$$\text{Verify}(item, gmpk, ipk[item], \mathcal{RL}, M, \text{Sign}(item, gmpk, ipk[item], gsk[i, item], usk[i], M)) = 1 \iff grt[i] \notin \mathcal{RL},$$

$$\text{Open}(gmpk, gmsk, M, \text{Sign}(item, gmpk, ipk[item], gsk[i, item], usk[i], M)) = i$$

and

$$\text{Link}(item, gmpk, ipk[item], (M', \text{Sign}(item, gmpk, ipk[item], gsk[i, item], usk[i], M')), \\ (M'', \text{Sign}(item, gmpk, ipk[item], gsk[i, item], usk[i], M''))) = 1.$$

Informally, that means

1. honestly created signatures of non-revoked users will be accepted by the Verify algorithm,
2. honestly created signatures can be traced back to the correct signer,
3. two different signatures for the same *item* created by a single user will be detected by the Link algorithm.

3.2 Security Notions

To model the different attack capabilities of an adversary, we introduce certain oracles, which will be used in the definitions of security. The oracle definitions given in Figure 3 are based on [5]. Therefore, we assume that a security experiment has run $\text{KeyGen}_{GM}()$ to obtain (*gmpk*, *gmsk*), and manages the global sets $\mathcal{HU}$, $\mathcal{CU}$, $\mathcal{RU}$, $\mathcal{JIU}$, $\mathcal{GS}$, *reg* and *ItemList*. Except *ItemList* and *reg* all sets are only used within the formal definitions of Figure 3 and Figure 4. With $\mathcal{HU}$ we denote the set of honest users, with $\mathcal{CU}$ the set of corrupted users. The set $\mathcal{RU}$ contains all identities of users that currently engage in the registration protocol. The set $\mathcal{JIU}$ contains all identities of users that currently engage in the join-issue protocol. With $\mathcal{GS}$ we denote the set of queried signatures. All sets are assumed to be initially empty.

AddU(i): To add honest users to the group the adversary can call this *add user* oracle with an identity $i \in \mathbb{N}$ as argument. The oracle adds i to the set of honest users and executes the registration protocol by running Register_{GM} and Register_{U}. When Register_{GM} accepts, the tuple $(i, upk[i])$ is stored in the registration table *reg*. When Register_{U} accepts, the pair $(usk[i], upk[i])$ is the key pair of user i. The oracle returns $upk[i]$ to the adversary.

AddItem(*item*): An adversary can add *item*s by using this *add item* oracle. The oracle then runs the KeyGen_{KI} algorithm to obtain a secret and a public key for the specified *item*. Afterwards, the *item* is added to the *ItemList* and the public key of the *item* is returned to the adversary.

USK(i): To get the secret key $usk[i]$ of an honest user $i \in \mathbb{N}$ an adversary can call the *user secret key* oracle with an identity i as argument. Then the user i is added to $\mathcal{CU}$.

GSK(i, *item*): To get the secret signing key $gsk[i, item]$ of a corrupted user $i \in \mathbb{N}$ for a specified *item*, an adversary can call the *signing key* oracle with an identity i and an already existing *item* as arguments. If no signing key is found, the oracle generates a new one.

RevU(i): To get the *revocation token* of an honest user $i \in \mathbb{N}$ an adversary can call this *revoke user* oracle with an identity i as argument. Then the revocation token is added to $\mathcal{RL}$ and returned to the adversary.

GSig(i, *item*, M): An adversary can use the *signing* oracle to obtain a valid signature for the message M with respect to the signing key of user $i \in \mathbb{N}$, and the *item*-based public key $ipk[item]$. The queried signature is added to $\mathcal{GS}$.

SndToKI(i, *item*, $upk[i]$, M_{KI}): After corruption of user $i \in \mathbb{N}$, the adversary can use this *send to key issuer* oracle to engage in a key issuing protocol with the key issuer. The adversary provides the *item* and the public key of user i for which he wants to get a secret signing

key. Furthermore, the message M_{KI} is sent to the key issuer. The oracle honestly executes the Issue protocol and computes a response to M_{KI}. If Issue accepts the communication, the user's secret signing key is saved in the identification list IL_{item} and M_U contains the tuple $(upk[i], gsk[i, item])$.
SndToGM(i, M_{GM}): Similarly to the SndToKI oracle, the *send to group manager* oracle can be used by an adversary to engage in a registration protocol with the honest group manager. The adversary provides an identity $i \in \mathbb{N}$ and a message M_{GM} sent to the group manager. The oracle executes honestly the Register_{GM} protocol and saves $(i, upk[i])$ in the registration table $reg[i]$, iff Register_{GM} accepts. The user i is added to $\mathcal{CU}$.
WItemList($item, ipk$): An adversary can use the *write to item list* oracle to manipulate the *item*-based public key of the specified *item*. If $ipk = \varepsilon$ the *item* is deleted from the list. Otherwise, the specified public key is set.
WIdentList($item, i, upk[i], gsk$): Using the *write to identification list* oracle an adversary can modify the secret signing key of user $i \in \mathbb{N}$ for the specified *item*. If $gsk = \varepsilon$ the key information about user i is deleted from the list.
Open($item, M, \sigma$): The *opening* oracle can be used by the adversary with a message M, a signature σ and an *item* as arguments to get the output of the Open algorithm.

Using the oracle definitions from Figure 3 we can define the security experiments. In our reputation system we need anonymity, traceability, public linkability and strong-exculpability. The anonymity and traceability experiments are based on [5], the public linkability experiment is based on [16] and the strong-exculpability experiment is based on [22, 3, 5]. The experiments are defined in Figure 4.

Definition 3.1:
Let $\mathcal{RS}$ be a reputation system. We denote the advantage of an algorithm $\mathcal{A}$ in solving $\text{Exp}^{\text{anon-b}}_{\mathcal{A},\mathcal{RS}}(k)$ by

$$\text{Adv}\,\text{Exp}^{\text{anon-b}}_{\mathcal{A},\mathcal{RS}}(k) \stackrel{\text{def}}{=} \left|\Pr\left[\text{Exp}^{\text{anon-1}}_{\mathcal{A},\mathcal{RS}}(k) = 1\right] - \Pr\left[\text{Exp}^{\text{anon-0}}_{\mathcal{A},\mathcal{RS}}(k) = 1\right]\right|.$$

The probability is over the random bits of $\mathcal{A}$, as well as the random bits used in the experiment. We call $\mathcal{RS}$ $(t, \varepsilon)-$*anonymous*, iff for every algorithm $\mathcal{A}$ running in time at most t the advantage $\text{Adv}\,\text{Exp}^{\text{anon-b}}_{\mathcal{A},\mathcal{RS}}(k)$ is at most ε.

The anonymity requirements can slightly be relaxed to an experiment where an adversary is not allowed to query the Open oracle. We denote this modification by CPA-anonymity and the anonymity experiment as defined in Figure 4 by CCA2-anonymity, analogously to [6].

Definition 3.2:
Let $\mathcal{RS}$ be a reputation system. We denote the advantage of an algorithm $\mathcal{A}$ in solving $\text{Exp}^{\text{publink}}_{\mathcal{A},\mathcal{RS}}(k)$ by

$$\text{Adv}\,\text{Exp}^{\text{publink}}_{\mathcal{A},\mathcal{RS}}(k) \stackrel{\text{def}}{=} \Pr\left[\text{Exp}^{\text{publink}}_{\mathcal{A},\mathcal{RS}}(k) = 1\right].$$

The probability is over the random bits of $\mathcal{A}$, as well as the random bits used in the experiment. We call $\mathcal{RS}$ $(t, \varepsilon)-$*public linkable*, iff for every algorithm $\mathcal{A}$ running in time at most t the advantage $\text{Adv}\,\text{Exp}^{\text{publink}}_{\mathcal{A},\mathcal{RS}}(k)$ is at most ε.

AddU(i): // everybody

If ($i \in \mathcal{HU} \cup \mathcal{CU}$) then return ε.
$\mathcal{HU} \coloneqq \mathcal{HU} \cup \{i\}$
$\mathrm{St}_U^i \coloneqq (gmpk, i)$
$M_U \coloneqq \varepsilon$
$\mathrm{St}_{GM}^i \coloneqq (gmpk, gmsk)$
$(\mathrm{St}_U^i, M_{GM}, \mathrm{mode}^i) \longleftarrow \mathrm{Register}_U(\mathrm{St}_U^i, M_U)$
while (mode^i = `continue`) do
  $(\mathrm{St}_{GM}^i, M_U, \mathrm{mode}^i) \longleftarrow \mathrm{Register}_{GM}(\mathrm{St}_{GM}^i, M_{GM})$
  If (mode^i = `accept`) then $reg[i] \coloneqq (i, upk[i])$
  $(\mathrm{St}_U^i, M_{GM}, \mathrm{mode}^i) \longleftarrow \mathrm{Register}_U(\mathrm{St}_U^i, M_U)$
return $upk[i]$

AddItem($item$): // everybody

If ($item \in ItemList$) then return ε.
$(ipk[item], isk[item]) \longleftarrow \mathrm{KeyGen}_{KI}(item)$
$ItemList \coloneqq ItemList \cup \{(item, ipk[item])\}$
return $ipk[item]$

USK(i): // corrupted users

If ($i \notin \mathcal{HU}$) then return ε.
$\mathcal{HU} \coloneqq \mathcal{HU} \setminus \{i\}$
$\mathcal{CU} \coloneqq \mathcal{CU} \cup \{i\}$
return $usk[i]$

GSK($i, item$): // corrupted users

If ($i \notin \mathcal{CU}$) then return ε.
If ($item \notin ItemList$) then return ε.
If ($(upk[i], \cdot) \notin IL_{item}$) then
  $\mathrm{St}_U^i \coloneqq (gmpk, ipk[item], upk[i], i)$
  $M_U \coloneqq \varepsilon$
  $\mathrm{St}_{KI}^i \coloneqq (gmpk, upk[i])$
  $(\mathrm{St}_U^i, M_{KI}, \mathrm{mode}^i) \longleftarrow \mathrm{Join}(\mathrm{St}_U^i, M_U)$
  while (mode^i = `continue`) do
    $(\mathrm{St}_{KI}^i, M_U, \mathrm{mode}^i) \longleftarrow \mathrm{Issue}(\mathrm{St}_{KI}^i, M_{KI})$
    If (mode^i = `accept`) then
      $IL_{item} \coloneqq IL_{item} \cup \{(upk[i], gsk[i, item])\}$
    $(\mathrm{St}_U^i, M_{KI}, \mathrm{mode}^i) \longleftarrow \mathrm{Join}(\mathrm{St}_U^i, M_U)$
return $gsk[i, item]$

RevU(i): // corrupted users

If ($i \notin \mathcal{HU}$) then return ε
$\mathcal{RL} \coloneqq \mathcal{RL} \cup \{grt[i]\}$
return $grt[i]$

Open($item, M, \sigma$): // everybody

If ($item \notin ItemList$) then return `failure`.
return $\mathrm{Open}(gmpk, gmsk, M, \sigma)$

SndToKI($i, item, upk[i], M_{KI}$): // corrupted users

If ($i \notin \mathcal{CU}$) then return ε.
If ($item \notin ItemList$) then return ε.
If ($i \notin \mathcal{JIU}$) then $\mathrm{St}_{KI}^i \coloneqq \varepsilon$
$\mathcal{JIU} \coloneqq \mathcal{JIU} \cup \{i\}$
$(\mathrm{St}_{KI}^i, M_U, \mathrm{mode}^i) \longleftarrow \mathrm{Issue}(\mathrm{St}_{KI}^i, M_{KI})$
If (mode^i = `accept`) then
  $(upk[i], gsk[i, item]) \coloneqq M_U$
  $IL_{item} \coloneqq IL_{item} \cup \{(upk[i], gsk[i, item])\}$
  $\mathcal{JIU} \coloneqq \mathcal{JIU} \setminus \{i\}$
If (mode^i = `deny`) then $\mathcal{JIU} \coloneqq \mathcal{JIU} \setminus \{i\}$
return (M_U, mode^i)

SndToGM(i, M_{GM}): // corrupted users

If ($i \in \mathcal{HU} \cup \mathcal{CU}$) then return ε.
If ($i \notin \mathcal{RU}$) then $\mathrm{St}_{GM}^i \coloneqq (gmpk, gmsk)$
$\mathcal{RU} \coloneqq \mathcal{RU} \cup \{i\}$
$(\mathrm{St}_{GM}^i, M_U, \mathrm{mode}^i) \longleftarrow \mathrm{Register}_{GM}(\mathrm{St}_{GM}^i, M_{GM})$
If (mode^i = `accept`) then
  $reg[i] \coloneqq (i, upk[i])$
  $\mathcal{CU} \coloneqq \mathcal{CU} \cup \{i\}$
  $\mathcal{RU} \coloneqq \mathcal{RU} \setminus \{i\}$
If (mode^i = `deny`) then $\mathcal{RU} \coloneqq \mathcal{RU} \setminus \{i\}$
return (M_U, mode^i)

WItemList($item, ipk$): // corrupted key issuer

$ItemList \coloneqq ItemList \setminus \{(item, ipk[item])\}$
If ($ipk \neq \varepsilon$) then $ItemList \coloneqq ItemList \cup \{(item, ipk)\}$
return 1

WIdentList($item, i, upk[i], gsk$): // corrupted key issuer

If ($item \notin ItemList$) then return 0.
If ($(i, upk[i]) \neq reg[i]$) return 0.
$IL_{item} \coloneqq IL_{item} \setminus \{(upk[i], gsk[i, item])\}$
If ($gsk \neq \varepsilon$) then $IL_{item} \coloneqq IL_{item} \cup \{(upk[i], gsk)\}$
return 1

GSig($i, item, M$): // everybody

If ($i \notin \mathcal{HU}$) then return ε
If ($item \notin ItemList$) then return ε
If ($(upk[i], \cdot) \notin IL_{item}$) then
  $\mathrm{St}_U^i \coloneqq (gmpk, ipk[item], upk[i], i)$
  $M_U \coloneqq \varepsilon$
  $\mathrm{St}_{KI}^i \coloneqq (gmpk, upk[i])$
  $(\mathrm{St}_U^i, M_{KI}, \mathrm{mode}^i) \longleftarrow \mathrm{Join}(\mathrm{St}_U^i, M_U)$
  while (mode^i = `continue`) do
    $(\mathrm{St}_{KI}^i, M_U, \mathrm{mode}^i) \longleftarrow \mathrm{Issue}(\mathrm{St}_{KI}^i, M_{KI})$
    If (mode^i = `accept`) then
      $IL_{item} \coloneqq IL_{item} \cup \{(upk[i], gsk[i, item])\}$
    $(\mathrm{St}_U^i, M_{KI}, \mathrm{mode}^i) \longleftarrow \mathrm{Join}(\mathrm{St}_U^i, M_U)$
$\sigma \longleftarrow \mathrm{Sign}(item, gmpk, ipk[item], gsk[i, item], usk[i], M)$
$\mathcal{GS} \coloneqq \mathcal{GS} \cup \{(item, i, M, \sigma)\}$
return σ

Figure 3: Oracle definitions.

Experiment $\mathrm{Exp}^{\mathrm{anon-b}}_{\mathcal{A},\mathcal{RS}}(k)$ // $b \in \{0,1\}$

$\mathcal{HU} \coloneqq \emptyset, \mathcal{CU} \coloneqq \emptyset, \mathcal{RL} \coloneqq \emptyset, \mathit{ItemList} \coloneqq \emptyset, \mathit{reg} \coloneqq \emptyset, \mathcal{RU} \coloneqq \emptyset, \mathcal{JIU} \coloneqq \emptyset, \mathcal{GS} \coloneqq \emptyset$
$(\mathit{gmsk}, \mathit{gmpk}) \longleftarrow \mathrm{KeyGen}_{GM}()$
$(i_0, i_1, \mathit{item}, M, \mathrm{St}) \longleftarrow \mathcal{A}(\mathit{gmpk}:$ AddU, USK, RevU, GSig, SndToGM, WItemList, WIdentList, Open, `choose`$)$
$\sigma \longleftarrow \mathrm{Sign}(\mathit{item}, \mathit{gmpk}, \mathit{ipk}[\mathit{item}], \mathit{gsk}[i_b, \mathit{item}], \mathit{usk}[i_b], M)$
$d \longleftarrow \mathcal{A}(\sigma, \mathrm{St}:$ AddU, USK, RevU, GSig, SndToGM, WItemList, WIdentList, Open, `guess`$)$
If $((\mathit{item}, i_0, \cdot, \cdot) \in \mathcal{GS}) \vee ((\mathit{item}, i_1, \cdot, \cdot) \in \mathcal{GS}) \vee (i_0 \notin \mathcal{HU}) \vee (i_1 \notin \mathcal{HU})$
$\vee(\mathit{grt}[i_0] \in \mathcal{RL}) \vee (\mathit{grt}[i_1] \in \mathcal{RL}) \vee (\mathcal{A}$ queried Open$(\mathit{item}, M, \sigma))$
then return 0
return d

Experiment $\mathrm{Exp}^{\mathrm{publink}}_{\mathcal{A},\mathcal{RS}}(k)$

$\mathcal{HU} \coloneqq \emptyset, \mathcal{CU} \coloneqq \emptyset, \mathcal{RL} \coloneqq \emptyset, \mathit{ItemList} \coloneqq \emptyset, \mathit{reg} \coloneqq \emptyset, \mathcal{RU} \coloneqq \emptyset, \mathcal{JIU} \coloneqq \emptyset, \mathcal{GS} \coloneqq \emptyset$
$(\mathit{gmsk}, \mathit{gmpk}) \longleftarrow \mathrm{KeyGen}_{GM}()$
$\{(\mathit{item}, m_i, \sigma_i)\}_{i=1}^{|\mathcal{CU}|+1} \longleftarrow \mathcal{A}(\mathit{gmpk}:$ AddItem, SndToKI, SndToGM$)$
If there exists an $i \in \{1, \ldots, |\mathcal{CU}|+1\}$ such that $\mathrm{Verify}(\mathit{item}, \mathit{gmpk}, \mathit{ipk}[\mathit{item}], \mathcal{RL}, m_i, \sigma_i) = 0$ then return 0
If there are $i, j \in \{1, \ldots, |\mathcal{CU}|+1\}$ with $i \neq j$ such that $\mathrm{Link}(\mathit{item}, \mathit{gmpk}, \mathit{ipk}[\mathit{item}], (m_i, \sigma_i), (m_j, \sigma_j)) = 1$ then return 0
return 1

Experiment $\mathrm{Exp}^{\mathrm{trace}}_{\mathcal{A},\mathcal{RS}}(k)$

$\mathcal{HU} \coloneqq \emptyset, \mathcal{CU} \coloneqq \emptyset, \mathcal{RL} \coloneqq \emptyset, \mathit{ItemList} \coloneqq \emptyset, \mathit{reg} \coloneqq \emptyset, \mathcal{RU} \coloneqq \emptyset, \mathcal{JIU} \coloneqq \emptyset, \mathcal{GS} \coloneqq \emptyset$
$(\mathit{gmsk}, \mathit{gmpk}) \longleftarrow \mathrm{KeyGen}_{GM}()$
$(\mathit{item}, m, \sigma) \longleftarrow \mathcal{A}(\mathit{gmpk}:$ AddU, AddItem, USK, GSK, RevU, GSig, SndToKI, SndToGM, Open$)$
If $(\mathrm{Verify}(\mathit{item}, \mathit{gmpk}, \mathit{ipk}[\mathit{item}], \mathcal{RL}, m, \sigma) = 0)$ then return 0
If $(\mathrm{Open}(\mathit{gmpk}, \mathit{gmsk}, m, \sigma) =$ `failure`$)$ then return 1
$i \longleftarrow \mathrm{Open}(\mathit{gmpk}, \mathit{gmsk}, m, \sigma)$
If $(i \in \mathcal{CU}) \vee ((\mathit{item}, i, m, \sigma) \in \mathcal{GS})$ then return 0
return 1

Experiment $\mathrm{Exp}^{\mathrm{str-ex}}_{\mathcal{A},\mathcal{RS}}(k)$

$\mathcal{HU} \coloneqq \emptyset, \mathcal{CU} \coloneqq \emptyset, \mathcal{RL} \coloneqq \emptyset, \mathit{ItemList} \coloneqq \emptyset, \mathit{reg} \coloneqq \emptyset, \mathcal{RU} \coloneqq \emptyset, \mathcal{JIU} \coloneqq \emptyset, \mathcal{GS} \coloneqq \emptyset$
$(\mathit{gmsk}, \mathit{gmpk}) \longleftarrow \mathrm{KeyGen}_{GM}()$
$(\mathit{item}, m, \sigma) \longleftarrow \mathcal{A}(\mathit{gmpk}:$ AddU, USK, RevU, GSig, SndToGM, WItemList, WIdentList, Open$)$
If $(\mathrm{Verify}(\mathit{item}, \mathit{gmpk}, \mathit{ipk}[\mathit{item}], \mathcal{RL}, m, \sigma) = 0)$ then return 0
If $(\mathrm{Open}(\mathit{gmpk}, \mathit{gmsk}, m, \sigma) =$ `failure`$)$ then return 0
$i \longleftarrow \mathrm{Open}(\mathit{gmpk}, \mathit{gmsk}, m, \sigma)$
If $(i \notin \mathcal{HU}) \vee ((\mathit{item}, i, m, \sigma) \in \mathcal{GS})$ then return 0
return 1

Figure 4: Experiment definitions.

Definition 3.3:
Let $\mathcal{RS}$ be a reputation system. We denote the advantage of an algorithm $\mathcal{A}$ in solving $\mathrm{Exp}^{\mathrm{trace}}_{\mathcal{A},\mathcal{RS}}(k)$ by

$$\mathrm{Adv}\,\mathrm{Exp}^{\mathrm{trace}}_{\mathcal{A},\mathcal{RS}}(k) \stackrel{\mathsf{def}}{=} \Pr\left[\mathrm{Exp}^{\mathrm{trace}}_{\mathcal{A},\mathcal{RS}}(k) = 1\right].$$

The probability is over the random bits of $\mathcal{A}$, as well as the random bits used in the experiment. We call $\mathcal{RS}$ $(t,\varepsilon)-$*traceable*, iff for every algorithm $\mathcal{A}$ running in time at most t the advantage $\mathrm{Adv}\,\mathrm{Exp}^{\mathrm{trace}}_{\mathcal{A},\mathcal{RS}}(k)$ is at most ε.

Definition 3.4:
Let $\mathcal{RS}$ be a reputation system. We denote the advantage of an algorithm $\mathcal{A}$ in solving $\mathrm{Exp}^{\mathrm{st-ex}}_{\mathcal{A},\mathcal{RS}}(k)$ by

$$\mathrm{Adv}\,\mathrm{Exp}^{\mathrm{str-ex}}_{\mathcal{A},\mathcal{RS}}(k) \stackrel{\mathsf{def}}{=} \Pr\left[\mathrm{Exp}^{\mathrm{str-ex}}_{\mathcal{A},\mathcal{RS}}(k) = 1\right].$$

The probability is over the random bits of $\mathcal{A}$, as well as the random bits used in the experiment. We call $\mathcal{RS}$ $(t, \varepsilon)-$*strong-exculpable*, iff for every algorithm $\mathcal{A}$ running in time at most t the advantage $\mathrm{Adv}\,\mathrm{Exp}^{\mathrm{str-ex}}_{\mathcal{A},\mathcal{RS}}(k)$ is at most ε.

3.2.1 Discussion:

The anonymity experiment $\mathrm{Exp}^{\mathrm{anon-b}}_{\mathcal{A},\mathcal{RS}}(k)$ asks an adversary to distinguish which of two group members signed a message for some *item*, where the identities, the message, and the *item* were chosen by the adversary. The adversary's attack capabilities are strong: it is possible to corrupt the key issuer and all but two users. These two users must be honest because otherwise the adversary could possibly link different signatures or use the revocation token of the users to determine their identities.
The strong-exculpability experiment $\mathrm{Exp}^{\mathrm{str-ex}}_{\mathcal{A},\mathcal{RS}}(k)$ asks an adversary to output a message-signature pair, for some *item* chosen by the adversary, which is valid and can be traced back to an honest user. We give an adversary the possibility to corrupt users and the key issuer, so the attack capabilities are very strong. Because the key issuer can always generate signing keys for non-existing users, we force the adversary to output a signature on behalf of an honest user.
The public linkability experiment $\mathrm{Exp}^{\mathrm{publink}}_{\mathcal{A},\mathcal{RS}}(k)$ asks an adversary to output message-signature pairs, for the same *item* chosen by the adversary, such that all pairs are valid and there are no two pairs that can be linked. The number of pairs must be one more than the number of users in the group. We allow the adversary to corrupt all users, but the key issuer has to be honest. If the key issuer is corrupted, then he can create signing keys for non-existing users. Hence, he can also create signatures which can not be linked to signatures created by the group members.
The traceability experiment $\mathrm{Exp}^{\mathrm{trace}}_{\mathcal{A},\mathcal{RS}}(k)$ asks an adversary to output a message-signature pair, for some *item* chosen by the adversary, which is valid but can not be traced back to a corrupted user. In this experiment the key issuer must be honest because he could generate signing keys for non-existing users.

Discussion: The defined experiments imply two different attack scenarios:

> In the first scenario, for anonymity and strong-exculpability, we allow an adversary to corrupt the key issuer and a set of users. One could argue, that there is an oracle missing to allow an adversary to send corrupted data to honest users in the join-issue protocol. But this functionality is covered by the SndToGM, WItemList, and WIdentList oracles. Due to the assumption that all signing keys are publicly verifiable by the users (as stated in Subsection 3.1), an honest user would only accept valid keys in the join-issue protocol. The same is implicitly done by our oracles and in the experiments. Hence, we omit such an oracle.
>
> In the second scenario, for public linkability and traceability, key issuers are assumed to be honest, whereas a set of users can be corrupted. In particular, this implies that users and key issuers are disjoint sets. The restriction to honest key issuers is necessary because a corrupted key issuer could generate secret keys for non-existing users. With an appropriate identity management this can be prevented and we could also allow corrupted key issuers in the experiments for public linkability and traceability.

An important issue is that of timing the operations. The key issuer may correlate transactions and ratings by their timing, thereby threatening the anonymity of users. Hence, our reputation

systems needs a mechanism to prevent such attacks. In [10], [21], and [17] different solutions to this problem are proposed, which can be incorporated into our construction.

4 Preliminaries

In this section we introduce the main building blocks for our reputation system.

4.1 Bilinear Maps

First we review some concepts related to bilinear maps, following the notation of [7]:

1. $\mathbb{G}_1$ and $\mathbb{G}_2$ are multiplicative groups of prime order p,
2. ψ is an isomorphism from $\mathbb{G}_2$ to $\mathbb{G}_1$,
3. g_1 is a generator of $\mathbb{G}_1$ and g_2 is a generator of $\mathbb{G}_2$, with $g_1 = \psi(g_2)$, and
4. e is a map $\mathrm{e}\colon \mathbb{G}_1 \times \mathbb{G}_2 \longrightarrow \mathbb{G}_T$ with the following properties:

   Bilinearity: for all $u \in \mathbb{G}_1$, $v \in \mathbb{G}_2$ and $a, b \in \mathbb{Z}_p$: $\mathrm{e}(u^a, v^b) = \mathrm{e}(u, v)^{ab}$

   Non-degeneracy: $\mathrm{e}(g_1, g_2) \neq 1_{\mathbb{G}_T}$.

We say the groups $(\mathbb{G}_1, \mathbb{G}_2)$ as described above are *bilinear groups*, iff the group operations in $\mathbb{G}_1$ and $\mathbb{G}_2$, the isomorphism ψ and the mapping e are efficiently computable.

4.2 Computational Assumptions

Here we introduce the computational assumptions we use to prove the security of our reputation system. Since these assumptions are standard, we can be brief.

Definition 4.1 - Decision Linear Problem – D-Linear1:
Let $\mathbb{G}$ be a cyclic group of prime order p. Given arbitrary generators $u, v, w \in \mathbb{G}$ and $u^\alpha, v^\beta, w^\gamma \in \mathbb{G}$, where $\alpha, \beta \xleftarrow{\$} \mathbb{Z}_p$, the Decision Linear Problem is to decide whether $\gamma = \alpha + \beta$.

Definition 4.2:
The advantage of an algorithm $\mathcal{A}$ in deciding the Decision Linear Problem is

$$\mathrm{Adv}_{\mathcal{A}}^{\mathrm{D-Linear1}} \stackrel{\mathsf{def}}{=} \left| \begin{array}{l} \Pr\left[\mathcal{A}(u, v, w, u^\alpha, v^\beta, w^{\alpha+\beta}) = 1 \colon u, v, w \xleftarrow{\$} \mathbb{G}, \alpha, \beta \xleftarrow{\$} \mathbb{Z}_p\right] \\ -\Pr\left[\mathcal{A}(u, v, w, u^\alpha, v^\beta, w^\gamma) = 1 \colon u, v, w \xleftarrow{\$} \mathbb{G}, \alpha, \beta, \gamma \xleftarrow{\$} \mathbb{Z}_p\right] \end{array} \right| .$$

The probability is over the uniform choices of the parameters $u, v, w, \alpha, \beta, \gamma$ and over the random bits of $\mathcal{A}$. We say that algorithm $\mathcal{A}$ (t, ε)-decides D-Linear1 if $\mathcal{A}$ runs in time at most t and $\mathrm{Adv}_{\mathcal{A}}^{\mathrm{D-Linear1}}$ is at least ε. We say that D-Linear1 (t, ε)-holds in $\mathbb{G}$ if no algorithm running in time t has advantage at least ε in solving D-Linear1 in $\mathbb{G}$.

Lemma 4.1 Random self-reducibility of D-Linear1:
Let $\mathbb{G}$ be a multiplicative group of prime order p and let $G := (u, v, w, u^a, v^b, w^c)$ be an instance of the Decision Linear Problem. Then we can construct another instance $H := (r, s, t, r^d, s^e, t^f)$ that is independent of G, but has the same distribution as G.

Proof. Choose the values $\alpha, \beta, \gamma, \delta, \varepsilon, \varphi \xleftarrow{\$} \mathbb{Z}_p$ and set $r := u^{\alpha\cdot\beta\cdot\varphi}$, $s := v^{\gamma\cdot\delta\cdot\varphi}$, $t := w^{\varphi}$, $r^d := (u^a)^{\alpha\cdot\beta\cdot\varepsilon\cdot\varphi}$, $s^e := (v^b)^{\gamma\cdot\delta\cdot\varepsilon\cdot\varphi}$, and $t^f := (w^c)^{\varepsilon\cdot\varphi}$. With $\alpha, \beta, \gamma, \delta, \varphi \xleftarrow{\$} \mathbb{Z}_p$ the first five components of H are distributed uniformly at random. For the exponents d, e and f holds $d = a\cdot\varepsilon$, $e = b\cdot\varepsilon$, and $f = c\cdot\varepsilon$. Hence, $f = d + e \Leftrightarrow c\cdot\varepsilon = a\cdot\varepsilon + b\cdot\varepsilon \Leftrightarrow c = a + b$, H is independent of G and H is distributed exactly as G. □

Definition 4.3 - q-Strong Diffie-Hellman Problem – q-SDH:
Let $\mathbb{G}_1, \mathbb{G}_2$ be two (multiplicative) groups of prime order p (where possibly $\mathbb{G}_1 = \mathbb{G}_2$), let ψ be an efficiently computable isomorphism from $\mathbb{G}_2$ to $\mathbb{G}_1$, let $g_2 \in \mathbb{G}_2$ be an arbitrary generator and let $g_1 = \psi(g_2)$. Given a $(q+2)$-tuple $(g_1, g_2, g_2^{\gamma}, g_2^{(\gamma^2)}, \ldots, g_2^{(\gamma^q)})$, the q-Strong Diffie-Hellman Problem is to output a pair $\left(g_1^{\frac{1}{x+\gamma}}, x\right)$, where $x \in \mathbb{Z}_p$.

Definition 4.4:
The advantage of an algorithm $\mathcal{A}$ in solving q-SDH in $(\mathbb{G}_1, \mathbb{G}_2)$ is

$$\mathrm{Adv}_{\mathcal{A}}^{\text{q-SDH}} \stackrel{\mathsf{def}}{=} \Pr\left[\mathcal{A}\left(g_1, g_2, g_2^{\gamma}, g_2^{(\gamma^2)}, \ldots, g_2^{(\gamma^q)}\right) = \left(g_1^{\frac{1}{x+\gamma}}, x\right) : g_2 \xleftarrow{\$} \mathbb{G}_2, g_1 = \psi(g_2), \gamma \xleftarrow{\$} \mathbb{Z}_p\right].$$

The probability is over the uniform choices of the parameters g_2, γ and over the random bits of $\mathcal{A}$. We say that algorithm $\mathcal{A}$ (t, ε)-solves q-SDH in $(\mathbb{G}_1, \mathbb{G}_2)$ if $\mathcal{A}$ runs in time at most t and $\mathrm{Adv}_{\mathcal{A}}^{\text{q-SDH}}$ is at least ε. We say that q-SDH (t, ε)-holds in $(\mathbb{G}_1, \mathbb{G}_2)$ if no algorithm running in time t has advantage at least ε in solving q-SDH in $(\mathbb{G}_1, \mathbb{G}_2)$.

Definition 4.5 - extended q-Strong Diffie-Hellman Problem – extended q-SDH:
Let $\mathbb{G}_1, \mathbb{G}_2$ be two (multiplicative) groups of prime order p (where possibly $\mathbb{G}_1 = \mathbb{G}_2$), let ψ be an efficiently computable isomorphism from $\mathbb{G}_2$ to $\mathbb{G}_1$, let $g_2 \in \mathbb{G}_2$ be arbitrary generator, let $g_1 = \psi(g_2)$ and $h \xleftarrow{\$} \mathbb{G}_1$. Given a $(q+3)$-tuple $\left(g_1, h, g_2, g_2^{\gamma}, g_2^{(\gamma^2)}, \ldots, g_2^{(\gamma^q)}\right)$, the extended q-Strong Diffie-Hellman Problem is to output a tuple $\left((g_1 \cdot h^y)^{\frac{1}{x+\gamma}}, x, y\right)$, where $x, y \in \mathbb{Z}_p$.

Definition 4.6:
The advantage of an algorithm $\mathcal{A}$ in solving extended q-SDH in $(\mathbb{G}_1, \mathbb{G}_2)$ is

$$\mathrm{Adv}_{\mathcal{A}}^{\text{ext-q-SDH}} \stackrel{\mathsf{def}}{=} \Pr\left[\mathcal{A}\left(g_1, h, g_2, g_2^{\gamma}, g_2^{(\gamma^2)}, \ldots, g_2^{(\gamma^q)}\right) = \left((g_1 \cdot h^y)^{\frac{1}{x+\gamma}}, x, y\right) : h \xleftarrow{\$} \mathbb{G}_1, g_2 \xleftarrow{\$} \mathbb{G}_2, \gamma \xleftarrow{\$} \mathbb{Z}_p, g_1 = \psi(g_2)\right].$$

The probability is over the uniform choices of the parameters h, g_2, γ and over the random bits of $\mathcal{A}$. We say that algorithm $\mathcal{A}$ (t, ε)-solves extended q-SDH in $(\mathbb{G}_1, \mathbb{G}_2)$ if $\mathcal{A}$ runs in time at most t and $\mathrm{Adv}_{\mathcal{A}}^{\text{ext-q-SDH}}$ is at least ε. We say that extended q-SDH (t, ε)-holds in $(\mathbb{G}_1, \mathbb{G}_2)$ if no algorithm running in time t has advantage at least ε in solving extended q-SDH in $(\mathbb{G}_1, \mathbb{G}_2)$.

Lemma 4.2:
Let $\mathbb{G}_1, \mathbb{G}_2$ be two cyclic groups of prime order p and let $\mathcal{A}$ be an algorithm that (t', ε')-solves extended q-SDH in $(\mathbb{G}_1, \mathbb{G}_2)$. Then there exists an algorithm $\mathcal{B}$ that (t, ε)-solves q-SDH in $(\mathbb{G}_1, \mathbb{G}_2)$, where $t = t' + \mathcal{O}(1)$.

Proof. Algorithm $\mathcal{B}$ is given an instance of a q-SDH problem $(g_1, g_2, g_2^{\gamma}, g_2^{(\gamma^2)}, \ldots, g_2^{(\gamma^q)})$. Now $\mathcal{B}$ chooses $\alpha \xleftarrow{\$} \mathbb{Z}_p$ and computes $h := g_1^{\alpha}$. Then the extended q-SDH instance $\left(g_1,\, h,\, g_2,\, g_2^{\gamma},\, g_2^{(\gamma^2)},\, \ldots,\, g_2^{(\gamma^q)}\right)$ is given to algorithm $\mathcal{A}$. $\mathcal{A}$ outputs a solution $\left((g_1 \cdot h^y)^{\frac{1}{x+\gamma}}, x, y\right)$ to this problem with advantage ε. Now $\mathcal{B}$ computes

$$c := (1 + \alpha \cdot y)^{-1} \quad \text{and} \quad \left((g_1 \cdot h^y)^{\frac{1}{x+\gamma}}\right)^c = (g_1 \cdot h^y)^{\frac{c}{x+\gamma}} = (g_1 \cdot g_1^{\alpha \cdot y})^{\frac{c}{x+\gamma}} = g_1^{\frac{(1+\alpha \cdot y)}{x+\gamma} \cdot \frac{1}{(1+\alpha \cdot y)}} = g_1^{\frac{1}{x+\gamma}}.$$

So $\mathcal{B}$ can output $\left(g_1^{\frac{1}{x+\gamma}}, x\right)$ as a solution to his q-SDH problem with advantage ε in time $t = t' + \mathcal{O}(1)$, as claimed. □

5 A Zero-Knowledge protocol for extended q-SDH – Intuition to the Reputation System

We give a zero-knowledge proof of knowledge to prove possession of a solution to an extended q-SDH problem. That means, the prover has to show that he knows a triple (A, x, y) such that $A^{x+\gamma} = g_1 \cdot h^y$, where $A, g_1, h \in \mathbb{G}_1$ and $x, y, \gamma \in \mathbb{Z}_p$ (here the γ is not known by the prover).
We assume to have an efficiently computable isomorphism ψ from $\mathbb{G}_2$ to $\mathbb{G}_1$ and an efficiently computable non-degenerative bilinear mapping $\mathrm{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$. The groups $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ are groups of prime order p. Furthermore, we assume that the Decision Linear Problem holds in $\mathbb{G}_1$.
The secret key of the prover is the triple (A, x, y), the public values are g_1, d, f, h, u, v, w, $C \in \mathbb{G}_1$ and $g_2, W \in \mathbb{G}_2$, where $g_2 \xleftarrow{\$} \mathbb{G}_2$, $W = g_2^{\gamma}$ for some (secret) $\gamma \in \mathbb{Z}_p$, $g_1 = \psi(g_2)$, $h, d, f, u, v, w \xleftarrow{\$} \mathbb{G}_1$, and $C = h^y$. The value A is determined by $A = (g_1 \cdot h^y)^{\frac{1}{x+\gamma}}$, but can not be computed by the prover because γ is unknown.
The triple (A, x, y) is a solution of the extended q-SDH problem, iff the equation $\mathrm{e}(A, g_2)^x \cdot \mathrm{e}(A, W) \cdot \mathrm{e}(h, g_2)^{-y} = \mathrm{e}(g_1, g_2)$ holds. This will be used in the protocol.

Protocol 5.1:

1. $\mathcal{P}$ chooses $\alpha, \beta, \mu \xleftarrow{\$} \mathbb{Z}_p$ and computes $T_1 := u^{\alpha}$, $T_2 := v^{\beta}$, $T_3 := A \cdot w^{\alpha+\beta}$, $T_4 := d^{\mu}$, $T_5 := f^{\mu+y}$, $\delta_1 := x \cdot \alpha$, and $\delta_2 := x \cdot \beta$. Now $\mathcal{P}$ and $\mathcal{V}$ undertake a proof of knowledge of values $(\alpha, \beta, x, y, \mu, \delta_1, \delta_2)$.

2. $\mathcal{P}$ chooses $r_{\alpha}, r_{\beta}, r_x, r_y, r_{\mu}, r_{\delta_1}, r_{\delta_2} \xleftarrow{\$} \mathbb{Z}_p$, computes $R_1 := u^{r_{\alpha}}$, $R_2 := v^{r_{\beta}}$, $R_3 := \mathrm{e}(T_3, g_2)^{r_x} \cdot \mathrm{e}(w, W)^{-r_{\alpha}-r_{\beta}} \cdot \mathrm{e}(w, g_2)^{-r_{\delta_1}-r_{\delta_2}} \cdot \mathrm{e}(h, g_2)^{-r_y}$, $R_4 := T_1^{r_x} \cdot u^{-r_{\delta_1}}$, $R_5 := T_2^{r_x} \cdot v^{-r_{\delta_2}}$, $R_6 := d^{r_{\mu}}$, and $R_7 := f^{r_{\mu}+r_y}$, and sends $(T_1, T_2, T_3, T_4, T_5, R_1, R_2, R_3, R_4, R_5, R_6, R_7)$ to $\mathcal{V}$.

3. The verifier $\mathcal{V}$ chooses $c \xleftarrow{\$} \mathbb{Z}_p$ as a challenge and sends c to the prover $\mathcal{P}$.

4. The prover computes $s_\alpha := r_\alpha + c \cdot \alpha$, $s_\beta := r_\beta + c \cdot \beta$, $s_x := r_x + c \cdot x$, $s_y := r_y + c \cdot y$, $s_\mu := r_\mu + c \cdot \mu$, $s_{\delta_1} := r_{\delta_1} + c \cdot \delta_1$, $s_{\delta_2} := r_{\delta_2} + c \cdot \delta_2$ and sends them to the verifier $\mathcal{V}$.

5. The verifier $\mathcal{V}$ checks the following equations: $R_1 \stackrel{?}{=} u^{s_\alpha} \cdot T_1^{-c}$, $R_2 \stackrel{?}{=} v^{s_\beta} \cdot T_2^{-c}$, $R_3 \stackrel{?}{=} \mathrm{e}(T_3, g_2)^{s_x} \cdot \mathrm{e}(w, W)^{-s_\alpha - s_\beta} \cdot \mathrm{e}(w, g_2)^{-s_{\delta_1} - s_{\delta_2}} \cdot \mathrm{e}(h, g_2)^{-s_y} \cdot \mathrm{e}(T_3, W)^c \cdot \mathrm{e}(g_1, g_2)^{-c}$, $R_4 \stackrel{?}{=} T_1^{s_x} \cdot u^{-s_{\delta_1}}$, $R_5 \stackrel{?}{=} T_2^{s_x} \cdot v^{-s_{\delta_2}}$, $R_6 \stackrel{?}{=} d^{s_\mu} \cdot T_4^{-c}$, $R_7 \stackrel{?}{=} f^{s_\mu + s_y} \cdot T_5^{-c}$ and accepts if all hold.

Lemma 5.1:
The Protocol 5.1 is complete (the verifier always accepts an interaction with an honest prover).

Proof. If the prover $\mathcal{P}$ is honest and in possession of a triple (A, x, y) such that $A^{x+\gamma} = g_1 \cdot h^y$, he follows the computations specified for him in the protocol. In this case the following equations hold:

$$u^{s_\alpha} \cdot T_1^{-c} = u^{r_\alpha + c \cdot \alpha} \cdot (u^\alpha)^{-c} = u^{r_\alpha} = R_1$$

$$v^{s_\beta} \cdot T_2^{-c} = v^{r_\beta + c \cdot \beta} \cdot (v^\beta)^{-c} = v^{r_\beta} = R_2$$

$$\begin{aligned}
&\frac{\mathrm{e}(T_3,g_2)^{s_x} \cdot \mathrm{e}(w,W)^{-s_\alpha - s_\beta} \cdot \mathrm{e}(w,g_2)^{-s_{\delta_1} - s_{\delta_2}} \cdot \mathrm{e}(h,g_2)^{-s_y}}{\mathrm{e}(T_3,W)^{-c} \cdot \mathrm{e}(g_1,g_2)^c} \\
&\quad = \frac{\mathrm{e}(T_3,g_2)^{r_x + c \cdot x} \cdot \mathrm{e}(w,W)^{-r_\alpha - c \cdot \alpha - r_\beta - c \cdot \beta} \cdot \mathrm{e}(w,g_2)^{-r_{\delta_1} - c \cdot \delta_1 - r_{\delta_2} - c \cdot \delta_2} \cdot \mathrm{e}(h,g_2)^{-r_y - c \cdot y}}{\mathrm{e}(T_3,W)^{-c} \cdot \mathrm{e}(g_1,g_2)^c} \\
&\quad = R_3 \cdot \left[\frac{\mathrm{e}(T_3,g_2)^x \cdot \mathrm{e}(w,W)^{-\alpha - \beta} \cdot \mathrm{e}(w,g_2)^{-\delta_1 - \delta_2} \cdot \mathrm{e}(h,g_2)^{-y}}{\mathrm{e}(T_3,W)^{-1} \cdot \mathrm{e}(g_1,g_2)} \right]^c \\
&\quad = R_3 \cdot \left[\frac{\mathrm{e}(A \cdot w^{\alpha+\beta},g_2)^x \cdot \mathrm{e}(w,W)^{-\alpha - \beta} \cdot \mathrm{e}(w,g_2)^{x \cdot (-\alpha - \beta)} \cdot \mathrm{e}(h,g_2)^{-y}}{\mathrm{e}(A \cdot w^{\alpha+\beta},W)^{-1} \cdot \mathrm{e}(g_1,g_2)} \right]^c \\
&\quad = R_3 \cdot \left[\frac{\mathrm{e}(A,g_2)^x \cdot \mathrm{e}(w,W)^{-\alpha - \beta} \cdot \mathrm{e}(h,g_2)^{-y} \cdot \mathrm{e}(A \cdot w^{\alpha+\beta},W)}{\mathrm{e}(g_1,g_2)} \right]^c \\
&\quad = R_3 \cdot \left[\frac{\mathrm{e}(A,g_2)^x \cdot \mathrm{e}(h,g_2)^{-y} \cdot \mathrm{e}(A,W)}{\mathrm{e}(g_1,g_2)} \right]^c = R_3 \cdot \left[\frac{\mathrm{e}(A^{x+\gamma} \cdot h^{-y},g_2)}{\mathrm{e}(g_1,g_2)} \right]^c = R_3 \cdot \left[\frac{\mathrm{e}(g_1,g_2)}{\mathrm{e}(g_1,g_2)} \right]^c = R_3
\end{aligned}$$

$$T_1^{s_x} \cdot u^{-s_{\delta_1}} = (u^\alpha)^{r_x + c \cdot x} \cdot u^{-r_{\delta_1} - c \cdot \delta_1} = (u^\alpha)^{r_x + c \cdot x} \cdot u^{-r_{\delta_1} - c \cdot x \cdot \alpha} = (u^\alpha)^{r_x} \cdot u^{-r_{\delta_1}} = R_4$$

$$T_2^{s_x} \cdot v^{-s_{\delta_2}} = (v^\beta)^{r_x + c \cdot x} \cdot v^{-r_{\delta_2} - c \cdot \delta_2} = (v^\beta)^{r_x + c \cdot x} \cdot v^{-r_{\delta_2} - c \cdot x \cdot \beta} = (v^\beta)^{r_x} \cdot v^{-r_{\delta_2}} = R_5$$

$$d^{s_\mu} \cdot T_4^{-c} = d^{r_\mu + c \cdot \mu} \cdot (d^\mu)^{-c} = d^{r_\mu} = R_6$$

$$f^{s_\mu + s_y} \cdot T_5^{-c} = f^{r_\mu + c \cdot \mu + r_y + c \cdot y} \cdot (f^{\mu + y})^{-c} = f^{r_\mu + r_y} = R_7$$

So the verifier will always accept when the prover is honest. □

Lemma 5.2:
Assuming the Decision Linear Problem holds in $\mathbb{G}_1$, transcripts of Protocol 5.1 can be simulated.

Proof. We describe a simulator that outputs transcripts for Protocol 5.1 that are indistinguishable from real protocol transcripts.
In the first step the simulator chooses $\hat{A} \stackrel{\$}{\leftarrow} \mathbb{G}_1$ and $\hat{y}, \alpha, \beta, \mu \stackrel{\$}{\leftarrow} \mathbb{Z}_p$. Then the values $T_1 := u^\alpha$, $T_2 := v^\beta$, $T_3 := \hat{A} \cdot w^{\alpha+\beta}$, $T_4 := d^\mu$, and $T_5 := f^{\mu+\hat{y}}$ are computed. In the second step

the simulator chooses the value $c \xleftarrow{\$} \mathbb{Z}_p$ as a simulated challenge and the values s_α, s_β, s_x, $s_{\hat{y}}$, s_μ, s_{δ_1}, $s_{\delta_2} \xleftarrow{\$} \mathbb{Z}_p$. Then the simulator computes $R_1 := u^{s_\alpha} \cdot T_1^{-c}$, $R_2 := v^{s_\beta} \cdot T_2^{-c}$, $R_3 := \mathrm{e}(T_3, g_2)^{s_x} \cdot \mathrm{e}(w, W)^{-s_\alpha - s_\beta} \cdot \mathrm{e}(w, g_2)^{-s_{\delta_1} - s_{\delta_2}} \cdot \mathrm{e}(h, g_2)^{-s_{\hat{y}}} \mathrm{e}(T_3, W)^c \cdot \mathrm{e}(g_1, g_2)^{-c}$, $R_4 := T_1^{s_x} \cdot u^{-s_{\delta_1}}$, $R_5 := T_2^{s_x} \cdot v^{-s_{\delta_2}}$, $R_6 := d^{s_\mu} \cdot T_4^{-c}$, and $R_7 := f^{s_\mu + s_{\hat{y}}} \cdot T_5^{-c}$ using the verification equations from Protocol 5.1 and outputs (T_1, T_2, T_3, T_4, T_5, R_1, R_2, R_3, R_4, R_5, R_6, R_7, c, s_α, s_β, s_x, $s_{\hat{y}}$, s_μ, s_{δ_1}, s_{δ_2}) as the transcript.
Now we show that the transcripts created by the simulator are indistinguishable from transcripts of the real protocol, assuming D-Linear1 holds in $\mathbb{G}_1$.
Together with the public values u, v, w, h, d, f, $C = h^y$ the 6-tuples (u, v, w, T_1, T_2, T_3) and (h, d, f, C, T_4, T_5) are completely random instances of the Decision Linear Problem in $\mathbb{G}_1$. Hence, the distribution of this tuples in the simulation can not be distinguished from the distribution in the real protocol.
By choosing the values $s_\alpha, s_\beta, s_x, s_{\hat{y}}, s_\mu, s_{\delta_1}, s_{\delta_2}$ and c uniformly at random, the values R_1, R_2, R_3, R_4, R_5, R_6, R_7 are fixed, such that the verification equations are satisfied. Therefore the tuple (R_1, R_2, R_3, R_4, R_5, R_6, R_7, c, s_α, s_β, $s_x, s_{\hat{y}}, s_\mu, s_{\delta_1}, s_{\delta_1}$) is distributed as in the real protocol. Using a standard hybrid argument, it follows that transcripts generated by the simulator are indistinguishable from transcripts of the real protocol, assuming the Decision Linear Problem holds in $\mathbb{G}_1$. □

Lemma 5.3:
The Protocol 5.1 is a proof of knowledge (there exists an extractor for this protocol).

Proof. Suppose an algorithm $\mathcal{E}$ (the extractor) that is given two transcripts of protocol 5.1 (T_1, T_2, T_3, T_4, T_5, R_1, R_2, R_3, R_4, R_5, R_6, R_7, c, s_α, s_β, s_x, s_y, s_μ, s_{δ_1}, s_{δ_2}) and (T_1, T_2, T_3, T_4, T_5, R_1, R_2, R_3, R_4, R_5, R_6, R_7, c', s'_α, s'_β, s'_x, s'_y, s'_μ, s'_{δ_1}, s'_{δ_2}) where $c \neq c'$. Then a valid extended q-SDH triple can be computed by $\mathcal{E}$ as follows: $\Delta c := c - c'$, $\Delta s_\alpha := s_\alpha - s'_\alpha$, $\Delta s_\beta := s_\beta - s'_\beta$, $\Delta s_x := s_x - s'_x$, $\Delta s_y := s_y - s'_y$, $\Delta s_\mu := s_\mu - s'_\mu$, $\Delta s_{\delta_1} := s_{\delta_1} - s'_{\delta_1}$, and $\Delta s_{\delta_2} := s_{\delta_2} - s'_{\delta_2}$. Dividing the two instances for each of the verification equations gives

$$1_{\mathbb{G}_1} = u^{s_\alpha} \cdot T_1^{-c} \cdot u^{-s'_\alpha} \cdot T_1^{c'} = u^{\Delta s_\alpha} \cdot T_1^{-\Delta c} \Longrightarrow T_1^{\Delta c} = u^{\Delta s_\alpha} \Longrightarrow T_1 = u^{\hat{\alpha}} \text{ where } \hat{\alpha} = \frac{\Delta s_\alpha}{\Delta c}$$

$$1_{\mathbb{G}_1} = v^{s_\beta} \cdot T_2^{-c} \cdot v^{-s'_\beta} \cdot T_2^{c'} = v^{\Delta s_\beta} \cdot T_2^{-\Delta c} \Longrightarrow T_2^{\Delta c} = v^{\Delta s_\beta} \Longrightarrow T_2 = v^{\hat{\beta}} \text{ where } \hat{\beta} = \frac{\Delta s_\beta}{\Delta c}$$

$$1_{\mathbb{G}_1} = d^{s_\mu} \cdot T_4^{-c} \cdot d^{-s'_\mu} \cdot T_4^{c'} = d^{\Delta s_\mu} \cdot T_4^{-\Delta c} \Longrightarrow T_4^{\Delta c} = d^{\Delta s_\mu} \Longrightarrow T_4 = d^{\hat{\mu}} \text{ where } \hat{\mu} = \frac{\Delta s_\mu}{\Delta c}$$

$$1_{\mathbb{G}_1} = f^{s_\mu + s_y} \cdot T_5^{-c} \cdot f^{-s'_\mu - s'_y} \cdot T_5^{c'} = f^{\Delta s_\mu + \Delta s_y} \cdot T_5^{-\Delta c} \Longrightarrow T_5^{\Delta c} = f^{\Delta s_\mu + \Delta s_y}$$
$$\Longrightarrow T_5 = f^{\hat{\mu} + \hat{y}} \text{ where } \hat{y} = \frac{\Delta s_y}{\Delta c}$$

$$1_{\mathbb{G}_1} = T_1^{s_x} \cdot u^{-s_{\delta_1}} \cdot T_1^{-s'_x} \cdot u^{s'_{\delta_1}} = T_1^{\Delta s_x} \cdot u^{-\Delta s_{\delta_1}} \Longrightarrow T_1^{\Delta s_x} = u^{\Delta s_{\delta_1}} \Longrightarrow \hat{\alpha} \cdot \Delta s_x = \Delta s_{\delta_1}$$

$$1_{\mathbb{G}_1} = T_2^{s_x} \cdot v^{-s_{\delta_2}} \cdot T_2^{-s'_x} \cdot v^{s'_{\delta_2}} = T_2^{\Delta s_x} \cdot v^{-\Delta s_{\delta_2}} \Longrightarrow T_2^{\Delta s_x} = v^{\Delta s_{\delta_2}} \Longrightarrow \hat{\beta} \cdot \Delta s_x = \Delta s_{\delta_2}$$

$$1_{\mathbb{G}_T} = \frac{\mathrm{e}(T_3, g_2)^{s_x} \cdot \mathrm{e}(w, W)^{-s_\alpha - s_\beta} \cdot \mathrm{e}(w, g_2)^{-s_{\delta_1} - s_{\delta_2}} \cdot \mathrm{e}(h, g_2)^{-s_y}}{\mathrm{e}(T_3, g_2)^{s'_x} \cdot \mathrm{e}(w, W)^{-s'_\alpha - s'_\beta} \cdot \mathrm{e}(w, g_2)^{-s'_{\delta_1} - s'_{\delta_2}} \cdot \mathrm{e}(h, g_2)^{-s'_y}} \cdot \frac{\mathrm{e}(T_3, W)^{-c'} \cdot \mathrm{e}(g_1, g_2)^{c'}}{\mathrm{e}(T_3, W)^{-c} \cdot \mathrm{e}(g_1, g_2)^{c}}$$
$$= \mathrm{e}(T_3, g_2)^{\Delta s_x} \cdot \mathrm{e}(w, W)^{-\Delta s_\alpha - \Delta s_\beta} \cdot \mathrm{e}(w, g_2)^{-\Delta s_{\delta_1} - \Delta s_{\delta_2}} \cdot \mathrm{e}(h, g_2)^{-\Delta s_y} \cdot \frac{\mathrm{e}(T_3, W)^{\Delta c}}{\mathrm{e}(g_1, g_2)^{\Delta c}}$$

$\Longleftrightarrow$

$$\left(\frac{\mathrm{e}(g_1,g_2)}{\mathrm{e}(T_3,W)}\right)^{\Delta c} = \mathrm{e}(T_3,g_2)^{\Delta s_x}\cdot \mathrm{e}(w,W)^{-\Delta s_\alpha-\Delta s_\beta}\cdot \mathrm{e}(w,g_2)^{-\Delta s_{\delta_1}-\Delta s_{\delta_2}}\cdot \mathrm{e}(h,g_2)^{-\Delta s_y}$$

taking Δc-th root and letting $\hat{x} = \frac{\Delta s_x}{\Delta c}$ we obtain

$$\frac{\mathrm{e}(g_1,g_2)}{\mathrm{e}(T_3,W)} = \mathrm{e}(T_3,g_2)^{\hat{x}}\cdot \mathrm{e}(w,W)^{-\hat{\alpha}-\hat{\beta}}\cdot \mathrm{e}(w,g_2)^{-\hat{\alpha}\cdot\hat{x}-\hat{\beta}\cdot\hat{x}}\cdot \mathrm{e}(h,g_2)^{-\hat{y}}$$

this can be rearranged as

$$\begin{aligned}\mathrm{e}(g_1,g_2) &= \mathrm{e}(T_3,g_2)^{\hat{x}}\cdot \mathrm{e}(w^{-\hat{\alpha}-\hat{\beta}},W)\cdot \mathrm{e}(w^{-\hat{\alpha}-\hat{\beta}},g_2)^{\hat{x}}\cdot \mathrm{e}(h,g_2)^{-\hat{y}}\cdot \mathrm{e}(T_3,W)\\ &= \mathrm{e}(T_3\cdot w^{-\hat{\alpha}-\hat{\beta}},g_2)^{\hat{x}}\cdot \mathrm{e}(h,g_2)^{-\hat{y}}\cdot \mathrm{e}(T_3\cdot w^{-\hat{\alpha}-\hat{\beta}},W)\end{aligned}$$

letting $\hat{A} = T_3\cdot w^{-\hat{\alpha}-\hat{\beta}}$ it holds

$$\mathrm{e}(g_1,g_2) = \mathrm{e}(\hat{A},g_2)^{\hat{x}}\cdot \mathrm{e}(h,g_2)^{-\hat{y}}\cdot \mathrm{e}(\hat{A},W).$$

Hence, the extractor obtains a tuple $(\hat{A},\hat{x},\hat{y})$, which is a valid extended q-SDH triple. Moreover, the $\hat{A}$ is exactly the same as that in the Linear Encryption (T_1,T_2,T_3), the $\hat{y}$ is the same as in T_5 and so also $\hat{x}$ is the same as in the transcripts. So the extractor gets exactly the same values the prover knows. □

6 Our Construction

In this section we define our reputation system based on extended q-SDH and D-Linear1. To give some intuition for this system we provide an honest-verifier zero-knowledge proof of knowledge in Section 5.

6.1 The Reputation System

We apply the Fiat-Shamir heuristic [14, 1] on Protocol 5.1 to obtain a signature of knowledge which is secure in the random oracle model. By extending this signature scheme we construct a reputation system. We use the challenge c as a part of the signature rather than the values $R_1,\ldots,R_7$, modelling the value c as the output of a random oracle. This technique is widely used in the context of group signatures [6, 8, 3].

In a reputation system, the key issuer publishes *item*s for which the signatures are created. Every user can create a single signature for every *item* without losing anonymity. Due to the public linkability two signatures for one *item* by the same user can be detected. In such a case, the anonymity of the cheating user is revoked by the group manager. By publishing a *revocation token* the group manager can declare signatures from the cheating user as *invalid*. This invalidity can be checked by every verifier using verifier-local revocation [8, 26].

We assume the communication between users and the group manager and between users and the key issuer to take place via secure channels. Furthermore, the user's public key $upk[i]$ is certified by the group manager, such that the key issuer can verify the integrity of the public keys during the Join-Issue protocol.

In the following definitions we consider bilinear groups $\mathbb{G}_1$ and $\mathbb{G}_2$, and two hash functions modeled as random oracles: $H\colon \{0,1\}^* \longrightarrow \mathbb{Z}_p$ and $H_1\colon \{0,1\}^* \longrightarrow \mathbb{G}_2$. Furthermore, as in [6], we use Linear Encryption - a CPA-secure Elgamal-like encryption scheme based on the Decision Linear Problem (Definition 4.1).

$\text{KeyGen}_{GM}()$:

The group manager's key generation algorithm proceeds as follows:

1. Select $w \xleftarrow{\$} \mathbb{G}_1$, $\xi_1, \xi_2 \xleftarrow{\$} \mathbb{Z}_p$ and compute $u := w^{\frac{1}{\xi_1}}, v := w^{\frac{1}{\xi_2}}$. The values (u, v, w) are the public key of the Linear Encryption, the values (ξ_1, ξ_2) are the corresponding secret key.
2. Select $\hat{d} \xleftarrow{\$} \mathbb{G}_2$, $\zeta \xleftarrow{\$} \mathbb{Z}_p$ and compute $d := \psi(\hat{d}), h := d^{\zeta}$ as the basis for public linkability and revocation.
3. Set $gmpk := (u, v, w, h, d, \hat{d})$ and $gmsk := (\xi_1, \xi_2, \zeta)$ as the group manager's public and secret keys.

$\text{KeyGen}_{KI}(item)$:

The key issuer's key generation algorithm proceeds as follows:

1. Select $g_{2_{item}} \xleftarrow{\$} \mathbb{G}_2$ and set $g_{1_{item}} := \psi(g_{2_{item}})$.
2. Select $\gamma_{item} \xleftarrow{\$} \mathbb{Z}_p$ and set $W_{item} := g_{2_{item}}^{\gamma_{item}}$.
3. Add the *item*-based public key $ipk[item] := (g_{1_{item}}, g_{2_{item}}, W_{item})$ to the *ItemList* and keep $isk[item] := \gamma_{item}$ secret as the *item*-based secret key.

$\text{KeyGen}_{U}(i)$:

The user's key generation algorithm proceeds as follows:

1. Select $y_i \xleftarrow{\$} \mathbb{Z}_p$, set $upk[i] := h^{y_i}$ and $usk[i] := y_i$ as the user's public and secret keys.

$\text{Register}_{GM}(\text{St}_{GM}, M_{GM}), \text{Register}_{U}(\text{St}_{U}, M_{U})$:

The interactive registration protocol proceeds as follows:

1. The user sends his identity i to the group manager.
2. The group manager checks if there already exists an entry $reg[i]$ in the registration table. If so, he declares failure and exits. Otherwise, the group manager runs KeyGen_{U} to obtain the tuple $(upk[i], usk[i])$, sets $reg[i] := (i, upk[i])$ and sends $(upk[i], usk[i])$ and a certificate for $upk[i]$ to the user i.

$\text{Join}(\text{St}_{U}, M_{U}), \text{Issue}(\text{St}_{KI}, M_{KI})$:

The interactive key issuing protocol proceeds as follows:

1. The user looks up the public key corresponding to the used *item* $ipk[item] = (g_{1_{item}}, g_{2_{item}}, W_{item})$ in the *ItemList* and sends $(i, upk[i])$ and the certificate for $upk[i]$ to the key issuer.
2. The key issuer verifies the certificate for $upk[i]$ and checks that user i is not in possession of a signing key for the given *item*, i.e. there exists no entry $(upk[i], \cdot)$ in IL_{item}. If the certificate is invalid or there already is a signing key in the list, then he declares failure and exits. Otherwise, he selects $x_{i_{item}} \xleftarrow{\$} \mathbb{Z}_p$, computes $A_{i_{item}} := (g_{1_{item}} \cdot upk[i])^{\frac{1}{x_{i_{item}} + \gamma_{item}}}$, gives $gsk[i, item] := (A_{i_{item}}, x_{i_{item}})$ to the user i as his signing key for the specified *item*, and saves $(upk[i], gsk[i, item])$ in the identification list IL_{item} for this *item*.

Revoke$(gmpk, gmsk, i)$:

The revocation algorithm proceeds as follows:

1. Look up $upk[i]$ in $reg[i]$.
2. Using the group manager's secret key $gmsk$ compute $D_i \coloneqq upk[i]^{\frac{1}{\zeta}} = (h^{y_i})^{\frac{1}{\zeta}} = d^{y_i}$ and add the revocation token $grt[i] \coloneqq D_i$ to the revocation list $\mathcal{RL}$.

Sign$(item, gmpk, ipk[item], gsk[i, item], usk[i], M)$:

The group signing algorithm proceeds as follows:

1. Obtain the value $\hat{f} \in \mathbb{G}_2$ by $\hat{f} \coloneqq H_1(item)$.
2. Choose $\alpha, \beta, \mu \xleftarrow{\$} \mathbb{Z}_p$, compute

$$T_1 \coloneqq u^{\alpha} \qquad T_2 \coloneqq v^{\beta} \qquad T_3 \coloneqq A_{i_{item}} \cdot w^{\alpha+\beta} \qquad T_4 \coloneqq d^{\mu} \qquad T_5 \coloneqq \psi(\hat{f})^{\mu+y_i}$$

   and the helper values $\delta_1 \coloneqq \alpha \cdot x_{i_{item}}$ and $\delta_2 \coloneqq \beta \cdot x_{i_{item}}$.

3. Choose $r_\alpha, r_\beta, r_x, r_y, r_\mu, r_{\delta_1}, r_{\delta_2} \xleftarrow{\$} \mathbb{Z}_p$ and compute

$$R_1 \coloneqq u^{r_\alpha} \qquad\qquad R_2 \coloneqq v^{r_\beta}$$

$$R_3 \coloneqq \mathrm{e}(T_3, g_{2_{item}})^{r_x} \cdot \mathrm{e}(w, W_{item})^{-r_\alpha - r_\beta} \cdot \mathrm{e}(w, g_{2_{item}})^{-r_{\delta_1} - r_{\delta_2}} \cdot \mathrm{e}(h, g_{2_{item}})^{-r_y}$$

$$R_4 \coloneqq T_1^{r_x} \cdot u^{-r_{\delta_1}} \qquad R_5 \coloneqq T_2^{r_x} \cdot v^{-r_{\delta_2}} \qquad R_6 \coloneqq d^{r_\mu} \qquad R_7 \coloneqq \psi(\hat{f})^{r_\mu + r_y}.$$

4. Compute a challenge value c using H:

$$c \coloneqq H(M, item, T_1, T_2, T_3, T_4, T_5, R_1, R_2, R_3, R_4, R_5, R_6, R_7).$$

5. Compute

$$s_\alpha \coloneqq r_\alpha + c \cdot \alpha \qquad s_\beta \coloneqq r_\beta + c \cdot \beta \qquad s_x \coloneqq r_x + c \cdot x_{i_{item}} \qquad s_y \coloneqq r_y + c \cdot y_i$$
$$s_\mu \coloneqq r_\mu + c \cdot \mu \qquad s_{\delta_1} \coloneqq r_{\delta_1} + c \cdot \delta_1 \qquad s_{\delta_2} \coloneqq r_{\delta_2} + c \cdot \delta_2.$$

6. Output the signature $\sigma \coloneqq (item, T_1, T_2, T_3, T_4, T_5, c, s_\alpha, s_\beta, s_x, s_y, s_\mu, s_{\delta_1}, s_{\delta_2})$.

Verify$(item, gmpk, ipk[item], \mathcal{RL}, M, \sigma)$:

The signature verification algorithm proceeds as follows:

1. Obtain the value $\hat{f} \in \mathbb{G}_2$ by $\hat{f} \coloneqq H_1(item)$.
2. The verifier computes

$$R_1 \coloneqq u^{s_\alpha} \cdot T_1^{-c} \qquad\qquad R_2 \coloneqq v^{s_\beta} \cdot T_2^{-c}$$

$$R_3 \coloneqq \frac{\mathrm{e}(T_3, g_{2_{item}})^{s_x} \cdot \mathrm{e}(w, W_{item})^{-s_\alpha - s_\beta} \cdot \mathrm{e}(w, g_{2_{item}})^{-s_{\delta_1} - s_{\delta_2}} \cdot \mathrm{e}(h, g_{2_{item}})^{-s_y}}{\mathrm{e}(T_3, W_{item})^{c} \cdot \mathrm{e}(g_1, g_{2_{item}})^{-c}}$$

$$R_4 \coloneqq T_1^{s_x} \cdot u^{-s_{\delta_1}} \quad R_5 \coloneqq T_2^{s_x} \cdot v^{-s_{\delta_2}} \quad R_6 \coloneqq d^{s_\mu} \cdot T_4^{-c} \quad R_7 \coloneqq \psi(\hat{f})^{s_\mu + s_y} \cdot T_5^{-c}.$$

3. Check that the challenge c is correct:

$$c \stackrel{?}{=} H(M, item, T_1, T_2, T_3, T_4, T_5, R_1, R_2, R_3, R_4, R_5, R_6, R_7).$$

If this holds, then accept, otherwise reject.

4. For each element $D \in \mathcal{RL}$ check whether D is encoded in (T_4, T_5): $\mathrm{e}\left(T_5, \hat{d}\right) \stackrel{?}{=} \mathrm{e}(D \cdot T_4, \hat{f})$. If this is false for all $D \in \mathcal{RL}$, then the signer of σ has not been revoked and Verify accepts, otherwise rejects.
5. If both checks accept, then output 1, otherwise 0.

Link$(item, gmpk, ipk[item], (M', \sigma'), (M'', \sigma''))$:

The public linking algorithm proceeds as follows:

1. At first, check that σ' is a valid signature for message M' and that σ'' is a valid signature for message M''. If not, output 0.
2. Obtain the value $\hat{f} \in \mathbb{G}_2$ by $\hat{f} := H_1(item)$.
3. Output 1, if $\mathrm{e}\left(\frac{T_5'}{T_5''}, \hat{d}\right) \stackrel{?}{=} \mathrm{e}\left(\frac{T_4'}{T_4''}, \hat{f}\right)$ holds and 0 otherwise.

Open$(gmpk, gmsk, M, \sigma)$:

The opening algorithm proceeds as follows:

1. Check that σ is a valid signature of knowledge for message M. If not, output `failure`.
2. Compute $A_{i_{item}} := T_3 \cdot T_1^{-\xi_1} \cdot T_2^{-\xi_2}$ using the group manager's secret key.
3. The group manager looks up the user index i from the identification list IL_{item}.
4. If no entry for $A_{i_{item}}$ can be found in IL_{item} return `failure`, otherwise return i.

Remarks: We assume the communication between users and the group manager and between users and the key issuer to take place via secure channels. Furthermore, the user's public key $upk[i]$ is certified by the group manager, such that the key issuer can verify the integrity of the public keys during the join-issue protocol. Since we assume the group manager to be honest, we can let him choose the user's public and secret keys. We need the honest group manager to prove security for the system.

Correctness: The correctness of the reputation system can be shown as follows:

- Protocol 5.1 is correct, i.e. every honestly created signature will be declared as valid.
- Revocation token are computed correctly.
- For honestly created signatures the group manager can always recover the identity of the signer, because of the correctness of the Linear Encryption.
- Two signatures for the same *item* by the same user are declared as publicly linked.
- Every secret signing key $gsk[i]$ created by the key issuer can be publicly verified by

$$\mathrm{e}(A_{i_{item}}, g_2)^{x_{i_{item}}} \cdot \mathrm{e}(A_{i_{item}}, W) \cdot \mathrm{e}(h, g_2)^{-y_i} \stackrel{?}{=} \mathrm{e}(g_1, g_2).$$

6.2 Security of the Reputation System

As mentioned in Section 3 the anonymity experiment as defined in Figure 4 can be relaxed to CPA-anonymity. We will prove security in this slightly weaker model, analogously to [6]. In the following lemmata $\mathcal{Q}$ is the overall number of oracle queries made by the adversary. The proofs of security are all in the random oracle model.

Lemma 6.1:
If D-Linear1 (t', ε')-holds in $\mathbb{G}_2$, then the reputation system defined in Section 6 is $(t,\ \mathcal{Q},\ q_{H_1},\ q_{AU},\ \varepsilon)$-CPA-anonymous, where $t = t' - \mathcal{Q} \cdot \mathcal{O}(1)$ and $\varepsilon = \varepsilon' \cdot 2 \cdot q_{AU} \cdot q_{H_1}$. Here q_{H_1} is the number of hash oracle queries to H_1 and q_{AU} is the number of AddU oracle queries.

Proof. Suppose $\mathcal{A}$ is an adversary that (t, ε)-breaks the anonymity of the reputation system. Then we can construct an adversary $\mathcal{B}$ that decides D-Linear1 in $\mathbb{G}_2$ with advantage at least $\frac{\varepsilon}{2 \cdot q_{AU} \cdot q_{H_1}}$ in time $t + \mathcal{Q} \cdot \mathcal{O}(1)$.
Algorithm $\mathcal{B}$ is given an instance of the Decision Linear Problem $(\hat{u}, \hat{v}, \hat{w}, \hat{u}^{\hat{a}}, \hat{v}^{\hat{b}}, \hat{w}^{\hat{c}}) \in \mathbb{G}_2^6$ and has to decide whether $\hat{c} = \hat{a} + \hat{b}$ holds. $\mathcal{B}$ does so by interacting with algorithm $\mathcal{A}$.
At first, $\mathcal{B}$ sets $\mathcal{HU} \coloneqq \emptyset$, $\mathcal{CU} \coloneqq \emptyset$, $\mathcal{RL} \coloneqq \emptyset$, $\mathit{ItemList} \coloneqq \emptyset$, $\mathit{reg} \coloneqq \emptyset$, $\mathcal{RU} \coloneqq \emptyset$, $\mathcal{JIU} \coloneqq \emptyset$, $\mathcal{GS} \coloneqq \emptyset$ and computes a second, independent instance of D-Linear1 $(\hat{h}, \hat{d}, \hat{f}, \hat{h}^{y^*}, \hat{d}^{\mu^*}, \hat{f}^{r^*})$ using the standard random self-reducibility technique (see Lemma 4.1).
Although $\mathcal{A}$ has to output two identities in his anonymity challenge, it suffices for $\mathcal{B}$ to guess one identity, namely the one that is used in the challenge signature. Identities are created using the AddU oracle, and there are at most q_{AU} queries to this oracle. Hence, $\mathcal{B}$ chooses $\ell_1 \xleftarrow{\$} \{1, \ldots, q_{AU}\}$ as his guess which identity can be used for $\mathcal{A}$'s challenge. Analogously, $\mathcal{B}$ has to guess for which *item* $\mathcal{A}$ wants to be challenged. For every *item* the hash value $H_1(\mathit{item})$ is needed to create a signature, and there are at most q_{H_1} queries to H_1. Hence, $\mathcal{B}$ chooses $\ell_2 \xleftarrow{\$} \{1, \ldots, q_{H_1}\}$ as his guess that the ℓ_2'th query to H_1 is for the *item* that $\mathcal{A}$ wants to be challenged on.
To compute the group manager's public key $\mathcal{B}$ sets $\mathit{gmpk} \coloneqq (u \coloneqq \psi(\hat{u}),\ v \coloneqq \psi(\hat{v}),\ w \coloneqq \psi(\hat{w}),\ h \coloneqq \psi(\hat{h}),\ d \coloneqq \psi(\hat{d}),\ \hat{d})$ and gives gmpk to $\mathcal{A}$. Then $\mathcal{A}$ starts to interact with $\mathcal{B}$ via the oracles. Algorithm $\mathcal{B}$ responds to oracle queries by running exactly the defined oracles from Figure 3, except to queries to AddU, USK, RevU and GSig. These oracles are realized as follows:
$\boldsymbol{H(\cdot)}$: For hash oracle queries to H $\mathcal{B}$ chooses $c \xleftarrow{\$} \mathbb{Z}_p$, gives c to $\mathcal{A}$ and ensures to respond identically to repeated queries.
$\boldsymbol{H_1(\mathit{item})}$: To the ℓ_2'th query to H_1 $\mathcal{B}$ responds by patching the oracle at *item* to match $\hat{f}$. If this causes a collision, $\mathcal{B}$ declares failure and exits. Collisions happen with negligible probability, hence here and in the remainder of the proof we ignore these probabilities. For queries not equal to ℓ_2, $\mathcal{B}$ chooses $f' \xleftarrow{\$} \mathbb{G}_2$ and gives f' to $\mathcal{A}$. $\mathcal{B}$ ensures to respond identically to repeated queries.
AddU($\boldsymbol{i}$): To the ℓ_1'th query $\mathcal{B}$ responds by setting $i^* \coloneqq i, \mathit{upk}[i^*] \coloneqq \psi(\hat{h}^{y^*})$ and returning $\mathit{upk}[i^*]$. For every other query $\mathcal{B}$ follows the oracle definition of Figure 3.
USK($\boldsymbol{i}$): If $i \neq i^*$ then $\mathcal{B}$ responds as defined in Figure 3. If $i = i^*$ then $\mathcal{B}$ cannot respond as $\mathit{usk}[i^*] = y^*$ is not known. Hence, $\mathcal{B}$ declares failure and exits.
RevU($\boldsymbol{i}$): If $i \neq i^*$ then $\mathcal{B}$ responds as defined in Figure 3. If $i = i^*$ then $\mathcal{B}$ cannot respond as

$usk[i^*] = y^*$ is not known and d^{y^*} can not be computed. Hence, $\mathcal{B}$ declares failure and exits.
GSig(i, *item*, M): Algorithm $\mathcal{B}$ has to handle three different cases:

- If $i \neq i^*$ then $\mathcal{B}$ responds as defined in Figure 3.
- If $i = i^* \wedge H_1(item) \neq \hat{f}$ then $\mathcal{B}$ simulates the signature using the simulator of Lemma 5.2. To do this, $\mathcal{B}$ checks that i is an honest user, that there exists a public key for the given *item* in the *ItemList*, and that user i owns a personal signing key $gsk[i, item] \in IL_{item}$. If one of this three checks does not succeed, $\mathcal{B}$ returns an empty string (as defined in the GSig oracle). Next $\mathcal{B}$ chooses $\alpha, \beta, \mu, y' \xleftarrow{\$} \mathbb{Z}_p$ and computes $T_1 := u^\alpha$, $T_2 := v^\beta$, $T_3 := A_{i^*_{item}} \cdot w^{\alpha+\beta}$, $T_4 := d^\mu$, $T_5 := \psi(H_1(item))^{\mu+y'}$. Afterwards, $\mathcal{B}$ chooses $c, s_\alpha, s_\beta, s_x, s_{y'}, s_\mu, s_{\delta_1}, s_{\delta_2} \xleftarrow{\$} \mathbb{Z}_p$ and computes $R_1, \ldots, R_7$ using the verification equations from the Verify algorithm. To ensure the signature is valid, $\mathcal{B}$ patches H at (M, *item*, T_1, T_2, T_3, T_4, T_5, R_1, R_2, R_3, R_4, R_5, R_6, R_7) to equal c. If this causes a collision, $\mathcal{B}$ declares failure and exits. Finally, $\mathcal{B}$ gives $\sigma := (item, T_1, T_2, T_3, T_4, T_5, c, s_\alpha, s_\beta, s_x, s_y, s_\mu, s_{\delta_1}, s_{\delta_2})$ to $\mathcal{A}$.
- If $i = i^* \wedge H_1(item) = \hat{f}$ then $\mathcal{B}$ declares failure and exits.

At some point $\mathcal{A}$ outputs a tuple $(i_0, i_1, item, M, \mathrm{St})$. If i_0 or i_1 are not honest users or the specified *item* does not exist in the *ItemList*, then $\mathcal{B}$ declares failure and exits. Also, $\mathcal{B}$ declares failure and exits, if $i_b \neq i^*$ or $H_1(item) \neq \hat{f}$. Otherwise, the challenge is computed as follows: $\mathcal{B}$ sets $T_1 := \psi(\hat{u}^{\hat{a}})$, $T_2 := \psi(\hat{v}^{\hat{b}})$, $T_3 := A_{i^*_{item}} \cdot \psi(\hat{w}^{\hat{c}})$, $T_4 := \psi(\hat{d}^{\mu^*})$, $T_5 := \psi(\hat{f}^{r^*})$ and simulates the values $(c, s_\alpha, s_\beta, s_x, s_y, s_\mu, s_{\delta_1}, s_{\delta_2})$ as described in the GSig oracle. If this causes a collision during the simulation of the hash value c, $\mathcal{B}$ declares failure and exits. Otherwise, the challenge signature is given to $\mathcal{A}$.
In the `guess`-phase $\mathcal{B}$ responds to $\mathcal{A}$'s queries to the oracles as before. When $\mathcal{A}$ outputs it's guess $b' \in \{0, 1\}$ $\mathcal{B}$ outputs 1, iff $i_{b'} = i^*$ as his guess for his D-Linear1 challenge.
All keys given to $\mathcal{A}$ and the responses to $\mathcal{A}$'s queries are properly distributed, except the completely simulated signatures, but these can not be distinguished from real signatures, assuming D-Linear1 holds.
Now we analyze the advantage of $\mathcal{B}$ in deciding D-Linear1 in $\mathbb{G}_2$. Suppose the D-Linear1 instance given to $\mathcal{B}$ is a real D-Linear1 tuple, i.e. $\hat{c} = \hat{a} + \hat{b}$. Then also $r^* = y^* + \mu^*$ and the challenge signature is a valid signature of user i_b. Hence, $\mathcal{A}$ has advantage ε in breaking the anonymity of the reputation system.
Suppose the D-Linear1 instance given to $\mathcal{B}$ is a random instance, i.e. $\hat{c} \xleftarrow{\$} \mathbb{Z}_p$. Then also $r^* \xleftarrow{\$} \mathbb{Z}_p$ and the challenge signature is completely independent of i_b. Hence, algorithm $\mathcal{A}$'s advantage is 0.
If $\mathcal{B}$ guesses the correct identity i^* and the correct *item* for the challenge signature, then $\mathcal{B}$ will not abort. Guessing the correct identity i^* happens with a probability of at least $\frac{1}{q_{AU}}$. Analogously, guessing the correct *item* happens with a probability of at least $\frac{1}{q_{H_1}}$. Hence, $\mathcal{B}$ outputs a guess for his D-Linear1 challenge with a probability of at least $\frac{1}{q_{AU} \cdot q_{H_1}}$ and the advantage in deciding D-Linear1 is at least $\varepsilon' = \frac{\varepsilon}{2 \cdot q_{AU} \cdot q_{H_1}}$. Algorithm $\mathcal{B}$ can compute a response to $\mathcal{A}$'s oracle queries in constant time. Because there are at most $\mathcal{Q}$ queries and $\mathcal{A}$ runs in time t, $\mathcal{B}$ runs in time $t' = t + \mathcal{Q} \cdot \mathcal{O}(1)$. □

Lemma 6.2:
If q-SDH (t', ε')-holds in $(\mathbb{G}_1, \mathbb{G}_2)$, then the reputation system defined in Section 6 is (t, ε)-publicly linkable, where $t = t' - \mathcal{Q} \cdot \mathcal{O}(1)$ and $\varepsilon = q_{AI} \cdot \sqrt{32 \cdot q_H \cdot (q-1) \cdot \varepsilon'} + \frac{q_{AI}}{p}$. Furthermore, the overall number of oracle queries to SndToGM must be at most $q-1$. Here q_H is the number of hash function queries to H, q_{AI} is the number of oracle queries to the AddItem oracle made by the adversary and p is the size of the groups $\mathbb{G}_1$ and $\mathbb{G}_2$.

Proof. Suppose $\mathcal{A}$ is an adversary that (t, ε)-breaks the public linkability of the reputation system. Then we can construct an adversary $\mathcal{B}$ that solves q-SDH in $(\mathbb{G}_1, \mathbb{G}_2)$ with advantage ε' of at least $\varepsilon' = \left(\frac{\varepsilon}{q_{AI}} - \frac{1}{p}\right)^2 \cdot \frac{1}{32 \cdot q_H \cdot (q-1)}$ in time $t' = t + \mathcal{Q} \cdot \mathcal{O}(1)$. Without loss of generality we assume that $\mathcal{A}$ creates exactly $q-1$ users via the SndToGM oracle.
Algorithm $\mathcal{B}$ is given an instance of the q-SDH problem $\left(G_1, G_2, G_2^{\hat{\gamma}}, \ldots, G_2^{(\hat{\gamma}^q)}\right)$ where $G_1 = \psi(G_2)$. $\mathcal{B}$ now chooses $\alpha, x_j, y_j \xleftarrow{\$} \mathbb{Z}_p$, for $j = 1, \ldots, q-1$, $k \xleftarrow{\$} \{1, \ldots, q-1\}$ and sets $\gamma := \hat{\gamma} - x_k$ which is unknown. Then the following values are computed by $\mathcal{B}$:

$$g_2 := G_2^{\alpha \cdot \prod_{i=1}^{q-1} (\hat{\gamma} - x_k + x_i) - y_k \cdot \prod_{i=1, i \neq k}^{q-1} (\hat{\gamma} - x_k + x_i)} \qquad g_1 := \psi(g_2) \qquad (1)$$

$$\hat{h} := G_2^{\prod_{i=1, i \neq k}^{q-1} (\hat{\gamma} - x_k + x_i)} \qquad h := \psi(\hat{h}) \qquad (2)$$

$$W := \left[G_2^{\alpha \cdot \prod_{i=1}^{q-1} (\hat{\gamma} - x_k + x_i) - y_k \cdot \prod_{i=1, i \neq k}^{q-1} (\hat{\gamma} - x_k + x_i)} \right]^{\hat{\gamma} - x_k} \qquad (3)$$

$$A_j := \psi \left[\left(G_2^{\alpha \cdot \prod_{i=1}^{q-1} (\hat{\gamma} - x_k + x_i) - y_k \cdot \prod_{i=1, i \neq k}^{q-1} (\hat{\gamma} - x_k + x_i)} \cdot G_2^{y_j \cdot \prod_{i=1, i \neq k}^{q-1} (\hat{\gamma} - x_k + x_i)} \right)^{\frac{1}{\gamma + x_j}} \right].$$

All products in the exponents are polynomials of $\hat{\gamma}$ of degree at most q. By expanding the products all the specified values can be computed using the given q-SDH instance. With $\gamma := \hat{\gamma} - x_k$, the value W equals g_2^{γ}, while γ is unknown to $\mathcal{B}$.
To generate the group manager's public key, $\mathcal{B}$ selects $w \xleftarrow{\$} \mathbb{G}_1, \zeta, \xi_1, \xi_2 \xleftarrow{\$} \mathbb{Z}_p$ and computes

$$u := w^{\frac{1}{\xi_1}} \qquad v := w^{\frac{1}{\xi_2}} \qquad \hat{d} := \hat{h}^{\frac{1}{\zeta}} \qquad d := \psi(\hat{d}).$$

Now $\mathcal{B}$ sets $\mathcal{HU} := \emptyset$, $\mathcal{CU} := \emptyset$, $\mathcal{RL} := \emptyset$, $\mathit{ItemList} := \emptyset$, $\mathit{reg} := \emptyset$, $\mathcal{RU} := \emptyset$, $\mathcal{JIU} := \emptyset$, $\mathcal{GS} := \emptyset$ and $\mathit{gmpk} := (u, v, w, h, d, \hat{d})$. Before the group manager's public key is given to $\mathcal{A}$, $\mathcal{B}$ has to guess an item^* as the item for which $\mathcal{A}$ will output the message-signature pairs as its solution to the public linkability experiment. To do so, $\mathcal{B}$ selects $\ell \xleftarrow{\$} \{1, \ldots, q_{AI}\}$ and uses the item of the ℓ'th query to the AddItem oracle as item^*.
Now $\mathcal{B}$ gives the group manager's public key gmpk to $\mathcal{A}$ and starts to interact with $\mathcal{A}$ via the oracles. $\mathcal{A}$'s queries are answered as follows:
$\boldsymbol{H}(\cdot)$: $\mathcal{B}$ chooses $c \xleftarrow{\$} \mathbb{Z}_p$, gives c to $\mathcal{A}$ and ensures to respond identically to repeated queries.
$\boldsymbol{H_1}(\boldsymbol{item})$: $\mathcal{B}$ chooses $\hat{f} \xleftarrow{\$} \mathbb{G}_2$, gives $\hat{f}$ to $\mathcal{A}$ and ensures to respond identically to repeated queries.

AddItem($item$): On the ℓ'th query $\mathcal{B}$ sets $item^* := item$ and $ipk[item^*] := (g_1, g_2, W)$, adds $ipk[item^*]$ to the *ItemList* and returns $ipk[item^*]$. For every other query $\mathcal{B}$ executes the KeyGen_{KI} algorithm and returns the resulting $item$-based public key $ipk[item]$.
SndToGM(i, M_{GM}): Algorithm $\mathcal{B}$ responds to the j'th query by running exactly the defined oracle from Figure 3 using y_j as the sercret key of user i, i.e. $usk[i] := y_j$ and $upk[i] := h^{y_j}$.
SndToKI(i, $item$, $upk[i]$, M_{KI}): If $item \neq item^*$ then $\mathcal{B}$ executes exactly the oracle defined in Figure 3. Otherwise, $\mathcal{B}$ runs the defined oracle using $gsk[i, item^*] := (A_j, x_j)$, where A_j and x_j correspond to user i's secret key $usk[i] = y_j$ defined in the SndToGM oracle for user i.
At some point $\mathcal{A}$ outputs an $item$ and exactly q message-signature pairs $((m_1, \sigma_1), \ldots, (m_q, \sigma_q))$. If $item \neq item^*$, at least one signature is invalid or there are at least two publicly linkable signatures then $\mathcal{B}$ declares failure and exits. Otherwise, $\mathcal{B}$ computes the revocation tokens $grt[i] = d^{y_i}$ for all $q-1$ group members, adds them to the revocation list $\mathcal{RL}$ and runs the verification algorithm for every message-signature pair $(m_i, \sigma_i), i = 1, \ldots, q$. Since the signatures are not publicly linkable, there must be at least one message-signature pair (m_{i^*}, σ_{i^*}) such that σ_{i^*} is still a valid signature. Now we apply the Forking Lemma [27] to obtain a second solution to the linkability experiment which can be used to solve q-SDH.
A run of $\mathcal{B}$ interacting with $\mathcal{A}$ is completely described by the randomness string ω used by $\mathcal{A}$ and $\mathcal{B}$, and by the vectors ρ_H, ρ_{H_1} of responses made by the random oracles H and H_1. We assume that the random oracle queries by $\mathcal{A}$ are distinct and we denote the i'th query to H by q_H^i and the i'th query to H_1 by $q_{H_1}^i$. The response to q_H^i is denoted by ρ_H^i, the response to $q_{H_1}^i$ is denoted by $\rho_{H_1}^i$. Hence, a random choice of ρ_H and ρ_{H_1} exactly corresponds to the random choices of H and H_1. From here on, we abbreviate signatures as $(m, \sigma^0, c, \sigma^1)$, where $\sigma^0 = (item, T_1, T_2, T_3, T_4, T_5, R_1, R_2, R_3, R_4, R_5, R_6, R_7)$, $\sigma^1 = (s_\alpha, s_\beta, s_x, s_y, s_\mu, s_{\delta_1}, s_{\delta_2})$ and c is the value derived by the random oracle H on input (m, σ^0). The values omitted in the signature can be computed according to the equations used in the Verify algorithm.
For a random choice of $(\omega, \rho_{H_1}, \rho_H)$ algorithm $\mathcal{A}$ outputs a solution to the public linkability experiment with advantage ε. With probability $\frac{1}{q_{AI}}$ this solution is for $item^*$. For the message-signature pair $(m_{i^*}, \sigma_{i^*}) = (m_{i^*}, \sigma_{i^*}^0, c, \sigma_{i^*}^1)$ the probability that c equals $H(m_{i^*}, \sigma_{i^*}^0)$ is $\frac{1}{p}$, unless it has been queried during the attack. Because this probability is negligable it is likely that $\mathcal{A}$ queried $H(m_{i^*}, \sigma_{i^*}^0)$. Let $Ind(\omega, \rho_{H_1}, \rho_H)$ be the index of this query, i.e. $q_H^{Ind(\omega, \rho_{H_1}, \rho_H)} = (m_{i^*}, \sigma_{i^*}^0)$ and let $Ind(\omega, \rho_{H_1}, \rho_H) = \infty$ if the query is never made by $\mathcal{A}$. Then we can define the sets

$$\mathcal{S} = \{(\omega, \rho_{H_1}, \rho_H) \mid \mathcal{A} \text{ succeeds and } Ind(\omega, \rho_{H_1}, \rho_H) \neq \infty\}$$

and

$$\mathcal{S}_i = \{(\omega, \rho_{H_1}, \rho_H) \mid \mathcal{A} \text{ succeeds and } Ind(\omega, \rho_{H_1}, \rho_H) = i\} \qquad \text{for } i \in \{1, \ldots, q_H\}.$$

With these definitions, $\nu := \Pr(\mathcal{S}) = \frac{\varepsilon}{q_{AI}} - \frac{1}{p}$ is the probability that $\mathcal{A}$ is succesful, $H(m_{i^*}, \sigma_{i^*}^0)$ was queried by $\mathcal{A}$ and $(m_{i^*}, \sigma_{i^*}^0, c, \sigma_{i^*}^1)$ is a signature for $item^*$. Now let I be the set of the most likely indices i:

$$I = \left\{ i \mid \Pr(\mathcal{S}_i \mid \mathcal{S}) \geq \frac{1}{2 \cdot q_H} \right\}.$$

Then

$$\Pr(Ind(\omega, \rho_{H_1}, \rho_H) \in I \mid \mathcal{S}) = \sum_{i \in I} \Pr(\mathcal{S}_i \mid \mathcal{S}) = 1 - \sum_{i \notin I} \Pr(\mathcal{S}_i \mid \mathcal{S}) \geq 1 - q_H \cdot \frac{1}{2 \cdot q_H} = \frac{1}{2}.$$

Let $\rho_H|_a^b$ denote the restriction of ρ_H to its elements $\rho_H^a, \rho_H^{a+1}, \ldots, \rho_H^b$ and let us define the sets

$$X = (\omega, \rho_{H_1}, \rho_H|_1^{j-1})$$
$$Y = (\rho_H|_j^{q_H})$$
$$\Omega_j = \left\{ (x,y) \in X \times Y \,\middle|\, \Pr_{y' \in Y}((x,y') \in \mathcal{S}_j) \geq \frac{\nu}{4 \cdot q_H} \right\} \qquad \text{for each } j \in I.$$

Then $\Pr(\mathcal{S}_j) = \Pr(\mathcal{S}_j) \cdot \Pr(\mathcal{S} \mid \mathcal{S}_j) = \Pr(\mathcal{S}) \cdot \Pr(\mathcal{S}_j \mid \mathcal{S}) \geq \frac{\nu}{2 \cdot q_H}$ and $\Pr(\Omega_j \mid \mathcal{S}_j) \geq \frac{1}{2}$, by the splitting lemma [27]. Furthermore, it holds

$$\begin{aligned}\Pr(\exists j \in I \colon \Omega_j \cap \mathcal{S}_j \mid \mathcal{S}) &= \Pr\left(\bigcup_{j \in I} (\Omega_j \cap \mathcal{S}_j) \mid \mathcal{S}\right) = \sum_{j \in I} \Pr(\Omega_j \cap \mathcal{S}_j \mid \mathcal{S}) \\ &= \sum_{j \in I} \Pr(\Omega_j \mid \mathcal{S}_j) \cdot \Pr(\mathcal{S}_j \mid \mathcal{S}) \geq \frac{1}{2} \cdot \sum_{j \in I} \Pr(\mathcal{S}_j \mid \mathcal{S}) \geq \frac{1}{4}\end{aligned}$$

because the subsets $\mathcal{S}_j$ are disjoint. This means, with probability $\frac{\nu}{4}$ algorithm $\mathcal{A}$ succeeds by outputting a signature $(m_{i^*}, \sigma_{i^*}^0, c, \sigma_{i^*}^1)$, which is derived from a tuple $(x,y) \in \Omega_j$ for some $j \in I$. Now we rewind $\mathcal{A}$ and $\mathcal{B}$ to the j'th query to H and proceed with an oracle vector ρ'_H, where $\rho'_H|_1^{j-1} = \rho_H|_1^{j-1}$ and $\rho'^k_H \neq \rho^k_H$ for all $k = j, \ldots, q_H$. This means, we run $\mathcal{A}$ and $\mathcal{B}$ with $(\omega, \rho_{H_1}, (\rho_H|_1^{j-1}, \rho'_H|_j^{q_H})) = (x, y') \in X \times Y$ and we know that $\Pr((x,y') \in \mathcal{S}_j) \geq \frac{\nu}{4 \cdot q_h}$ from the definition of Ω_j. Hence, $\mathcal{A}$ succeeds a second time by outputting a signature $(m_{i^*}, \sigma_{i^*}^0, c', \sigma_{i^*}^{1'})$ where $c \neq c'$ and $\sigma_{i^*}^1 \neq \sigma_{i^*}^{1'}$ (of course, $\mathcal{A}$ again outputs q message-signaute pairs, but we are only interested in the one with index i^*).
By using the extractor of Lemma 5.3, we obtain from $(\sigma_{i^*}^0, c, \sigma_{i^*}^1)$ and $(\sigma_{i^*}^0, c', \sigma_{i^*}^{1'})$ a q-SDH tuple (A^*, x^*, y^*). This can be transformed into a solution to $\mathcal{B}$'s q-SDH problem as follows:

$$\begin{aligned}A^* &= \psi\left[\left(G_2^{\alpha \cdot \prod_{i=1}^{q-1}(\hat{\gamma} - x_k + x_i) - y_k \cdot \prod_{i=1, i \neq k}^{q-1}(\hat{\gamma} - x_k + x_i)} \cdot G_2^{y^* \cdot \prod_{i=1, i \neq k}^{q-1}(\hat{\gamma} - x_k + x_i)}\right)^{\frac{1}{\gamma + x^*}}\right] \\ &= \psi\left[\left(G_2^{\alpha \cdot \prod_{i=1}^{q-1}(\hat{\gamma} - x_k + x_i) + (y^* - y_k) \cdot \prod_{i=1, i \neq k}^{q-1}(\hat{\gamma} - x_k + x_i)}\right)^{\frac{1}{\hat{\gamma} - x_k + x^*}}\right]. \qquad (4)\end{aligned}$$

Let $f(X) := \prod_{i=1}^{q-1}(X - x_k + x_i)$ and $g(X) := \prod_{i=1, i \neq k}^{q-1}(X - x_k + x_i)$ then

$$f(\hat{\gamma}) \cdot \frac{1}{\hat{\gamma} - x_k + x^*} = \tau(\hat{\gamma}) + \frac{\beta}{\hat{\gamma} - x_k + x^*}$$

and

$$g(\hat{\gamma}) \cdot \frac{1}{\hat{\gamma} - x_k + x^*} = \tau'(\hat{\gamma}) + \frac{\beta'}{\hat{\gamma} - x_k + x^*}$$

for some polynomials τ and τ' of degree at most $q-2$, where β or β' equals 0, iff $(\hat{\gamma} - x_k + x^*)$ is a factor of $f(\hat{\gamma})$ or $g(\hat{\gamma})$. These cases will be discussed later. Using this representation we

obtain

$$A^* = \psi\left[G_2^{\alpha\cdot\left(\tau(\hat{\gamma})+\frac{\beta}{\hat{\gamma}-x_k+x^*}\right)+(y^*-y_k)\cdot\left(\tau'(\hat{\gamma})+\frac{\beta'}{\hat{\gamma}-x_k+x^*}\right)}\right]$$

and we can define

$$\begin{aligned}
A &= \psi\left[\left(\frac{G_2^{\alpha\cdot\left(\tau(\hat{\gamma})+\frac{\beta}{\hat{\gamma}-x_k+x^*}\right)+(y^*-y_k)\cdot\left(\tau'(\hat{\gamma})+\frac{\beta'}{\hat{\gamma}-x_k+x^*}\right)}}{G_2^{\alpha\cdot\tau(\hat{\gamma})+(y^*-y_k)\cdot\tau'(\hat{\gamma})}}\right)^{\frac{1}{\alpha\cdot\beta+(y^*-y_k)\cdot\beta'}}\right] \\
&= \psi\left[\left(\frac{G_2^{\alpha\cdot\tau(\hat{\gamma})+\alpha\cdot\frac{\beta}{\hat{\gamma}-x_k+x^*}+(y^*-y_k)\cdot\tau'(\hat{\gamma})+(y^*-y_k)\cdot\frac{\beta'}{\hat{\gamma}-x_k+x^*}}}{G_2^{\alpha\cdot\tau(\hat{\gamma})+(y^*-y_k)\cdot\tau'(\hat{\gamma})}}\right)^{\frac{1}{\alpha\cdot\beta+(y^*-y_k)\cdot\beta'}}\right] \\
&= \psi\left[\left(G_2^{\alpha\cdot\frac{\beta}{\hat{\gamma}-x_k+x^*}+(y^*-y_k)\cdot\frac{\beta'}{\hat{\gamma}-x_k+x^*}}\right)^{\frac{1}{\alpha\cdot\beta+(y^*-y_k)\cdot\beta'}}\right] \\
&= \psi\left[\left(G_2^{\frac{1}{\hat{\gamma}-x_k+x^*}\cdot(\alpha\cdot\beta+(y^*-y_k)\cdot\beta')}\right)^{\frac{1}{\alpha\cdot\beta+(y^*-y_k)\cdot\beta'}}\right] \\
&= \psi\left[G_2^{\frac{1}{\hat{\gamma}-x_k+x^*}}\right].
\end{aligned}$$

Hence, $(A, x^* - x_k)$ is a solution to $\mathcal{B}$'s q-SDH problem.
Now we discuss the different cases that can occur during the described transformation.

Case 1: $(A^*, x^*, y^*) \in \{(A_j, x_j, y_j)\}_{j=1}^{q-1}$: Obviously, if (A^*, x^*, y^*) is one of the triples $\mathcal{B}$ generated himself, no new information is obtained from (A^*, x^*, y^*). Hence, $\mathcal{B}$ cannot compute A and has to abort.

Case 2: $x^* \notin \{x_j\}_{j=1}^{q-1}$: In this case the values β and β' are not equal to 0. Hence, the value A can be computed as described above and $(A, x^* - x_k)$ is a solution to $\mathcal{B}$'s q-SDH problem.

Case 3: $x^* \in \{x_j\}_{j=1}^{q-1}$: This case has to be devided into two different subcases:

a) $x^* \neq x_k$: Since x^* is equal to x_j for some $j \neq k$, $(\hat{\gamma} - x_k + x_j)$ is a factor of both polynoms $f(\hat{\gamma})$ and $g(\hat{\gamma})$. Hence, it holds $\beta = \beta' = 0$, A cannot be computed and $\mathcal{B}$ has to abort.

b) $x^* = x_k$: In this case $(\hat{\gamma} - x_k + x^*) = \hat{\gamma}$ is a factor of $f(\hat{\gamma})$, but not one of $g(\hat{\gamma})$. Hence, $\beta = 0$ and $\beta' \neq 0$ holds. Also $y^* \neq y_k$ (because otherwise A^* would be equal to A_k) and $(y^* - y_k) \cdot \beta' \neq 0$ holds, so $(A, 0)$ is a solution to $\mathcal{B}$'s q-SDH problem.

We know that $\mathcal{B}$ obtains a tuple (A^*, x^*, y^*) with a probability of at least $\frac{\nu^2}{16 \cdot q_H}$. Because $\mathcal{A}$ was successful it holds $y^* \notin \{y_i\}_{i=1}^{q-1}$. Hence, case 1 does not occur. If case 2 occurs, $\mathcal{B}$ can compute a solution to its q-SDH problem with probability 1. For case 3 $\Pr(x^* = x_k) = \frac{1}{q-1}$ holds and either case 2 or case 3 occurs with a probability of at least $\frac{1}{2}$.
Putting all together, assuming the more pessimistic scenario of case 3, $\mathcal{B}$ can compute a solution to its q-SDH problem with a probability ε' of at least

$$\varepsilon' \geq \frac{\nu^2}{16 \cdot q_H} \cdot \frac{1}{q-1} \cdot \frac{1}{2} = \left(\frac{\varepsilon}{q_{AI}} - \frac{1}{p}\right)^2 \cdot \frac{1}{32 \cdot q_H \cdot (q-1)}.$$

The transformation of $\mathcal{B}$'s q-SDH instance into $q-1$ tuples (A_j, x_j, y_j) and an *item*-based public key can be done in constant time, for fixed q. Also algorithm $\mathcal{B}$ can respond to an oracle query of $\mathcal{A}$ in constant time, and there are at most $\mathcal{Q}$ of such queries. The computation of (A, x) using the extracted values (A^*, x^*, y^*) needs constant time, too. Hence, $\mathcal{B}$ can solve q-SDH in time $t' = t + \mathcal{Q} \cdot \mathcal{O}(1)$. □

Lemma 6.3:
If q-SDH (t', ε')-holds in $(\mathbb{G}_1, \mathbb{G}_2)$, then the reputation system defined in Section 6 is (t, ε)-traceable, where $t = t' - \mathcal{Q} \cdot \mathcal{O}(1)$ and $\varepsilon = q_{AI} \cdot q_{AU} \cdot \sqrt{\varepsilon' \cdot (64 \cdot q_H) \cdot (q-1)} + \frac{q_{AI} \cdot q_{AU}}{p}$. Here q_{AU} is the number of oracle queries to AddU, q_{AI} is the number of oracle queries to AddItem, the number of oracle queries to SndToGM is a most $q - 1 - q_{AU}$, q_H is the number of queries to the random oracle H and p is the size of the groups $\mathbb{G}_1$ and $\mathbb{G}_2$.

Proof. Suppose $\mathcal{A}$ is an adversary that (t, ε)-solves the traceability of the reputation system. Then we can construct an adversary $\mathcal{B}$ that solves q-SDH in $(\mathbb{G}_1, \mathbb{G}_2)$ with advantage ε' of at least $\varepsilon' = \left(\frac{\varepsilon}{q_{AI} \cdot q_{AU}} - \frac{1}{p}\right)^2 \cdot \frac{1}{64 \cdot q_H} \cdot \frac{1}{q-1}$ in time $t' = t + \mathcal{Q} \cdot \mathcal{O}(1)$. Without loss of generality we assume that $\mathcal{A}$ creates exactly $q-1$ users via the AddU and SndToGM oracles.
Analogously to [6] we have to distinguish between two different forger types: the *Type-I forger* outputs a valid message-signature pair (m, σ), for some *item* of his choice, such that the Open algorithm outputs `failure`; the *Type-II forger* outputs a valid message-signature pair (m, σ), for some *item* of his choice, that can be traced back to an honest user. Hence, $\mathcal{B}$ guesses the forger type, with probability $\frac{1}{2}$, i.e. $b \xleftarrow{\$} \{I, II\}$, and behaves slightly different in the two cases. Algorithm $\mathcal{B}$ transforms the q-SDH problem into tuples (A_j, x_j, y_j), for $j = 1, \ldots, q-1$, and values $(g_1, g_2, \hat{h}, h, W)$ using the same technique as in the proof of public linkability (Lemma 6.2). Furthermore, $\mathcal{B}$ guesses an $item^*$ as the *item* for which $\mathcal{A}$ will output the message-signature pair as its solution to the traceability experiment. This can be done as in the proof of Lemma 6.2 by choosing $\ell \xleftarrow{\$} \{1, \ldots, q_{AI}\}$ and handling the ℓ'th query to the AddItem oracle appropriately. To generate the group manager's public key, $\mathcal{B}$ selects $w \xleftarrow{\$} \mathbb{G}_1, \zeta, \xi_1, \xi_2 \xleftarrow{\$} \mathbb{Z}_p$ and computes

$$u := w^{\frac{1}{\xi_1}} \qquad v := w^{\frac{1}{\xi_2}} \qquad \hat{d} := \hat{h}^{\frac{1}{\zeta}} \qquad d := \psi(\hat{d}).$$

Now $\mathcal{B}$ sets $\mathcal{HU} := \emptyset$, $\mathcal{CU} := \emptyset$, $\mathcal{RL} := \emptyset$, $\mathit{ItemList} := \emptyset$, $\mathit{reg} := \emptyset$, $\mathcal{RU} := \emptyset$, $\mathcal{JIU} := \emptyset$, $\mathcal{GS} := \emptyset$ and $gmpk := (u, v, w, h, d, \hat{d})$. In the case that $b = II$, $\mathcal{B}$ selects $A_q \xleftarrow{\$} \mathbb{G}_1$ and $y_q \xleftarrow{\$} \mathbb{Z}_p$, sets $x_q := \star$ and guesses the honest user i^* for which $\mathcal{A}$ will output the forged signature σ. This can be done by choosing $\ell_1 \xleftarrow{\$} \{1, \ldots, q_{AU}\}$ and handling the ℓ_1'th query to the AddU oracle appropriately. The number of already registered users is counted using the variable $\hat{j}$, which is initially set to 0.
Now $\mathcal{B}$ gives the group manager's public key $gmpk$ to $\mathcal{A}$ and starts to interact with $\mathcal{A}$ via the oracles. $\mathcal{A}$'s queries are answered as follows:
$\boldsymbol{H(\cdot)}$: $\mathcal{B}$ chooses $c \xleftarrow{\$} \mathbb{Z}_p$, gives c to $\mathcal{A}$ and ensures to respond identically to repeated queries.
$\boldsymbol{H_1(item)}$: $\mathcal{B}$ chooses $\hat{f} \xleftarrow{\$} \mathbb{G}_2$, gives $\hat{f}$ to $\mathcal{A}$ and ensures to respond identically to repeated queries.
AddItem(*item*): On the ℓ'th query $\mathcal{B}$ sets $item^* := item$ and $ipk[item^*] := (g_1, g_2, W)$, adds $ipk[item^*]$ to the *ItemList* and returns $ipk[item^*]$. For every other query $\mathcal{B}$ executes exactly

the oracle as defined in Figure 3.
AddU(i): If $b = II$, on the ℓ_1'th query $\mathcal{B}$ sets $i^* := i, usk[i^*] := y_q, upk[i^*] := h^{y_q}, reg[i^*] := (i^*, upk[i^*])$ and returns $upk[i^*]$. In any other case, $\mathcal{B}$ sets $\hat{j} := \hat{j} + 1$ and executes the oracle defined in Figure 3, using $y_{\hat{j}}$ as the secret key of user i.
SndToGM(i, M_{GM}): Algorithm $\mathcal{B}$ sets $\hat{j} := \hat{j} + 1$ and executes the oracle defined in Figure 3, using $y_{\hat{j}}$ as the secret key of user i.
USK(i): If $b = II$ and $i = i^*$ then $\mathcal{B}$ declares failure and exits. Otherwise, algorithm $\mathcal{B}$ responds by running exactly the defined oracle from Figure 3.
GSK($i, item$): Algorithm $\mathcal{B}$ responds by running exactly the defined oracle from Figure 3.
RevU(i): Algorithm $\mathcal{B}$ responds by running exactly the defined oracle from Figure 3.
SndToKI($i, item, upk[i], M_{KI}$): If $item = item^*$ then $\mathcal{B}$ runs the defined oracle using $gsk[i, item^*] := (A_j, x_j)$, where A_j and x_j correspond to user i's secret key $usk[i] = y_j$ defined in the AddU or SndToGM oracle for user i. In any other case, $\mathcal{B}$ responds by running exactly the defined oracle from Figure 3.
Open($item, m, \sigma$): Algorithm $\mathcal{B}$ responds by running exactly the defined oracle from Figure 3.
GSig($i, item, m$): Algorithm $\mathcal{B}$ has to handle two different cases:

- If $(b = I) \vee (b = II \wedge (i \neq i^* \vee item \neq item^*))$ then $\mathcal{B}$ responds as defined in Figure 3. If for an honest user i the signing key $gsk[i, item]$ is empty, such a key can be generated, before the signature is created. If $item = item^*$ the tuple (A_j, x_j, y_j) corresponding to $usk[i] = y_j$ has to be used as secret signing key.

- If $b = II, i = i^*$ and $item = item^*$ then $\mathcal{B}$ simulates the signature using the simulator of Lemma 5.2. Here the first step of the simulator is replaced by setting the following values: $\mathcal{B}$ chooses α, β, $\mu \xleftarrow{\$} \mathbb{Z}_p$ and computes $T_1 := u^\alpha$, $T_2 := v^\beta$, $T_3 := A_{i^*_{item^*}} \cdot w^{\alpha+\beta}$, $T_4 := d^\mu$, $T_5 := \psi(H_1(item^*))^{\mu + y_{i^*}}$. Afterwards, $\mathcal{B}$ chooses $c, s_\alpha, s_\beta, s_x, s_y, s_\mu, s_{\delta_1}, s_{\delta_2} \xleftarrow{\$} \mathbb{Z}_p$ and computes $R_1, \ldots, R_7$ using the verification equations from the Verify algorithm. To ensure the signature is valid, $\mathcal{B}$ patches H at (M, $item^*$, T_1, T_2, T_3, T_4, T_5, R_1, R_2, R_3, R_4, R_5, R_6, R_7) to equal c. If this causes a collision, $\mathcal{B}$ declares failure and exits. Finally, $\mathcal{B}$ gives $\sigma := (item^*, T_1, T_2, T_3, T_4, T_5, c, s_\alpha, s_\beta, s_x, s_y, s_\mu, s_{\delta_1}, s_{\delta_2})$ to $\mathcal{A}$.

At some point $\mathcal{A}$ outputs a triple $(item, m, \sigma)$. If $item \neq item^*$ or the signature is invalid, verified with an empty revocation list $\mathcal{RL}$, $\mathcal{B}$ decleares failure and exits. Otherwise, depending on $b \in \{I, II\}$, $\mathcal{B}$ has to distinguish two different cases:

$b = I$: If σ opens to some $A^* \in \{A_j\}_{j=1}^{q-1}$, $\mathcal{B}$ declares failure and exits. Otherwise, $\mathcal{A}$ successfully forged a signature for a non-existing user.

$b = II$: If σ opens to some $A^* \neq A_q$, $\mathcal{B}$ declares failure and exits. Otherwise, $\mathcal{A}$ successfully forged a signature for user i^*.

For a *Type-I forger* the environment is simulated perfectly, because $\mathcal{B}$ knows all secret and public keys. Hence, $\mathcal{B}$ is always able to compute correct and properly distributed responses to $\mathcal{A}$'s queries and $\mathcal{A}$ outputs a valid forgery $(item^*, m^*, \sigma^*)$, for the guessed $item^*$, with a probability of at least $\frac{\varepsilon}{q_{AI}}$.
For a *Type-II forger* the environment is simulated perfectly unless $\mathcal{A}$ queries the USK oracle for user i^* or a collision occurs while simulating a signature (since the probability of such a collision

is negligable we will ignore it in the analysis). Hence, $\mathcal{A}$ outputs a valid forgery $(item^*, m^*, \sigma^*)$, for the guessed $item^*$, that traces to i^* with a probability of at least $\frac{\varepsilon}{q_{AI} \cdot q_{AU}}$.
Now we apply the Forking Lemma [27] to obtain a second solution to the traceability experiment which can be used to solve q-SDH. The needed technique is exactly the same as in the proof of public linkability in Lemma 6.2. That means, with $\nu = \frac{\varepsilon}{q_{AI}} - \frac{1}{p}$ for a *Type-I forger* and $\nu = \frac{\varepsilon}{q_{AI} \cdot q_{AU}} - \frac{1}{p}$ for a *Type-II forger*, we obtain a q-SDH tuple $(A^*.x^*, y^*)$ with a probability of at least $\frac{\nu^2}{16 \cdot q_H}$. This tuple is not one of the tuples $\mathcal{B}$ created, because otherwise the forger would not be successful. Hence, either case 2 or case 3 described in Lemma 6.2 occurs with a probability of at least $\frac{1}{2}$. Moreover, $\mathcal{B}$ guesses the correct forger type with a probability of at least $\frac{1}{2}$ and can compute a solution (A, x) to its q-SDH problem with a probability ε' of at least

$$\varepsilon' \geq \frac{\nu^2}{16 \cdot q_H} \cdot \frac{1}{q-1} \cdot \frac{1}{2} \cdot \frac{1}{2} = \left(\frac{\varepsilon}{q_{AI} \cdot q_{AU}} - \frac{1}{p} \right)^2 \cdot \frac{1}{64 \cdot q_H} \cdot \frac{1}{q-1}$$

assuming the more pessimistic scenario of case 3 and a *Type-II forger*.
As shown in Lemma 6.2, the transformation of $\mathcal{B}$'s q-SDH instance can be done in constant time, for fixed q. Also algorithm $\mathcal{B}$ can respond to an oracle query of $\mathcal{A}$ in constant time, and there are at most $\mathcal{Q}$ of such queries. The computation of (A, x) using the extracted values (A^*, x^*, y^*) needs constant time, too. Hence, $\mathcal{B}$ can solve q-SDH in time $t' = t + \mathcal{Q} \cdot \mathcal{O}(1)$. □

Lemma 6.4:
If the discrete logarithm problem is (t', ε')-hard in $\mathbb{G}_2$, then the reputation system defined in Section 6 is (t, ε)-strong exculpable, where $t = t' - \mathcal{Q} \cdot \mathcal{O}(1)$ and $\varepsilon = q_{AU} \cdot \sqrt{\varepsilon' \cdot 16 \cdot q_H} + \frac{q_{AU}}{p}$. Here q_{AU} is the number of oracle queries made by the adversary to AddU, q_H is the number of queries to the random oracle H and p is the size of the groups $\mathbb{G}_1$ and $\mathbb{G}_2$.

Proof. Suppose $\mathcal{A}$ is an adversary that (t, ε)-breaks the strong-exculpability of the reputation system above. Then we construct an adversary $\mathcal{B}$ that solves the discrete logarithm problem in $\mathbb{G}_2$ with advantage at least $\varepsilon' = \left(\frac{\varepsilon}{q_{AU}} - \frac{1}{p} \right)^2 \cdot \frac{1}{16 \cdot q_H}$ in time $t' = t + \mathcal{Q} \cdot \mathcal{O}(1)$.
$\mathcal{B}$ sets $\mathcal{HU} := \emptyset$, $\mathcal{CU} := \emptyset$, $\mathcal{RL} := \emptyset$, $ItemList := \emptyset$, $reg := \emptyset$, $\mathcal{RU} := \emptyset$, $\mathcal{JIU} := \emptyset$, $\mathcal{GS} := \emptyset$ and guesses the user i^* for which the adversary $\mathcal{A}$ will output a signature σ as its solution to the strong-exculpability experiment. This can be done by choosing $i^* \xleftarrow{\$} \{1, ..., q_{AU}\}$ and handling the i^*-th query to the GSig, USK, RevU, and AddU oracles appropriately.
The proof is divided into three parts. In the first part of the proof, we describe a simulation for $\mathcal{B}$ interacting with $\mathcal{A}$.
Setup: $\mathcal{B}$ is given $(\hat{h}, \mathcal{D})$ as an instance of the discrete logarithm problem in $\mathbb{G}_2$, where $\hat{h} \xleftarrow{\$} \mathbb{G}_2$ and $\mathcal{D} \xleftarrow{\$} \mathbb{G}_2$. The goal of $\mathcal{B}$ is to output the logarithm of $\mathcal{D}$ with respect to $\hat{h}$. Pick $w \xleftarrow{\$} \mathbb{G}_1$, $\xi_1, \xi_2 \xleftarrow{\$} \mathbb{Z}_p$ and compute $u := w^{\frac{1}{\xi_1}}$, $v := w^{\frac{1}{\xi_2}}$. Select $\zeta \xleftarrow{\$} \mathbb{Z}_p$. Set $h = \psi(\hat{h})$, $\hat{d} = \hat{h}^{\frac{1}{\zeta}}$, and $d = \psi(\hat{d})$. Set $gmpk := (u, v, w, h, d, \hat{d})$ and $gmsk := (\xi_1, \xi_2, \zeta)$ as the group manager's public and secret keys. Make a list of pairs $(upk[i], usk[i])$ for $i = 1, ..., q_{AU}$ as follows: For i^*, set $usk[i] = \star$ indicating that y_i corresponding to $upk[i] = \psi(\mathcal{D})$ is not known. Otherwise $(upk[i], usk[i])$ is a pair constructed as $y_i \xleftarrow{\$} \mathbb{Z}_p$, $upk[i] = h^{y_i}$ and $usk[i] = y_i$. To run $\mathcal{A}$, give $gmpk$ to $\mathcal{A}$. $\mathcal{A}$'s oracle queries are answered as follows:

$\boldsymbol{H(\cdot)}$: $\mathcal{B}$ responds with an element chosen uniformly at random of $\mathbb{Z}_p$ and ensures that repeated queries are answered consistently.
$\boldsymbol{H_1(item)}$: When $\mathcal{A}$ asks for a hash of an *item*, $\mathcal{B}$ chooses $r_{item} \xleftarrow{\$} \mathbb{Z}_p$, responds with $\hat{f} = \hat{h}^{r_{item}}$, and ensures that repeated queries are answered consistently.
$\mathbf{GSig(}\boldsymbol{i, item, m}\mathbf{)}$: $\mathcal{B}$ checks, if $upk[i] \in IL_{item}$. If not, $\mathcal{B}$ answers by running exactly the defined oracle from Figure 3. Otherwise: If $i \neq i^*$, $\mathcal{B}$ runs the reputation system signing algorithm to obtain a signature σ and gives σ to $\mathcal{A}$. If $i = i^*$, $\mathcal{B}$ chooses $\alpha, \beta, \mu \xleftarrow{\$} \mathbb{Z}_p$, uses $A_{i_{item}}$ and computes $T_1 := u^{\alpha}$, $T_2 := v^{\beta}$, $T_3 := A_{i_{item}} \cdot w^{\alpha+\beta}$, $T_4 := d^{\mu}$ and $T_5 := \psi(\hat{h}^{r_{item} \cdot \mu} \cdot \mathcal{D}^{r_{item}})$. The value T_5 is correct, since

$$T_5 = \psi(\hat{h}^{r_{item} \cdot \mu} \cdot \mathcal{D}^{r_{item}}) = \psi(\hat{h}^{r_{item} \cdot \mu} \cdot \hat{h}^{y_i \cdot r_{item}}) = \psi(\hat{h}^{r_{item}})^{\mu + y_i} = \psi(\hat{f})^{\mu + y_i}.$$

Now, $\mathcal{B}$ runs the protocol simulator with T_1, T_2, T_3, T_4, T_5 and obtains a transcript (T_1, T_2, T_3, T_4, T_5, R_1, R_2, R_3, R_4, R_5, R_6, R_7, c, s_α, s_β, s_x, $s_{\hat{y}}$, s_μ, s_{δ_1}, s_{δ_2}). To ensure that the signature $\sigma :=$(*item*, T_1, T_2, T_3, T_4, T_5, c, s_α, s_β, s_x, s_y, s_μ, s_{δ_1}, s_{δ_2}) is valid, $\mathcal{B}$ patches H at (M, *item* ,T_1,T_2,T_3,T_4,T_5, R_1,R_2,R_3,R_4,R_5,R_6,R_7) to equal c. If there is a collision, $\mathcal{B}$ declares failure and exits. This happens only with negligible probability, thus we ignore it in the analysis. Otherwise $\mathcal{B}$ returns σ to $\mathcal{A}$.
$\mathbf{USK(}\boldsymbol{i}\mathbf{)}$: If $i \neq i^*$, $\mathcal{B}$ looks up $usk[i]$ in the list of key pairs and returns y_i to $\mathcal{A}$. If $i = i^*$, $\mathcal{B}$ declares failure and exits.
$\mathbf{RevU(}\boldsymbol{i}\mathbf{)}$: If $i \neq i^*$, $\mathcal{B}$ looks up $upk[i]$ in $reg[i]$. Using $gmsk$, $\mathcal{B}$ computes $D_i := upk[i]^{\frac{1}{\zeta}} = (h^{y_i})^{\frac{1}{\zeta}} = d^{y_i}$, adds the revocation token $grt[i] := D_i$ to the revocation list, and gives D_i as response to $\mathcal{A}$. If $i = i^*$, $\mathcal{B}$ sets $grt[i^*] := \psi(\mathcal{D})^{\frac{1}{\zeta}}$ and gives it to $\mathcal{A}$.
$\mathbf{AddU(}\boldsymbol{i}\mathbf{)}$: If $i \neq i^*$, $\mathcal{B}$ looks up the $upk[i]$ in the list of key pairs and answers with $upk[i]$. If $i = i^*$, $\mathcal{B}$ gives $\psi(\mathcal{D})$ to $\mathcal{A}$.
The oracle queries to SndToGM, WItemList, WIdentList, and Open are answered by $\mathcal{B}$ running exactly the defined oracles from Figure 3.
Finally $\mathcal{A}$ outputs $\sigma :=$($item, T_1, T_2, T_3, T_4, T_5, c, s_\alpha, s_\beta, s_x, s_y, s_\mu, s_{\delta_1}, s_{\delta_2}$) on M for an *item*.
In the second part of the proof, we analyze the simulation above. Except for USK and GSig, $\mathcal{B}$ can answer all queries exactly as defined in Figure 3. Hence, the simulation is perfect, unless $\mathcal{A}$ queries the USK oracle for user i^*. In case $i \neq i^*$ the signing oracle GSig produces signatures by following the signing algorithm. Hence, those signatures are properly distributed. In case $i = i^*$ the signature is obtain by the simulator of Protocol 5.1. T_1, T_2, T_3,T_4, and T_5 are properly distributed, so by using the simulator with these values, we obtain a signature σ that is distributed as in the real reputation scheme. Hence, the probability that $\mathcal{A}$ outputs a valid signature σ for an honest user is ε. The probability that this signature traces to user i^* is at least $\frac{\varepsilon}{q_{AU}}$.
In the third part of this proof, we use the Forking Lemma [27] to obtain a solution for the discrete logarithm problem. The technique is exactly the same as in the proof of public linkability (Lemma 6.2).
Using the Forking Lemma one can compute two forged signatures (σ_0, c, σ_1) and $(\sigma_0, c', \sigma_1')$ that trace to the honest user i^* with probability

$$\frac{\nu^2}{16 \cdot q_H} = \left(\frac{\varepsilon}{q_{AU}} - \frac{1}{p}\right)^2 \cdot \frac{1}{16 \cdot q_H}.$$

Using the extractor from Lemma 5.3, we obtain a triple (A, x, y), where y is the secret key corresponding to i^*'s public key $upk[i^*] = \psi(\mathcal{D})$. Hence $y = \log_h(\psi(\mathcal{D}))$, and $y = \log_{\hat{h}}(\mathcal{D})$, as required.
Algorithm $\mathcal{B}$ can respond to an oracle query of $\mathcal{A}$ in constant time, and there are at most $\mathcal{Q}$ of such queries. The computation of (A, x, y) needs constant time, too. Hence, $\mathcal{B}$ can solve the discrete logarithm problem in time $t' = t + \mathcal{Q} \cdot \mathcal{O}(1)$. □

References

[1] Michel Abdalla, Jee Hea An, Mihir Bellare, and Chanathip Namprempre. From Identification to Signatures via the Fiat-Shamir Transform: Minimizing Assumptions for Security and Forward-Security. In *EUROCRYPT 2002*, volume 2332 of *LNCS*, pages 418–433. Springer, 2002.

[2] Elli Androulaki, SeungGeol Choi, Steven M. Bellovin, and Tal Malkin. Reputation Systems for Anonymous Networks. In *Privacy Enhancing Technologies*, volume 5134 of *LNCS*, pages 202–218. Springer, 2008.

[3] Giuseppe Ateniese, Jan Camenisch, Marc Joye, and Gene Tsudik. A Practical and Provably Secure Coalition-Resistant Group Signature Scheme. In *CRYPTO 2000*, volume 1880 of *LNCS*, pages 255–270. Springer, 2000.

[4] Mihir Bellare, Daniele Micciancio, and Bogdan Warinschi. Foundations of Group Signatures: Formal Definitions, Simplified Requirements, and a Construction Based on General Assumptions. In *EUROCRYPT 2003*, pages 614–629. Springer, 2003.

[5] Mihir Bellare, Haixia Shi, and Chong Zhang. Foundations of Group Signatures: The Case of Dynamic Groups. In *CT-RSA 2005*, volume 3376 of *LNCS*, pages 136–153. Springer, 2005.

[6] Dan Boneh, Xavier Boyen, and Hovav Shacham. Short Group Signatures. In *CRYPTO 2004*, volume 3152 of *LNCS*, pages 41–55. Springer, 2004.

[7] Dan Boneh, Ben Lynn, and Hovav Shacham. Short Signatures from the Weil Pairing. In *ASIACRYPT 2001*, volume 2248 of *LNCS*, pages 514–532. Springer, 2001.

[8] Dan Boneh and Hovav Shacham. Group Signatures with Verifier-local Revocation. In *CCS 2004*, pages 168–177. ACM, 2004.

[9] Sherman S.M. Chow, Willy Susilo, and Tsz Hon Yuen. Escrowed Linkability of Ring Signatures and Its Applications. In *VIETCRYPT 2006*, volume 4341 of *LNCS*, pages 175–192. Springer, 2006.

[10] Sebastian Clauß, Stefan Schiffner, and Florian Kerschbaum. k-anonymous Reputation. In *ASIA CCS 2013*, pages 359–368. ACM, 2013.

[11] Cécile Delerablée and David Pointcheval. Dynamic Fully Anonymous Short Group Signatures. In *VIETCRYPT 2006*, volume 4341 of *LNCS*, pages 193–210. Springer, 2006.

[12] Chrysanthos Dellarocas. Immunizing Online Reputation Reporting Systems Against Unfair Ratings and Discriminatory Behavior. In *EC 2000*, pages 150–157. ACM, 2000.

[13] Roger Dingledine, Nick Mathewson, and Paul Syverson. Reputation in P2P Anonymity Systems. In *Workshop on Economics of Peer-to-Peer Systems*, volume 92, 2003.

[14] Amos Fiat and Adi Shamir. How To Prove Yourself: Practical Solutions to Identification and Signature Problems. In *CRYPTO 1986*, volume 263 of *LNCS*, pages 186–194. Springer, 1987.

[15] Matthew Franklin and Haibin Zhang. Unique Group Signatures. In *ESORICS 2012*, volume 7459 of *LNCS*, pages 643–660. Springer, 2012.

[16] Eiichiro Fujisaki and Koutarou Suzuki. Traceable Ring Signature. In *PKC 2007*, volume 4450 of *LNCS*, pages 181–200. Springer, 2007.

[17] Micheal T. Goodrich and Florian Kerschbaum. Privacy-Enhanced Reputation-Feedback Methods to Reduce Feedback Extortion in Online Auctions. In *CODASPY 2011*, pages 273–282. ACM, 2011.

[18] Jung Yeon Hwang, Sokjoon Lee, Byung-Ho Chung, Hyun Sook Cho, and DaeHun Nyang. Group signatures with controllable linkability for dynamic membership. *Information Sciences*, 222:761–778, 2013.

[19] Audun Jøsang and Roslan Ismail. The Beta Reputation System. In *BLED 2002*, pages 41–55, 2002.

[20] Sepandar D. Kamvar, Mario T. Schlosser, and Hector Garcia-Molina. The EigenTrust Algorithm for Reputation Management in P2P Networks. In *WWW 2003*, pages 640–651. ACM, 2003.

[21] Florian Kerschbaum. A Verifiable, Centralized, Coercion-free Reputation System. In *WPES 2009*, pages 61–70. ACM, 2009.

[22] Aggelos Kiayias and Moti Yung. Group Signatures: Provable Security, Efficient Constructions and Anonymity from Trapdoor-Holders, 2004.

[23] Joseph K. Liu, Victor K. Wei, and Duncan S. Wong. Linkable Spontaneous Anonymous Group Signature for Ad Hoc Groups. In *ACISP 2004*, volume 3108 of *LNCS*, pages 325–335. Springer, 2004.

[24] Mark Manulis, Ahmad-Reza Sadeghi, and Jörg Schwenk. Linkable Democratic Group Signatures. In *ISPEC 2006*, volume 3903 of *LNCS*, pages 187–201. Springer, 2006.

[25] Antonis Michalas and Nikos Komninos. The Lord of the Sense: A Privacy Preserving Reputation System for Participatory Sensing Applications. In *Computers and Communication*, volume 23 of *ISCC*, pages 1–6. IEEE, 2014.

[26] Toru Nakanishi and Nobuo Funabiki. A Short Verifier-Local Revocation Group Signature Scheme with Backward Unlinkability. In *Advances in Information and Computer Security*, volume 4266 of *LNCS*, pages 17–32. Springer, 2006.

[27] David Pointcheval and Jacques Stern. Security Arguments for Digital Signatures and Blind Signatures. *Journal of Cryptology*, 13:361–396, 2000.

[28] Sandra Steinbrecher. Design Options for Privacy-Respecting Reputation Systems within Centralised Internet Communities. In *Security and Privacy in Dynamic Environments*, volume 201 of *IFIP*, pages 123–134. Springer, 2006.

[29] Patrick P. Tsang and Victor K. Wei. Short Linkable Ring Signatures for E-voting, E-cash and Attestation. In *IPSEC 2005*, volume 3439 of *LNCS*, pages 48–60. Springer, 2005.