Reputation Systems for Anonymous Networks

Elli Androulaki, Seung Geol Choi, Steven M. Bellovin, and Tal Malkin

Department of Computer Science, Columbia University
{elli,sgchoi,smb,tal}@cs.columbia.edu

Abstract. We present a reputation scheme for a pseudonymous peer-to-peer (P2P) system in an anonymous network. Misbehavior is one of the biggest problems in pseudonymous P2P systems, where there is little incentive for proper behavior. In our scheme, using ecash for reputation points, the reputation of each user is closely related to his real identity rather than to his current pseudonym. Thus, our scheme allows an honest user to switch to a new pseudonym keeping his good reputation, while hindering a malicious user from erasing his trail of evil deeds with a new pseudonym.

1 Introduction

Pseudonymous System. Anonymity is a desirable attribute to users (or peers) who participate in peer-to-peer (P2P) system. A peer, representing himself via a pseudonym, is free from the burden of revealing his real identity when carrying out transactions with others. He can make his transactions unlinkable (i.e., hard to tell whether they come from the same peer) by using a different pseudonym in each transaction. Complete anonymity, however, is not desirable for the good of the whole community in the system: an honest peer has no choice but to suffer from repeated misbehaviors (e.g. sending an infected file to others) of a malicious peer, which lead to no consequences in this perfectly pseudonymous world.

Reputation System. We present a reputation system as a reasonable solution to the above problem. In our system, two peers, after carrying out a transaction, evaluate each other by giving (or not) a *reputation point*. Reputation points assigned to each peer sum up to create that peer's reputation value. In addition, reputation values are public, which helps peers to decide whether it is safe or not to interact with a particular peer (more exactly a pseudonym).

Identity Bound Reputation System. We stress that, in our system, the reputation value is bound to each peer. In existing reputation systems [KP03,KTR05], the reputation value is bound to each pseudonym. Consequently, a new pseudonym of a malicious peer will have a neutral reputation, irrespective of his past evil deeds. Thus, honest peers may still suffer from future misbehavior. On the other side, honest users won't use a new pseudonym, in order to keep the reputation

they have accumulated. Thus, they cannot fully enjoy anonymity and unlinkability. Motivated by this discussion, our goal in this paper is to design an identity bound reputation system, combining the advantages of anonymity and reputation.

Our Contribution. First, we formally define security for identity bound reputation systems (Section 3). As far as we are aware, this is the first such security definition. Our definition captures the following informal requirements:

- Each peer has a reputation which he cannot lie about or shed. In particular, though each peer generates as many one time pseudonyms as he needs for his transactions, all of them must share the same reputation. Also, our system is robust against a peer's deliberate attempts to increase his own reputation.
- Reputation are updated and demonstrated in a way that does not compromise anonymity. In particular, the system maintains unlinkability between the identity of a peer and his pseudonyms and unlinkability among pseudonyms of the same peer.

Our second contribution is the construction of a reputation scheme that satisfies the security definition. It is a nontrivial task to realize a secure identity bound reputation scheme, as the requirements of anonymity and reputation maintenance are (seemingly) conflicting. Here, we only briefly give basic ideas for the construction (see Section 2 for high level description of our scheme and Section 5 for the detail). To satisfy the first item, we need a central entity, *Bank*. Bank, aware of the identity of each peer, keeps reputation accounts by the peer, and is considered trusted to perform its functional operations — reputation updates etc. — correctly. Since we do not consider Bank trusted in terms of the anonymity requirements, we need to utilize a two-stage reputation deposit procedure. For the second item, we use the concept of *e-cash*. E-cash is well-suited to our system since it can be spent anonymously, even to Bank. We also use other primitives, such as anonymous credential system and blind signatures.

Organization. In Section 2 we provide a high level description of our scheme. In Section 3 we present our model, including security requirements. The building blocks used by our system are described in Section 4, followed by a detailed description of our system in Section 5. Related work and future directions are discussed in Sections 6 and 7 respectively.

2 System Considerations and High Level Approach

In this section we discuss system considerations and present a high level description of our scheme.

System Considerations and Assumptions. We assume that all communication takes place over an anonymous communication network, e.g., a Mixnet [C81] or an Onion Router [SGR97,DMS04]. We further assume that this network is, in fact, secure. While we are not minimizing the difficulty of achieving that — see, for example, [KDA+06] or [ØS06] — we regard that problem as out of scope for this paper.

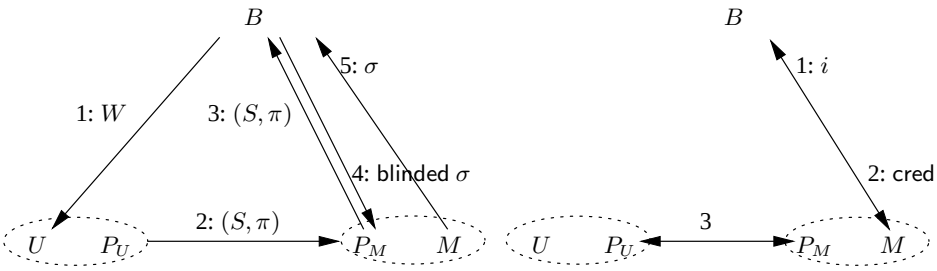
We also assume certain out-of-band properties that are necessary for correspondence to the real world. The most important such assumption is that there is some limit to the number of reputation points any party can hand out per unit time. While we don't specify how this limit is set, we tentatively assume that it costs real money to obtain such points to hand out. This might, for example, be the daily membership fee for participation in the P2P network. Note that the assumption corresponds quite well to the best-known existing reputation system, Ebay. One can only dispense reputation points there after making a purchase; that in turn requires payment of a fee to the auction site. Bhattacharjee and Goel have derived a model for what this fee should be [BG05]; they call the necessary property "inflation resistance".

A last assumption is unbounded collusion. That is, any number of parties on this network may collude to break anonymity of some other party. We specifically include the bank in this assumption. We assume collusion because in most real environments, it is possible for one party to open multiple accounts on the system. It may cost more money, but it does achieve the goal. Since a bank employee can do the same, we assume that the bank is colluding, too, albeit perhaps in response to a court order. Even if we assume a foolproof system for restricting accounts to one per person, two or more people could communicate via a private back channel, thus effectively creating multiple accounts under control of a single entity.

On the other hand, the bank is trusted to behave honestly in its functional transactions, which involve maintenance of reputation levels and repcoins for each peer (see below). Thus, if the bank is misbehaving (possibly in coalition with other adversarial users), it can compromise the correctness of the system, but not the anonymity. It is possible to distribute the bank functionality among several parties in order to increase fault tolerance and reduce any trust assumptions, but we will not describe this here.

Protocol Overview. Bank keeps the record of each peer's reputation in the *reputation database*. As shown on the left of Figure 1, a peer U (via his pseudonym P_U) can increase the reputation of a pseudonym P_M by giving a *repcoin*,[1] which

[1] If M wants to increase of reputation of P_U, they can carry out the same protocol with their roles reversed.



- Reputation granting process (left): (1) U withdraws a wallet W (i.e., repcoins) from the Bank B. (2) U, via P_U, awards (i.e., spends) a repcoin (S, π) to M. (3) M, via P_M, deposits the repcoin (S, π). (4) If the deposit is successful, P_M obtains from B a blind permission σ. Note that σ is blind to B and only visible to M. (5) M deposits σ, and B increases M's reputation point.
- Reputation demonstration process (right): (1) M requests a credential for the group G_i. (2) If M has enough reputation count for G_i, B issues a credential cred to M. (3) By using cred, P_M proves its membership of G_i to P_U .

Fig. 1. Reputation granting and demonstration

is basically an e-coin. Bank manages the number of repcoins that each peer has using another database: *repcoin quota database*.

Note that M does not deposit the repcoin using his identity. This is for the sake of maintaining unlinkability between a pseudonym and a peer. If M directly deposited the repcoin, collusion of Bank and U would reveal that M and P_M are linked. In fact, this shows the difficulty of realizing a secure reputation scheme: it is not obtained by using an ecash scheme naively. To preserve unlinkability, we use a level of indirection. When P_M successfully deposits the repcoin, it gets a blind permission from Bank. The blind permission is basically a blind signature, which therefore does not contain any information about P_M. So, M can safely deposit the permission.

We chose to employ an anonymous credential system (see Section 4) to construct the reputation demonstration procedure (on the right side of Figure 1). The anonymous credential enables M, via his pseudonym P_M, to prove his membership in group G_i anonymously. Thus, unlinkability between M and P_M is maintained.

We also note that P_M, instead of revealing its exact reputation value, shows the membership of a group G_i. Demonstration of exact reputation value could allow an attacker who continuously queries for the reputation of many pseudonyms — without even needing to transact with them — to infer whether two pseudonyms correspond to the same user. To make matters worse, with Bank's collaboration, pseudonyms can be linked to a limited number of identities that have the exact same reputation value with the pseudonym. On the other hand, group-

ing together identities which belong to the same reputation level, makes small changes in reputation accounts invisible to other pseudonyms. Bank can still see the changes that take place in peers' reputations, but cannot link them to specific pseudonyms any more. The reputation levels (i.e., groups G_i) are defined as a system parameter. Reputation levels are not necessarily required to be disjoint. One example would be that G_i contains peers who has more than 2^i different reputation values.

Punishing Misbehaving Parties. When modeling the security of the system, we aim to achieve our goals (such as anonymity, no lying about reputation level, no over-awarding reputations beyond the allowed limit, etc) by rendering a break of the security computationally infeasible (modulo some standard cryptographic assumptions). However, some security breaches are impossible to completely prevent. For example, as long as there is no central party involved on-line in each transaction, a user can always award the same reppoint twice to different parties. As another example, if anonymity and unlinkability is to be preserved, a peer with a high reputation level can always give away all his data and secret keys to another peer, allowing the latter to claim and prove the high reputation as his own. In these cases, we build into our model an incentive structure (similar to previous work, e.g., [LRSW99]), whereby such security breaches would hurt the offender. In particular, for the first case above, we require that a double awarding of a reppoint would reveal the identity of the offender (which can then lead to consequences outside of our model). For the second case, we require that in order for Alice to empower Bob, who has a lower reputation level, to prove a reputation level as high as Alice's, Alice would have to effectively give Bob her master private key. This information may be quite sensitive, especially if the private key used within the reputation system is the same one used for a public-key infrastructure outside the system.

3 A Model for Anonymous Reputation Systems

In this section, we present our model for anonymous reputation systems. We first enumerate the types of entities and the operations considered in the system, followed by the security definition. The motivation and rationale for our model and choices were discussed in Section 2. We note that some of these definitions were inspired by previous work on other primitives, such as [CL01,CHL05].

3.1 Participating Entities

The entities in an anonymous reputation system are as follows.

- **Peers.** Peers are the regular users of a P2P network. A peer interacts with other peers via pseudonyms of his choice and can be either a User (buyer) or a Merchant in different transactions. Peers can award reputation points to other peers (through their pseudonyms), and can show their reputation level to other peers.
- **Bank**. Bank manages information with respect to each peer's reputation (where the information is tied to actual identities — public keys — of peers, not to pseudonyms). Specifically, it maintains three databases: the repcoin quota database (denoted D_{quota}), the reputation database (denoted D_{rep}), and the history database (denoted D_{hist}).

  D_{quota} holds the amount of repcoins that each peer is allowed to award to other peers. When a peer withdraws a wallet of repcoins, the amount of his repcoin quota is decreased correspondingly. Bank also replenishes all the peer's account periodically, as per system parameters (for example, every day each peer can award at most 20 repcoins to others; see the discussion in Section 2). D_{rep} contains the amount of reputation points that each peer has earned by receiving repcoins from other peers. In order to prevent peers from double-awarding (awarding two peers with same-serial-numbered repcoins), D_{hist} holds all the repcoins that are deposited.

3.2 Operations

The operations supported in our system are listed below. When an operation is an interactive procedure (or a protocol consisting of multiple procedures) between two entities A and B, we denote it by $\langle O_A, O_B\rangle \leftarrow \mathsf{Pro}(I_C)[A(I_A), B(I_B)]$, where Pro is the name of the procedure (or protocol). O_A (resp. O_B) is the private output of A (resp. B), I_C is the common input of both entities, and I_A (resp. I_B) is the private input of A (resp. B). We also note that depending on the setup, some operations may require additional global parameters (e.g., some common parameters for efficient zero-knowledge proofs, a modulus p, etc). Our system will need these additional parameters only when using underlying schemes that use such parameters, e.g., e-cash systems or anonymous credential systems. To simplify notation, we omit these potential global parameters from the inputs to all the operations.

- $(pk_B, sk_B) \leftarrow \mathsf{Bkeygen}(1^k)$ is the key generation algorithm for Bank.
- $(pk_U, sk_U) \leftarrow \mathsf{Ukeygen}(1^k)$ is the key generation algorithm for peers. We call pk_U the (master) public key of U, and sk_U the master secret key of U.
- $(P, si_P) \leftarrow \mathsf{Pnymgen}(1^k)$ is the pseudonym generation algorithm for peers. The si_P is the secret information used to generate the pseudonym P.
- $\langle W, D'_{\mathsf{quota}}\rangle / \langle \perp, \perp\rangle \leftarrow \mathsf{RepCoinWithdraw}\ (pk_B, pk_U, n)\ [U(sk_U), B(sk_B, D_{\mathsf{quota}})]$. A peer U tries to withdraw n repcoins (in the form of a wallet W) from Bank B. Bank, using D_{quota}, checks if U is eligible for withdrawal. If so, the withdrawal is carried out and D_{quota} is changed accordingly.

- $\langle (W', S, \pi), (S, \pi) \rangle / \langle \perp, \perp \rangle \leftarrow$ Award (P_U, P_M, pk_B) $[U(si_{P_U}, W, pk_U, sk_U), M(si_{P_M})]$. A peer U (via P_U), using his wallet W, gives a repcoin (S, π) to M (via P_M). Here S is a serial number and π is the proof of a valid repcoin.
- $\langle \top, (D'_{\mathsf{rep}}, D'_{\mathsf{hist}}) \rangle / \langle \perp, \perp \rangle \leftarrow$ RepCoinDeposit (pk_B, S, π) $[M(\ P_U,\ si_{P_U},\ pk_U,\ sk_U\ ),\ B(\ sk_B,\ D_{\mathsf{rep}},\ D_{\mathsf{hist}}\ )]$. A peer M deposits the repcoin into his reputation account. If the repcoin (S, π) is valid and not double-awarded, then the coin is stored in the history database D_{hist}, and the amount of reputation of pk_M in D_{rep} is increased by one.
- $(pk_U, \Pi_G) / \perp \leftarrow$ Identify(S, π_1, π_2). If a repcoin is double-awarded with (S, π_1) and (S, π_2), Bank can find the peer who double-awarded the coin using this operation. Here, Π_G is a proof that pk_U double-awarded the repcoin with the serial number S.
- $\top / \perp \leftarrow$ VerifyGuilt(S, Π_G, pk_U) outputs $\top$ if the peer U (represented by pk_U) indeed double-awarded the coin with the serial number S.
- $\langle C_U^l, \top \rangle / \langle \perp, \perp \rangle \leftarrow$ RepCredRequest (pk_B, pk_U, l) $[U(sk_U), B(sk_B, D_{\mathsf{rep}})]$. A peer U requests a credential that will enable U to prove to another peer that he has reputation level l. Bank B refers to D_{rep}, and if U has sufficient reputation it issues a credential C_U^l. (As discussed in Section 2, how exactly the reputation levels are defined is a system parameter).
- $\langle \top, \top \rangle / \langle \perp, \perp \rangle \leftarrow$ ShowReputation $(P_{U_1}, P_{U_2}, pk_B, l)$ $[U_1(sk_{U_1}, si_{P_{U_1}}, C_{U_1}^l), U_2(si_{P_{U_2}})]$. A peer U_1 (via P_{U_1}) proves to U_2 (via P_{U_2}) that he has reputation level l.

3.3 Security

In this section we define security for anonymous reputation systems.

Adversarial Model. We will consider two adversarial models, assuming the stronger one for the anonymity-related security properties (unlinkability and exculpability), and the weaker one for the reputation-handling properties (no over-awarding and reputation unforgeability).

For the weaker adversarial model, we assume Bank is *honest-but-curious*, that is, it follows the protocol specification correctly. All other peers may become malicious, and behave in arbitrary ways in the protocol. Adversarial parties may collude with each other, and as long as they are peers, they may decide to share any of their state or secret information with each other, and coordinate their actions; Bank may share the content of its maintained databases ($D_{\mathsf{quota}}, D_{\mathsf{rep}}$, and D_{hist}), but *not* Bank's secret keys (thus it is meaningful for Bank to be *honest-but-curious*, even when in coalition with other players).[2]

For the stronger adversarial model, we remove the honest-but-curious restriction on Bank: we assume all parties (including Bank) may be corrupted, collaborating with each other, and behaving arbitrarily.

Correctness.

- If an honest peer U_1, who has enough repcoins in his repcoin quota, runs RepCoinWithdraw with an honest Bank B, then neither will output an error message; if the peer U_1, using the wallet (output of RepCoinWithdraw), runs

[2] Note that if we allowed Bank to share its secret keys and to behave arbitrarily, it could issue more repcoins than allowed, generate reputation credentials that do not correspond to the correct reputation level, etc.

Award with an honest peer U_2 (via his pseudonym), then U_2 accepts a repcoin (S, π); if the peer U_2 runs RepCoinDeposit with the honest Bank to deposit the repcoin (S, π) then U_2's reputation in Bank will be increased by one.

- If an honest peer U_1 runs RepCredRequest with an honest Bank and a reputation level for which he is eligible, then U_1 gets a valid credential. For a valid credential C_U^l, its owner can always prove his reputation through ShowReputation$(l, C_U^l, \ldots)$ procedure.

Unlinkability.

For an adversary A who has corrupted certain parties including Bank, we say that a peer U appears consistent with a pseudonym P to A, if U and P's owner are uncorrupted, and if the levels for which P successfully invoked ShowReputation are a subset of the levels for which U successfully invoked RepCredRequest. We now define the following two unlinkability properties:

Peer-Pseudonym Unlinkability. Consider an adversary who, having corrupted some parties including Bank, is participating in the system for some arbitrary sequence of operations executed by honest and by corrupted parties. Given a pseudonym P that does not belong to a corrupted party, the adversary can learn which peer owns P no better than guessing at random among all non-corrupted peers that appear consistent with P.

Pseudonym-Pseudonym Unlinkability. Consider an adversary who, having corrupted some peers (but not Bank), is participating in the system for some arbitrary sequence of operations executed by honest and corrupted parties. Given two pseudonyms P_1, P_2 that do not belong to corrupted parties, the adversary has no advantage in telling whether P_1, P_2 belong to the same peer or not. Next, consider an adversary who corrupted some peers and Bank as well. Then the above requirement should hold as long as there are *at least two* non-corrupted peers who appear consistent with both P_1 and P_2 (because if there is only one such uncorrupted peer, clearly both pseudonyms belong to the same one).

No Over-Awarding.

- No collection of peers should be able to award more repcoins than they withdrew. Suppose that n peers $U_1, \ldots, U_n$ collude together, and that the sum of the amount of repcoins allowed to them is N. Then, the number of different serial numbers of repcoins that can be awarded to other peers is at most N.
- Suppose that one or more colluding peers run the Award protocol with two pseudonyms P_{M_1} and P_{M_2} such that P_{M_1} gets (S, π_1) and P_{M_2} gets (S, π_2). Then, we require that Identify(S, π_1, π_2) outputs a public key pk_U and a proof of guilt Π_G such that VerifyGuilt(pk_U, S, Π_G) accepts.
- Each repcoin that is accepted but not double-awarded in the Award protocol increases exactly one reputation point in the database D_{rep} irrespective of the beneficiary of the repcoin. However, we don't regard it as a breach of security

when a peer M_1 received a repcoin but passed it to M_2, who deposited it into his reputation account; in any event, this is just another form of collusion. Another justification is that the peer M_1 sacrifices one reputation point.

Exculpability.
This property is to protect the honest peer from any kind of framing attack against him. No coalition of peers, even with Bank, can forge a proof Π_G that $\mathsf{VerifyGuilt}(pk_U, S, \Pi_G)$ accepts where pk_U is an honest peer U's public key who did not double-award a repcoin with the serial number S.

Reputation Unforgeability.

- No coalition of peers, where l is the highest reputation level of any one of them, can show a reputation level higher than l for any of their pseudonyms. This implies as a special case that a single peer cannot forge his reputation.
- Consider a peer U with reputation level l, who owns a pseudonym P. Suppose that some coalition of peers has empowered U with the ability to prove that P has reputation level $l' > l$. Let Bad be the set of peers with reputation level at least l' among the coalition (note that by the previous requirement, there must be at least one peer in Bad). Then, it must be that U can learn the master secret key of a peer $U' \in \mathsf{Bad}$.

4 Building Blocks of our Scheme

Anonymous Credential Systems. In anonymous credential systems — see, for example, [LRSW99,CL01,BCKL07] — there are three types of players: *users, organizations, and verifiers*. Users receive credentials, organizations grant and verify the credentials of users, and verifiers verify credentials of the users. Below are the supported procedures.

- $(pk_O, sk_O) \leftarrow \mathsf{AC.OKeyGen}(1^k)$. Key generation algorithm for an organization. (pk_O, sk_O) denotes the key pair of the organization O.
- $(pk_U, sk_U) \leftarrow \mathsf{AC.UKeyGen}(1^k)$. Key generation algorithm for a user. (pk_U, sk_U) denotes the key par of the user U. Sometimes sk_U is called the master secret key of U.[3]
- $\langle (N, \mathsf{NSecr}_N), (N, \mathsf{NLog}_N) \rangle \leftarrow \mathsf{AC.FormNym}(pk_O)\ [U(sk_U),\ O(sk_O)]$. Nym [4] generation protocol between U and O, where N is output nym, NSecr_N is secret information with respect to N, and NLog_N is the corresponding log on the organization side.

[3] Anonymous credential systems do not typically require a specific form for the master public and secret keys, but assume it is inherited from some PKI, where users are motivated to keep their secret key secret. In other variations of anonymous credential systems (with all-or-nothing non-transferability) there is no master public key. Our scheme can be adapted to such systems as well.

[4] Usually, nym and pseudonym are used interchangeably. But to avoid confusion with the term pseudonym in our reputation scheme, we stick to the term nym in anonymous credential systems.

- $\langle \mathsf{cred}_N, \mathsf{CLog}_{\mathsf{cred}_N} \rangle \leftarrow \mathsf{AC.GrantCred}(N, pk_O)\ [U(pk_U, sk_U, \mathsf{NSecr}_N),\ O(sk_O, \mathsf{NLog}_N)]$. Credential granting protocol, where cred_N is a credential for the nym N, and $\mathsf{CLog}_{\mathsf{cred}_N}$ is the corresponding log on the organization side.
- $\langle \top, \top \rangle / \langle \bot, \bot \rangle \leftarrow \mathsf{AC.VerifyCred}(pk_O)\ [U(N, \mathsf{cred}_N), V]$. Credential verification protocol.
- $\langle \top, \top \rangle / \langle \bot, \bot \rangle \leftarrow \mathsf{AC.VerifyCredOnNym}\ (N, pk_O, pk_{O_1})\ [U(N_1, \mathsf{cred}_{N_1}), O(\mathsf{NLog}_N)]$. In this protocol, U proves to O that N is his valid nym issued by O and that cred_{N_1} on the nym N_1 issued by O_1.

Secure anonymous credential systems satisfy the following conditions (see [LRSW99,CL01,BCKL07] for more details): (1) *Unique User for Each Nym.* Even though the identity of a user who owns a nym must remain unknown, the owner should be unique. (2) *Unlinkability of Nyms.* Nyms of a user are not linkable at any time with a probability better than random guessing. (3) *Unforgeability of Credentials.* A credential may not be issued to a user without the organization's cooperation. (4) *Consistency of Credentials.* It is not possible for different users to team up and show some of their credentials to an organization and obtain a credential for one of them that the user alone would not have gotten. (5) *Non-Transferability.* Whenever Alice discloses some information that allows Bob to user her credentials or nyms, she is effectively disclosing her master secret key to him.

E-Cash. An e-cash system consists of three types of players: the *bank*, *users* and *merchants*. Below are the supported procedures (see [CHL05]).

- $(pk_B, sk_B) \leftarrow \mathsf{EC.BKeyGen}(1^k)$ is the key generation algorithm for the bank.
- $(pk_U, sk_U) \leftarrow \mathsf{EC.UKeyGen}(1^k)$ is the key generation algorithm for users.
- $\langle W, \top \rangle \leftarrow \mathsf{EC.Withdraw}(pk_B, pk_U, n)\ [U(sk_U), B(sk_B)]$. The user U withdraws a wallet W of n coins from the bank.
- $\langle W', (S, \pi) \rangle \leftarrow \mathsf{EC.Spend}(pk_M, pk_B, n)\ [U(W), M(sk_M)]$. The user U spends a coin by giving it to the merchant M. U gets the updated wallet W, and M obtains a coin (S, π) where S is a serial number and π is a proof.
- $\langle \top / \bot, L' \rangle \leftarrow \mathsf{EC.Deposit}(pk_M, pk_B)\ [M(sk_M, S, \pi),\ B(sk_B, L)]$. M deposits (S, π) into its account in the bank B. L' is the updated list of the spent coins (i.e., (S, π) is added to the list).
- $(pk_U, \Pi_G) \leftarrow \mathsf{EC.Identify}(S, \pi_1, \pi_2)$. Given two coins with the same serial number, i.e., (S, π_1) and (S, π_2), B finds the identity of the double-spender pk_U and the corresponding proof Π_G.
- $\top / \bot \leftarrow \mathsf{EC.VerifyGuilt}(S, pk_U, \Pi_G)$. It verifies the proof Π_G that the user pk_U is guilty of double-spending coin S.

Secure e-cash scheme satisfies the following condition: (1) *Correctness.* If an honest user runs EC.Withdraw with an honest bank, then neither will output an error message. If an honest user runs EC.Spend with an honest merchant, then the merchant accepts the coin. (2) *Balance.* No collection of users and merchants can ever spend more coins than they withdrew. (3) *Identification of double-spenders.* Suppose the bank B is honest, and M_1 and M_2 are honest merchants who ran the EC.Spend protocol with the adversary whose public key

is pk_U. Suppose the outputs of M_1 and M_2 are (S, π_1) and (S, π_2) respectively. This property guarantees that, with high probability, $\mathsf{EC.Identify}(S, \pi_1, \pi_2)$ outputs a key pk_U and proof Π_G such that $\mathsf{EC.VerifyGuilt}(S, pk_U, \Pi_G)$ accepts. (4) *Anonymity of users.* The bank, even when cooperating with any collection of malicious users and merchants, cannot learn anything about a user's spendings other than what is available from side information from the environment. (5) *Exculpability.* When S is a coin serial number not double-spent by user U with public key pk_U, the probability that $\mathsf{EC.VerifyGuilt}(S, \Pi_G, pk_U, n)$ accepts is negligible.

Blind Signatures. Blind signatures have two types of players: the *bank* and the *users*. A user requests the bank to generate a signature on a message m. Then the bank generates a signature without knowing the message m. Below are the supported procedures (see [JLO97]).

- $(pk_B, sk_B) \leftarrow \mathsf{BS.KeyGen}(1^k)$. Key-generation algorithm for the bank B.
- $\langle \top/\bot,\ \sigma/\bot \rangle \leftarrow \mathsf{BS.Sign}(pk_B)[B(sk_B),\ U(m)]$. Signing protocol.
- $\top/\bot \leftarrow \mathsf{BS.Verify}(m, \sigma, pk_B)$. Verification algorithm.

Secure blind signature scheme satisfies the following conditions: (1) *Unforgeability.* Only the bank who owns the secret key sk_B can generate valid signatures. (2) *Blindness.* The bank B does not learn any information about the message m on which it generates a signature σ.

5 Anonymous Identity-Bound Reputation System

In this section we describe a general scheme based on any implementation of the building blocks. See Appendix A for a specific instantiation of the scheme.

E-cash schemes will be used for the implementation of repcoins, blind signatures will be used in repcoin-withdraw and reputation-update procedures, and anonymous credential systems will be used for the reputation-demonstration procedures. As we shall see, while the first two are used in a relatively straightforward manner, the last one is used in a more complex way, since the reputation demonstration setting presents a new type of hurdle to overcome if unlinkability is to be achieved even against colluding bank and peers.

Underlying Protocols and Requirements. Our scheme will work with any implementation of these underlying primitives, as long as the master public and secret keys for peers in our system are of the same form as those in the underlying e-cash scheme and anonymous credential system. That is, the key generation algorithms $\mathsf{Ukeygen}$, $\mathsf{EC.UKeyGen}$, and $\mathsf{AC.Ukeygen}$ are all the same.[5]

[5] As discussed in Section 2, an important part our system setup is the assumption that peers are motivated to keep their master private key secret. For this reason, it is beneficial to have the master public and private keys be part of an external PKI which is used for other purposes (e.g., signing documents) outside our system.

Our scheme will also require a zero knowledge proof of knowledge of both the master secret key corresponding to a master public key, and the secret information of a nym's owner (which is given as an output of the AC.FormNym operation). Thus, when instantiating our scheme with specific primitives, it is useful to choose underlying primitives that admit efficient proofs of this form (as we do in the Appendix A).

Setup. We start with the setup procedure on Bank's side.

- Bank B executes EC.BKeyGen procedure of e-cash scheme to create a digital signature key-pair (pk_B, sk_B). This is the key-pair that will be used for creating the repcoins. Bank publishes pk_B.
- B executes BS.BkeyGen procedure of blind signatures scheme to create a blind signature key pair to be used in the Reputation Deposit procedure (pk_B^b, sk_B^b). Bank publishes pk_B^b.
- B defines fixed reputation levels l_i, represented by a group G_i. These "reputation" groups — although managed by Bank — play a role similar to the one organizations play in anonymous credential systems. For each one of these groups, Bank runs AC.OKeyGen protocol to generate public-secret key pairs (pk_{G_i},sk_{G_i}). Bank also publishes pk_{G_i}s.
- B does the appropriate setup (if any) for the pseudonym generation. For example, this may involve selecting an appropriate algebraic group G_p.

On the peers' side, each peer U_i invokes EC.UKeyGen to create a master public-secret keypair (pk_{U_i}, sk_{U_i}).

Operations. As mentioned, we assume that messages are exchanged through perfectly secure channels. The system operations are realized as follows.

1. Generation of Pseudonyms. Each peer generates his own pseudonyms. There is no particular structure imposed on the pseudonyms, and they need not be certified or registered with Bank (or any other entity). The only requirement is that the pseudonym generation leaves the owner with some secret information (e.g., the random string used for the generation procedure), such that possession of this information proves ownership of the pseudonym. We will also need such a proof to be executed. Thus, in principle, we can simply use a random string r as the secret information and $P = f(r)$ as the pseudonym, where f is some one-way function, with an associated zero-knowledge proof of knowledge of the inverse of P. However, a more efficient solution is to let the pseudonym generation procedure to be a digital signature key generation, keeping the signing key as the secret information and the verification key as the pseudonym. Here, being able to produce valid signatures will prove ownership of the pseudonym, without a need for a zero-knowledge proof.

2. RepCoin Withdrawal. RepCoin Withdrawal takes place between Bank B and a peer U. Both U and B engage in EC.Withdraw procedure of a e-cash scheme. For simplicity purposes, we assume that a wallet W of n repcoins has been withdrawn. Since the only properties related to repcoins are anonymity of an honest withdrawer and repudiation of any double spender, the wallet can be like the one suggested in [CHL05], or n separate digital coins withdrawn through any known e-cash scheme.

3. Reputation Award. This procedure is executed between two pseudonyms, one (i.e., P_U) belonging to a peer U and one (i.e., P_M) belonging to a peer M. Both engage in EC.Spend protocol of a e-cash scheme. However, this protocol takes place strictly between the two pseudonyms P_U and P_M instead of involving the actual identities U and M. Thus, P_U gives a repcoin to P_M, where no information about identities of the parties involved is revealed.

4. Reputation Update. This protocol is invoked when a peer M wants to increase his reputation based on the repcoins that his pseudonyms have received since the last time he updated his reputation record. As previously discussed, maintaining unlinkability between a pseudonym and its owner is a crucial feature of our system. Towards this end, a single interaction for update (with a merchant presenting himself to Bank either as a peer or as a pseudonym) will not work, as we explain below.

Assume peer M wants to deposit a repcoin he received as P_M from pseudonym P_U of User U. Note that no one except M knows who is the owner of P_M. Given the fact that U knows the exact form of the repcoin he gave to M, if M tried to deposit the repcoin by presenting himself as M to Bank, a collusion of Bank and U would reveal that M is the owner of P_M. Trying to solve this by letting M "rerandomize" the repcoin in some way before depositing it presents problems for enforcing the no over-awarding requirement. On the other hand, if Reputation Update procedure was done by the pseudonym P_M of M, there would be a problem in persuading the Bank to update M's record without revealing that M is the owner of P_M.

Therefore, our Reputation Update protocol has two stages. First, P_M contacts Bank and gets a blind permission from it that shows a repcoin has been deposited and is valid. Second, M deposits that blind permission. In particular, the following procedure takes place:

4.1 Obtaining Blind Permission. Peer M executes EC.Deposit procedure of e-cash scheme using his pseudonym P_M, but here the actual deposit does not happen. Rather, if Bank B accepts the repcoin, M gets from B a blind signature on a random message. That is, P_M sends to B a repcoin that it has

received. If B accepts the coin as valid, P_M chooses a random message C and gets a blind signature of C: σ_B^b. We call (C, σ_B^b) a *blind permission*.
4.2 Deposit of the Blind Permission. M sends B the permission (C, σ_B^b). Then, B checks if the tuple is fresh and increases the reputation of M.

5. Reputation Demonstration. This protocol is invoked when one peer wants to demonstrate his reputation to another peer, both interacting strictly through their pseudonyms. We will utilize predefined groups G_i corresponding to reputation levels l_i, which are managed by Bank. For a peer U who wants, via P_U, to prove his reputation level l_i to a pseudonym P_V of a peer-verifier V, the protocol proceeds as follows:

- If he has not done it before, U contacts the bank to register in the group G_i that corresponds to the desired reputation level l_i. U interacts with G_i (Bank) by invoking AC.FormNym protocol of a anonymous credential system, in order to generate a nym $N_U^{l_i}$ for U under that group.[6] (U can generate as many nyms as he wants.)
- U contacts G_i, providing its master public pk_U key and a zero knowledge proof of knowledge π that he possesses the corresponding master secret key sk_U. U also presents $N_U^{l_i}$ and a zero-knowledge proof π_N that it has been created correctly and he is the owner.
- G_i checks that U is valid and that his reputation is indeed in that group (or higher), and executes AC.GrantCred to generate a credential $C_N^{l_i}$ for $N_U^{l_i}$.
- U interacts with the verifier P_V under his pseudonym P_U. P_U proves by executing AC.VerifyCred that he possesses a credential from group G_i. Specifically, P_U proves that its owner has registered under a nym to G_i and has acquired — through that nym — a credential of membership.

5.1 Security

The following theorem states the correctness and security of our general scheme. For lack of space, we refer the reader to our technical report [ACBM07] for proofs.

Theorem 1. *If the underlying primitives (anonymous credential system, e-cash system, and blind signatures) are secure, then our scheme satisfies* correctness, peer-pseudonym unlinkability, pseudonym-pseudonym unlinkability, no over-awarding, exculpability, *and* reputation unforgeability.

[6] Recall that there is a big difference between pseudonyms and nyms. As discussed before, Pseudonyms are public-secret key-pairs, used as means to preserve peers' anonymity when involved in transactions. A nym of a peer will be associated with a particular reputation group. Bank, as the manager of the reputation groups, will be able to link the nyms with the peer identities (master public key). In contrast, unlinkability of peers and pseudonyms is maintained, as per our security definitions.

6 Related Work

A number of papers have addressed the issue of reputation and privacy.

There are many papers on reputation systems for peer-to-peer networks. Most focus on building distributed reputation systems, rather than worrying about privacy; [GJA03] is typical.

The difficulty of building systems like this is outlined by Dingledine, Mathewson, and Syverson [DMS03]. They present a number of similar systems and show why bolting on reputation is hard.

A typical approach is typified by [VHM05], who incorporate privacy into their scheme. However, their system does not provide unlinkability. It also requires a trusted "observer" module for full functionality.

The work closest to ours is by Kinateder et al. [KP03,KTR05]. The system in [KP03] differs from ours in two notable ways. First, its reputations are linkable. Indeed, they see this as a virtue, in that recommendations can be weighted depending on the reputation of the recommender. Second, they assume a trusted hardware module (i.e., a TPM chip) on every endpoint.

In [KTR05], they describe a more general system based on UniTEC [KR03]. Reputation statements are signed by a pseudonym's private key. Unlinkability is achieved by switching public keys. Apparently, the UniTEC layer can share reputations between different pseudonyms, but the authors do not explain how this is done. Presumably, this is handled by bookkeeping at that layer. More seriously, although they assert that a trusted module is desirable but not necessary, they do not explain how that could work, and in particular how they can prevent cheating.

The most interesting system in the context of our paper, is that of Pavlov et al. [PRT04]. Their system, based on secret-sharing, has many of the same properties as ours. However, it depends on locating "witnesses", other parties with knowledge of the target's reputation. In a sufficiently-large community with a low density of interaction, this may be difficult. Furthermore, it does not provide unlinkability; witness testify about a known party's past behavior.

7 Future Directions

A few interesting open problems remain.

First, our current scheme uses unit coins for reputation. That is, all reputation credits are worth the same amount. It would be nice to permit variable values; we suspect that this is easy.

More seriously, we do not have negative feedback. There is a vast difference between knowing that a seller has performed well on m transactions and knowing that that seller has performed well on m out of n. The difficulty is forcing the

seller to commit to depositing a coin indicating bad behavior; most sellers know when they have done something wrong. In the technical report [ACBM07], we developed a partial solution. The scheme does not satisfy the complete unlinkability requirement stipulated in our definition, as Bank knows the number of transactions a peer had interacted in as a seller (modulo this information being leaked, all anonymity requirements are preserved).

Finally, we would like to get rid of the bank, which in our scheme is trusted to maintain reputation balances correctly (though not trusted from the privacy perspective). A fully decentralized scheme would eliminate single points of failure, and would be more in keeping with a widespread, anonymous, peer-to-peer network. Note that this would require two significant changes: using a digital cash scheme that does not require a central bank, and devising some other mechanism for inflation resistance.

Acknowledgment

We are grateful to Moti Yung for useful discussions regarding this work

References

[ACBM07] E. Androulaki, S. G. Choi, S. M. Bellovin, and T. Malkin. Reputation systems for anonymous networks. Technical Report Technical Report CUCS-029-07, Dept. of Computer Science, Columbia University, 2007. http://www.cs.columbia.edu/research/publications.

[BCKL07] M. Belenkiy, M. Chase, M. Kohlweiss, and A. Lysyanskaya. Non-interactive anonymous credentials. Cryptology ePrint Archive, Report 2007/384, 2007. http://eprint.iacr.org/.

[BG05] R. Bhattacharjee and A. Goel. Avoiding ballot stuffing in ebay-like reputation systems. In *P2PECON '05: Proceeding of the 2005 ACM SIGCOMM workshop on Economics of peer-to-peer systems*, pages 133–137, New York, NY, USA, 2005. ACM Press.

[C81] D. L. Chaum. Untraceable Electronic Mail, Return Addresses, and Digital Psuedonyms. *Communications of the ACM*, 1981.

[CHL05] J. Camenisch, S. Hohenberger, and A. Lysyanskaya. Compact e-cash. In *Advances in Cryptology - EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 302–321. Springer-Verlag, 2005.

[CL01] J. Camenisch and A. Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In *Advances in Cryptology - EUROCRYPT 2001*, volume 2045 of *Lecture Notes in Computer Science*, pages 93–118. Springer-Verlag, 2001.

[CS97] J. Camenisch and M. Stadler. Effcient group signature schemes for large groups. In *Advances in Cryptology — CRYPTO '97*, volume 1296 of *Lecture Notes in Computer Science*, pages 410–424. Springer-Verlag, 1997.

[DMS03] R. Dingledine, N. Mathewson, and P. Syverson. Reputation in p2p anonymity systems. In *Workshop on Economics of Peer-to-Peer Systems*, 2003.

[DMS04] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The second-generation onion router. In *Proceedings of the 13th USENIX Security Symposium*, August 2004.

[GJA03] M. Gupta, P. Judge, and M. Ammar. A reputation system for peer-to-peer networks. In *NOSSDAV*, 2003.

[JLO97] A. Juels, M. Luby, and R. Ostrovsky. Security of blind digital signatures (extended abstract). In *Advances in Cryptology - CRYPTO '97*, volume 1294 of *Lecture Notes in Computer Science*, pages 150–164. Springer-Verlag, 1997.

[KDA+06] D. Kesdogan, Dakshi, Agrawal, V. Pham, and D. Rautenbach. Fundamental limits on the anonymity provided by the mix technique. In *Proceedings of the IEEE Symposium on Security and Privacy*, May 2006.

[KP03] M. Kinateder and S. Pearson. A Privacy-Enhanced Peer-to-Peer Reputation System. In K. Bauknecht, A. M. Tjoa, and G. Quirchmayr, editors, *Proceedings of the 4th International Conference on Electronic Commerce and Web Technologies (EC-Web 2003)*, volume 2738 of *LNCS*, pages 206–215, Prague, Czech Republic, September 2003. Springer-Verlag.

[KR03] M. Kinateder and K. Rothermel. Architecture and Algorithms for a Distributed Reputation System. In P. Nixon and S. Terzis, editors, *Proceedings of the First International Conference on Trust Management*, volume 2692 of *LNCS*, pages 1–16, Crete, Greece, May 2003. Springer-Verlag.

[KTR05] M. Kinateder, R. Terdic, and K. Rothermel. Strong pseudonymous communication for peer-to-peer reputation systems. In *SAC '05: Proceedings of the 2005 ACM symposium on Applied computing*, pages 1570–1576, New York, NY, USA, 2005. ACM Press.

[LRSW99] A. Lysyanskaya, R. Rivest, A. Sahai, and S. Wolf. Pseudonym systems. In *Selected Areas in Cryptography '99*, volume 1758 of *Lecture Notes in Computer Science*, pages 184–199. Springer-Verlag, 1999.

[O92] T. Okamoto. Provably secure and practical identification schemes and corresponding signature schemes. In *Advances in Cryptology - CRYPTO '92*, volume 740 of *Lecture Notes in Computer Science*, pages 31–53. Springer-Verlag, 1992.

[ØS06] L. Øverlier and P. Syverson. Locating hidden servers. In *Proceedings of the IEEE Symposium on Security and Privacy*, 2006.

[PRT04] E. Pavlov, J. S. Rosenschein, and Z. Topol. Supporting privacy in decentralized additive reputation systems. In *Second International Conference on Trust Management: iTrust*. Springer, 2004. LNCS 2995.

[SGR97] P. F. Syverson, D. M. Goldschlag, and M. G. Reed. Anonymous connections and onion routing. In *IEEE Symposium on Security and Privacy*, pages 44–54, Oakland, California, 4–7 1997.

[VHM05] M. Voss, A. Heinemann, and M. Muhlhauser. A privacy preserving reputation system for mobile information dissemination networks. In *SECURECOMM '05: Proceedings of the First International Conference on Security and Privacy for Emerging Areas in Communications Networks (SECURECOMM'05)*, pages 171–181, Washington, DC, USA, 2005. IEEE Computer Society.

A An Example of Scheme Instantiation

In this section we give a specific instantiation of our scheme, where we make use of the anonymous credential system by Camenisch and Lysyanskaya [CL01] (denoted by CL), the e-cash scheme by Camenisch et al. [CHL05] (denoted by

CHL), and the blind signature scheme by Okamoto [O92] (denoted by Ok). We do so in order to present a concrete and efficient construction (we include the efficiency analysis, relying on that of the underlying primitives, with each of the operations).

Setup(1^k).
Bank B does the setup as follows:

- B executes $\mathsf{CHL.BKeygen}(1^k)$ to generate an e-cash key pair (pk_B^{ec}, sk_B^{ec}), and publishes $pk_B^{\mathsf{ec}} = (g_{\mathsf{ec}}, \hat{g}_{\mathsf{ec}}, \tilde{g}_{\mathsf{ec}})$.
- B executes $\mathsf{Ok.KeyGen}(1^k)$ to generate a blind signature key pair (pk_B^{bs}, sk_B^{bs}) and publishes pk_B^{bs}.
- For each reputation group G_i ($1 \leq i \leq k$), B executes $\mathsf{CL.OKeyGen}(1^k)$ to generate the anonymous credential system key pair $(pk_B^{\mathsf{ac}_i}, sk_B^{\mathsf{ac}_i})$ for G_i, and publishes $pk_B^{\mathsf{ac}_i} = (n_{\mathsf{ac}_i}, a_{\mathsf{ac}_i}, b_{\mathsf{ac}_i}, d_{\mathsf{ac}_i}, g_{\mathsf{ac}_i}, h_{\mathsf{ac}_i})$.
- B creates a cyclic group $G_p = \langle g_p \rangle$ of order $p = \Theta(2^k)$ where the DDH assumption holds. This algebraic group is used for pseudonym generation on the peer's side.

On the peers' side, each peer U executes $\mathsf{CHL.UKeyGen}(1^k)$ to obtain (pk_U, sk_U) = ($g_{\mathsf{ec}}^{x_U}$, x_U), and publishes pk_U. Note that x_U will be used as the master secret key of U in the anonymous credential system (and this discrete-log based key is a reasonable choice for a more general PKI key as well).

Operations.

1. Generation of Pseudonyms. Each peer generates his pseudonyms locally using $\mathbb{G}_p$. Specifically, he chooses a random number $r_i \in \mathbb{Z}_p$ and compute $g_p^{r_i}$. The value $g_p^{r_i}$ is considered a pseudonym P_U^i of peer U.

2. RepCoin Withdrawal. A peer U executes $\mathsf{CHL.Withdraw}$ with Bank, and obtains a wallet W of 2^w repcoins. This procedure takes $\mathrm{O}(1)$ exponentiations and $\mathrm{O}(1)$ rounds.

3. Reputation Award. A pseudonym P_U gives a repcoin to P_M by executing $\mathsf{CHL.Spend}$ with P_M. This procedure also takes $\mathrm{O}(1)$ exponentiations and $\mathrm{O}(1)$ rounds.

4. Reputation Update.

4.1 Obtaining Blind Permission. A pseudonym P_M and Bank B participate in $\mathsf{CHL.Deposit}$ protocol, which takes $\mathrm{O}(1)$ exponentiations and $\mathrm{O}(1)$ rounds. If $\mathsf{CHL.Deposit}$ accepts, P_M acquires the blind permission $\sigma_B^{\mathsf{bs}} = \mathsf{Ok.Sign}(sk_B^{\mathsf{bs}}, r_{perm})$ where r_{perm} is a random message. Obtaining the blind permission takes $\mathrm{O}(1)$ exponentiations and $\mathrm{O}(1)$ rounds.

4.2 Deposit of the Blind Permission. M (the owner of P_M) sends σ_B^{bs} to B. B checks if the permission $(r_{perm}, \sigma_B^{\mathsf{bs}})$ is fresh; if so, it increases M's reputation value. This procedure takes $\mathrm{O}(1)$ exponentiations and $\mathrm{O}(1)$ rounds.

5. Reputation Demonstration. Suppose that a pseudonym P_U asks P_M to demonstrate its reputation level, and that M (the owner of P_M) wants to show to P_U that it belongs to G_i, i.e., his reputation is at least at level l_i.

- Obtaining a nym under G_i. M contacts Bank B and executes CL.FormNym with respect to G_i[7]. Let $N_M^{l_i}$ be the nym that M obtained from this procedure. Note that $N_M^{l_i}$ is of the form: $g_{\mathsf{ac}_i}^{x_U} \cdot h_{\mathsf{ac}_i}^{r}$. This takes $\mathrm{O}(1)$ exponentiations and $\mathrm{O}(1)$ rounds.
- Obtaining a credential for G_i. M contacts B, and he sends B the message $(pk_M, N_M^{l_i})$. Then, M executes with B a zero-knowledge proof of knowledge

$$PK\{(\alpha, \beta) : pk_M = g_{\mathsf{ec}}^{\alpha}, N_M^{l_i} = g_{\mathsf{ac}_i}^{\alpha} \cdot h_{\mathsf{h}_i}^{\beta}\}.^{8}$$

  This takes $\mathrm{O}(1)$ exponentiations and $\mathrm{O}(1)$ rounds.

  Now, B verifies the proof. If the proof is verified so that M is eligible for a credential of the group G_i, B executes the CL.GrantCred (protocol4) with respect to G_i. Let C_{l_i} be the output credential. This takes $\mathrm{O}(1)$ exponentiations and $\mathrm{O}(1)$ rounds.
- Showing reputation using the credential. P_M contacts P_U and executes CL.VerifyCred (protocol3) with respect to G_i to prove that owner of P_M has a credential for the group G_i. This takes $\mathrm{O}(1)$ exponentiations and $\mathrm{O}(1)$ rounds.

[7] We use both protocol1 and protocol6 of [CL01] instead of just protocol1 to ensure the non-transferability of credentials.

[8] This proof can be parsed as "I know the exponent α and β that was used in generating pk_M and $N_M^{l_i}$". See [CS97,CL01] for more detail. The proof can be regarded as an authentication procedure.