A Real World Identity Management System with Master Secret Revocation

Elli Androulaki, Binh Vo and Steven Bellovin
{elli, binh, smb}@cs.columbia.edu

Technical Report CUCS-008-10

Abstract. Cybersecurity mechanisms have become increasingly important as online and offline worlds converge. Strong authentication and accountability are key tools for dealing with online attacks, and we would like to realize them through a token-based, centralized identity management system. In this report, we present a privacy-preserving group of protocols comprising a unique per user digital identity card, with which its owner is able to authenticate himself, prove possession of attributes, register himself to multiple online organizations (anonymously or not) and provide proof of membership. Unlike existing credential-based identity management systems, this card is revocable, i.e., its legal owner may invalidate it if physically lost, and still recover its content and registrations into a new credential. This card will protect an honest individual's anonymity when applicable as well as ensure his activity is known only to appropriate users.

1 Introduction

Existing master secret based privacy preserving identity systems offer many advantages. Users in these systems may obtain anonymous certificates from a central certificate authority, use these certificates to register anonymously in various organizations and obtain pseudonymous membership credentials. The credentials issued are unforgeable and can provide conditional anonymity and user-activity untraceability that is revokable for misbehaving users. Concurrently, master secret being so important for their online activity, users are motivated not to share their master secret; thus, credentials cannot be used by others. Unfortunately, current anonymous credential systems fail to consider many "real world" issues and this is the theme of this report.

First of all, most credential systems do not have a scalable solution for *credential blacklistability*. This is important given the large number of identity theft cases in the real world. Another "real world" issue is *variety* in organization registration policies. For example, a bank will probably require a lot more information to register a user than an online subscription service. However, the most important deficiency of current credential systems is the the lack of a proper *secret information update* procedure. In "master-secret"-driven centralized systems, master secrets may be compromised or stolen and, currently, there is no efficient way for a user to transfer his old registrations and credentials to his new account, meaning that he has no advantage over an attacker who has compromised his old secret.

In this report we address all of these issues in an identity management architecture which will be the base of our system. In particular, we present a privacy preserving and master secret-based identity management system, where every individual can generate a single master secret, which he can

- use to prove identity multiple times unlinkably and anonymously,
- use to register to multiple organizations along with other credentials depending on each organization's policy

– update in a manner that recursively updates all generated sub-credentials
– recover, when lost or destroyed

We emphasize the fact that our system is deployable, in other words:

– we make "real world" assumptions on our adversary's powers,
– we consider "real world" settings for the various organizations
– we propose protocols that scale for a large group of participants

Organization. This report is organized as follows: in sections 2, 3.2 and 3.3 we present the architecture, threat model and requirements of our system; in section 4 we present our protocols, while in section 5 we elaborate on how privacy, security and security are incorporated in our protocols.

2 A centralized privacy-preserving architecture

As mentioned before, in our system individuals have a single identity, register with a public authority, and obtain government-issued credentials. Individuals may use these credentials to participate in a number of real-world *interactions*, which include, without being limited to, the most important id-based activities of an individual, such as handling of employment, management of bank accounts, verification of specific attributes of the individual (e.g. legal drinking age), and registration in multiple online or offline clubs, associations or services. As discussed in section **??**, our identity management system consists of the following entities:

– *Users*, who may interact with other users or organizations in order to perform various tasks. A single user should map directly to a citizen.
– The *Registration Authority* (RA), which is responsible for registering users and managing the construction, modification, and destruction of government-issued credentials. In keeping with the mapping between users and citizens, this will likely be realized by a country's official citizenship registry, for example the social security office.
– *Special financial organizations*, who include employers that form employment relationships with users and banks that allow users to open accounts for the purpose of storing cash and handling financial transactions.
– *Non-financial organizations*, who may wish to extend membership to users and are not responsible for tax reporting. Such organizations may include gym centers, schools, commercial websites, etc.

Currently in the real world, citizens collaborate with an authority similar to RA to issue a national identity card. This card provides a physical proof of identity which they use to open accounts in banks, to be employed and receive their payments, and to prove their age. However, users are not currently required to prove their national identity in online communications and, thus, cannot be punished for misbehaving in these systems. In order to deal with this, we propose an architecture, where each valid user interacts with RA to obtain credentials stored in an identity card IDC. Each user with an IDC can prove that he is associated with a valid national identity without revealing it. He can thus open bank accounts, register to many other online and offline organizations, while assuring them that he can be held accountable if he misbehaves according to their established policy. In the following section we will elaborate on the specifications of our system.

3 A model for Privacy Preserving Identity Management

To achieve the required *user-activity centralization* — imposed by our needs for accountability and strong authentication — in our system, each user U generates a unique master secret $\mathsf{ms_U}$, which will be accounted for when the former misbehaves.[1] Users must use their master secret to authenticate themselves towards multiple organizations and are thus highly motivated not to share it. Seemingly contradictory to accountability, privacy requires that IDC-ownership demonstration does not leak any information regarding its owner or link multiple interactions of the latter. To achieve the aforementioned properties we use [TAKS07] (see section 4.1) as a base for the user-membership in our system (RA-registration). The registration authority RA maintains two important databases: the $\mathsf{DB_{RA}}$, where the registered-user information is stored, and the $\mathsf{BL_{RA}}$, which serves card and user blacklistability purposes. In addition RA maintains $\mathsf{BL_{RA}^{hold}}$, a temporary list of the anonymous owners of accounts in debt.

Apart from the generation and validation of his $\mathsf{ms_U}$, each user U collaborates with the RA to issue three types of credentials, which are stored in U's IDC and can be used to demonstrate U-validity multiple times anonymously:

1. a registration credential $\mathsf{regtick}$, which authorizes U as a valid user of the system and serves user-blacklistability purposes,
2. a wallet of one-time-use credentials perm-credentials, which can be used for U to register to various organizations, and
3. a wallet of one-time-use cred-credentials, which can be used for U to register to organizations, who limit registrations to one per user, i.e., hospitals etc.

Every time U registers to an organization, he makes use of his $\mathsf{ms_U}$ to demonstrate knowledge of his $\mathsf{regtick}$, and to use one of his credentials. The transcripts of these demonstrations will be used as the local identity of U within the organization and serve blacklistability purposes when U misbehaves. Afterwards, U may have credentials generated within that organization using any known anonymous credential system depending on the type of the organization.

In addition to his registration credentials, U may apply for a number of attribute credentials from the RA, such as possession of driving license or of a car, adulthood, etc. and their time of validity may vary. After verifying U's real-world validity, RA issues blind attribute related permissions, att-credentials, bound to U's $\mathsf{ms_U}$, for the former to deposit to the corresponding attribute services. Ultimately, proof of possession of attribute att is realized through proof of U-membership to the corresponding (att) group. U contacts the attribute-agencies anonymously but using his $\mathsf{regtick}$ and att-credential and obtains membership to that group the same way he obtains membership to organizations.

To fully recover his $\mathsf{ms_U}$ and the content of his card, when lost or compromised, U participates in a recovery setup protocol which has two important phases. The first one is performed at the end of his RA-registration, where U shares his $\mathsf{ms_U}$ and a recovery secret key $\mathsf{dk_U}$ in a shared secret fashion; $\mathsf{dk_U}$ will serve U-authentication purposes. The second phase takes place at the end of each organization-registration procedure, when the organization U collaborates with uses a central database, $\mathsf{DB_{MS}}$, to store confidentially subscription information anonymously but authoritatively information. The latter is encrypted with the encryption key $\mathsf{ek_U}$ which corresponds to $\mathsf{dk_U}$. Second level authentication information stored in RA ($\mathsf{RegInfo_U}$) and the organizations ($\mathsf{AuthData}$) for each

[1] As we will see later on, in our system *misbehavior* is a concept, whose definition varies across different organizations.

member, authorize an honest user, who has lost his card and properly recovered his $\mathsf{dk_U}$ to order the blacklisting of his old $\mathsf{regtick}$, trace his subscriptions, prove ownership and blacklist all the credentials issued with his old master secret. An attacker in this case, would not have the appropriate information to succeed in this authentication and will thus be unable to make use of the card.

3.1 Operations

To define the operations of our system more strictly we will use the following notation: 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).

1. $(pk_{\mathsf{RA}}, sk_{\mathsf{RA}}) \leftarrow \mathsf{RAKeyGen}(1^k)$ is the key generation algorithm for the registration authority.
2. $(pk_{\mathsf{O}}, sk_{\mathsf{O}}) \leftarrow \mathsf{OKeyGen}(1^k)$ is the key generation algorithm for the an organization O.
3. $(pk_{\mathsf{U}}, sk_{\mathsf{U}}) \leftarrow \mathsf{UKeyGen}(1^k)$ is the key generation algorithm for users. We call pk_{U} the (master) public key of U, and sk_{U} the master secret key of U.
4. $\langle (si_{\mathsf{regtick}}, \mathsf{W_U^{att}}, \mathsf{W_U^{perm/cred}}), (\mathsf{regtick}, \mathsf{DB_{RA}}')\rangle / \langle \perp, \perp\rangle \leftarrow$

   $$\leftarrow \mathsf{RARegistration}(pk_{\mathsf{RA}})[\mathsf{U}(sk_{\mathsf{U}}), \mathsf{RA}(sk_{\mathsf{RA}}, \mathsf{DB_{RA}})]$$

   is the registration of a user to the registration authority RA. The common output of this procedure is U's public registration and recovery information $\mathsf{regtick}$ and $\mathsf{RegInfo_U}$ respectively. U's private output is the corresponding secret information si_{regtick}, and wallets of various credentials $\mathsf{W_U^{att}}, \mathsf{W_U^{perm/cred}}$. RA's private output is information related to U's (and credentials') blacklisting and U's activity tracing which are stored in $\mathsf{DB_{RA}}$.
5. $\langle si_{\mathsf{att}}^{\mathsf{U}}, \mathsf{CLog_{att}}\rangle \leftarrow \mathsf{GrantAtt}(\mathsf{att}, \mathsf{gpk_{att}})\ [\mathsf{U}(pk_{\mathsf{U}}, sk_{\mathsf{U}}),\ \mathsf{att}(\mathsf{gsk_{att}})]$. This interactive procedure between a user and an attribute service of RA and the user U to generate a certificate for ownership of attribute att. The output for the user is the secret information $si_{\mathsf{att}}^{\mathsf{U}}$ regarding the att-credential $\mathsf{cred_{att}}$. The output for the organization is some information $\mathsf{CLog_{att}}$ for the credential.
6. $\langle \top, \top\rangle / \langle \perp, \perp\rangle \leftarrow \mathsf{ShowAttribute}(pk_{\mathsf{V}}, pk_{\mathsf{RA}}, \mathsf{att})\ [\mathsf{U}(si_{\mathsf{att}}^{\mathsf{U}}, sk_{\mathsf{U}}), \mathsf{V}(si_{\mathsf{V}})]$. A user U shows possession of attribute att to a verifier V.
7. $\langle \top, \mathsf{SID}\rangle / \langle \perp, \perp\rangle \leftarrow \mathsf{UserAuthenticate}(\mathsf{BL})[\mathsf{U}(sk_{\mathsf{U}}, si_{\mathsf{regtick}})\mathsf{V}(sk_{\mathsf{V}})]$, where a user U proves to a verifier V that he is not among the users in the blacklist BL. The result for the V is the transcript of U-authentication SID.
8. $\langle (\mathsf{W_U}', \mathsf{S}, \pi), (\mathsf{S}, \pi)\rangle / \langle \perp, \perp\rangle \leftarrow \mathsf{CredDemonstrate}(pk_{\mathsf{RA}}pk_{\mathsf{V}})[\mathsf{U}(sk_{\mathsf{U}}\mathsf{W_U}), \mathsf{V}(sk_{\mathsf{V}})]$. A user U, using his wallet $\mathsf{W_U}$ of some type of registration credentials ($\mathsf{perm/cred}$ /rev), demonstrates ownership of one of them to a verifier V. Here S is the serial number connected to the credential and π is the proof of a valid demonstration of it.
9. $\langle (\mathsf{W_U^{perm/cred'}}, \mathsf{MemSec_U^O}), (\pi, \mathsf{MemPub_U^O}, \mathsf{SID}, \mathsf{AuthData}, \mathsf{RecData}, \mathsf{D_O}')\rangle / \langle \perp, \perp\rangle \leftarrow$

   $$\leftarrow \mathsf{OrgRegistration}(pk_{\mathsf{O}}, pk_{\mathsf{RA}}, \mathsf{BL_{RA}})[\mathsf{U}(sk_{\mathsf{U}}, si_{\mathsf{regtick}}, \mathsf{W_U^{perm/cred}}), \mathsf{O}(sk_{\mathsf{O}}, \mathsf{D_O})]$$

   is the registration of a user U to an organization O. U's private input consists of his master secret sk_{U}, the secret related to his RA-registration si_{regtick}, as well as the wallets of his credentials $\mathsf{W_U^{perm/cred}}$. His private output is secret information regarding his O-membership

$\mathsf{MemSec}_{\mathsf{U}}^{\mathsf{O}}$. The common output consists of U's membership public information $\mathsf{MemPub}_{\mathsf{U}}^{\mathsf{O}}$, and information to enable U to authenticate himself in case of IDC-loss AuthData. O obtains U-blacklisting information SID, credential validity information π and updates his members' database $\mathsf{D_O}$ with the U's membership data: $\mathsf{MemPub}_{\mathsf{U}}^{\mathsf{O}}$, π, SID, AuthData. Both parties also obtain obtain RecData, which will be anonymously uploaded to $\mathsf{DB_{MS}}$ by O.

10. $\langle(\mathsf{TempSec}, \mathsf{W}_{\mathsf{U}}^{\mathsf{rev}}, \mathsf{RegInfo_U}), (\mathsf{TempPub}, \mathsf{BL_{RA}}')\rangle/\langle\perp, \perp\rangle \leftarrow$

$$\leftarrow \mathsf{LossReport}(pk_{\mathsf{RA}}, pk_{\mathsf{U}}, \mathsf{regtick})[\mathsf{U}(sk_{\mathsf{U}}, si_{\mathsf{regtick}}), \mathsf{RA}(sk_{\mathsf{RA}}, \mathsf{BL_{RA}})],$$

where a user U reports the loss of his IDC, recovers his registration data $\mathsf{RegInfo_U}$, and blacklists his membership credential regtick. U obtains revocation credentials $\mathsf{W}_{\mathsf{U}}^{\mathsf{rev}}$ and a temporary RA-membership (TempPub, TempSec) to update his registrations.

11. $\langle\top, (\mathsf{BL_{RA}}')\rangle/\langle\perp, \perp\rangle \leftarrow \mathsf{RegBlackList}(pk_{\mathsf{RA}}, \mathsf{regtick})[\mathsf{O}(sk_{\mathsf{O}}, \mathsf{SID}, proof), \mathsf{RA}(sk_{\mathsf{RA}}, \mathsf{SID})]$, where an organization O blacklists a userU with session id SID, who has misbehaved. O provides the session id of the misbehaving party along with proof of his misbehavior.
12. $\langle\top, \mathsf{BL_{RA}}'\rangle/\langle\perp, \perp\rangle \leftarrow \mathsf{AttBlackList}(\mathsf{S}^{\mathsf{att}})[\mathsf{U}(sk_{\mathsf{U}}, \mathsf{TempSec}, \mathsf{W}_{\mathsf{U}}^{\mathsf{rev}}), \mathsf{RA}(sk_{\mathsf{RA}}, \mathsf{BL_{RA}})]$, where the user U contacts again the RA to blacklist all of his attribute-related credentials.
13. $\langle(\mathsf{O}, \mathsf{RecData}), \top\rangle/\langle\perp, \mathsf{BL_{RA}}'\rangle \leftarrow \mathsf{OrgRegistrationRecovery}$

$$(pk_{\mathsf{RA}})[\mathsf{U}(\mathsf{TempSec}, sk_{\mathsf{U}}, \mathsf{W}_{\mathsf{U}}^{\mathsf{rev}}, \mathsf{S}), \mathsf{DB_{MS}}(\mathsf{BL_{RA}}, \mathsf{RecData})]$$

In this procedure, the user U utilizes his revocation permissions and the recovered serial numbers to trace the organizations he has registered to. U uses a piece of information contained in RecData to authenticate himself and immediately advertise blacklisting information for O.

14. $\langle\mathsf{MemSec}_{\mathsf{U}}^{\mathsf{O}'}, (\mathsf{BL_O}', \mathsf{MemPub}_{\mathsf{U}}^{\mathsf{O}'})\rangle/\langle\perp, \perp\rangle \leftarrow \mathsf{OrgRegistrationUpdate}$

$$(pk_{\mathsf{RA}}, \mathsf{RecData})[\mathsf{U}(sk_{\mathsf{U}}, \mathsf{TempSec}, \mathsf{W}_{\mathsf{U}}^{\mathsf{rev}}), \mathsf{O}(sk_{\mathsf{O}}, \mathsf{AuthData}, \mathsf{D_O})],$$

where the user U contacts the organization to change his credentials. U authenticates himself using RecData and the corresponding AuthData stored in $\mathsf{D_O}$. He then obtains new membership credentials.

15. $\langle\top, \mathsf{DB_{RA}}'\rangle/\langle\perp, \perp\rangle \leftarrow \mathsf{CredValidityCheck}(pk_{\mathsf{RA}}, \mathsf{S}, \pi)[\mathsf{E}(sk_{\mathsf{E}}), \mathsf{RA}(sk_{\mathsf{RA}}, \mathsf{DB_{RA}})]$. An entity E (which may be either a user or an organization) deposits the credential (S, π) to the RA, to check its validity. If the the credential (S, π) is valid and not double-used, then the credential is stored in the history database of $\mathsf{DB_{RA}}$.
16. $(pk_{\mathsf{U}}, \Pi_G)/\perp \leftarrow \mathsf{Identify}(\mathsf{S}, \pi_1, \pi_2)$. If a perm or cred credential is double-used with (S, π_1) and (S, π_2), the RA can find the person who double-used a credential using this operation. Here, Π_G is a proof that pk_{U} double-used the credential with the serial number S.
17. $\top/\perp \leftarrow \mathsf{VerifyGuilt}(\mathsf{S}, \Pi_G, pk_{\mathsf{U}})$ outputs $\top$ if the user U (represented by pk_{U}) indeed double-used the credential with the serial number S.

3.2 A "real world" threat model

The threat model of our identity management system has been presented in chapter **??** The assumptions we make regarding our adversaries are based on "real-world" settings and motives. In particular, we assume that:

– *Users may try to cheat.* A user may try to avoid paying for his purchases or bills or falsely claim he has not received a payment. In addition, he may try to impersonate other users to use their funds (impersonation attack) or frame other users for various misbehaviors.

– *(Non-)Financial Organizations are "honest but curious".* Aiming to maintain their clientele, banks, and other organizations, are trusted to perform all their functional operations correctly, i.e., they issue credentials, open and update accounts as instructed by their customers. However, they may attempt to learn more than is appropriate from their views of these transactions (this could be motivated by profiting from this information, i.e. selling behavior profiles to advertising companies). They may also try to collaborate with each other to improve their information advantage.
– *The Registration Authority is considered to be "honest but curious".* We assume that is operated by the government who wants to protect honest users, but it is not trusted to protect users' privacy.

3.3 Requirements

Privacy and *security* are our system's core requirements, which we adjust to the context of a centralized identity management system.

Privacy consists of *user anonymity* and *user activities' Unlinkability. User anonymity* requires that

1. given the transcript of the demonstration of ownership of a RA-membership credential that does not belong to a corrupted party SID, the adversary can learn which user owns SID no better than guessing at random among all non-corrupted users that appear consistent with SID.
2. given the transcript of proof of ownership of a perm/cred credential by a non-corrupted user, $\mathsf{CredTrans}(\pi, \mathsf{S})$ the adversary can learn which user owns CredTrans no better than guessing at random among all non-corrupted users.
3. given a public organization membership information $\mathsf{MemPub}_{\mathsf{U}}^{\mathsf{O}}$ that does not belong to a corrupted party, the adversary can learn which user owns $\mathsf{MemPub}_{\mathsf{U}}^{\mathsf{O}}$ no better than guessing at random among all non-corrupted users that appear in the system.
4. given the transcript of the demonstration of ownership of an attribute AttTrans, that does not belong to a corrupted party, the adversary may identify the user who owns AttTrans no better than guessing at random among all non-corrupted users who have been granted that attribute.
5. assuming that RA is not corrupted, given the transcript of the authentication procedure at the OrgRegistrationRecovery procedure of a non corrupted user, the adversary cannot learn which user participates in that procedure no better than guessing at random among all non-corrupted users; if we now include RA in the adversary, user anonymity is satisfied if the adversary cannot distinguish the user among all honest users who have run the LossReport procedure.
6. assuming that RA is not corrupted, given the transcript of OrgRegistrationUpdate operation of a non-corrupted user U with an organization O, the adversary cannot learn U's identity no better than guessing at random among all non-corrupted users; if we now include RA in the adversary, this user-anonymity is satisfied when the same holds among users who have run the LossReport procedure.

User activity unlinkability requires that activities of the same user cannot be linked as been committed by the same individual. In particular, we require:

– given the transcripts of two UserAuthenticate procedures SID_1 and SID_2, that do not belong to a corrupted party, the adversary has no advantage in telling whether SID_1 and SID_2 belong to the same user or not.

– given the transcripts of proof of ownership of two perm/cred /rev credentials $\mathsf{CredTrans}_1$ (π_1, S^1) and $\mathsf{CredTrans}_2$ (π_2, S^2) that do not belong to corrupted parties, the adversary has no advantage in telling whether $\mathsf{CredTrans}_1$ and $\mathsf{CredTrans}_2$ belong to the same user or not.
– given two memberships $\mathsf{MemPub}_{\mathsf{U}_1}^{\mathsf{O}_i/\mathsf{RA}}$ and $\mathsf{MemPub}_{\mathsf{U}_2}^{\mathsf{O}_j/\mathsf{RA}}$ to any organizations O_i, O_j or the RA, that do not belong to corrupted parties, the adversary has no advantage in telling whether $\mathsf{MemPub}_{\mathsf{U}_1}^{\mathsf{O}_i/\mathsf{RA}}$ $\mathsf{MemPub}_{\mathsf{U}_2}^{\mathsf{O}_i/\mathsf{RA}}$ belong to the same user or not.
– given the transcripts of two OrgRegistrationRecovery or OrgRegistrationUpdate operations, not performed by corrupted users, the adversary has no advantage in deciding whether they belong to the same user or not.

In any case, *privacy* is conditional on proper user behavior. On the other hand, *security*, is consists of:

- *Strong authentication*; no party can lie about his identity. It consists of *credential unforgeability*, *credential non-transferability* and *mis-authentication resistance*:
  – *Credential unforgeability* requires that no user or coalition of users may issue RA-membership credentials or any other type of credentials (perm/cred /rev) such that the UserAuthenticate and CredValidityCheck accept.
  – *Mis-authentication resistance* requires that the probability that UserAuthenticate accepts when a user does not have valid RA-membership is negligible, as well as the probability that UserAuthenticate rejects when the user has valid membership credentials.
  – IDC *and Credential non-transferability* implies that no one but the legal owner of the card may demonstrate ownership of the card or possession of the corresponding credentials. More specifically, assuming two users U_1 and U_2 and a credential cred_1 of U_1, we require that the probability that U_2 runs CredValidityCheck, CredDemonstrate or UserAuthenticate using cred_1 successfully is negligible.
- *Accountability*; misbehaving parties are punished. More specifically, we assume a misbehaving user U with an RA-membership regtick. Accountability requires that the probability that UserAuthenticate accepts is negligible.
- *Unframability - Denial of Service attack resistance*; no party should be able to frame another user for misbehavior or cause malfunction of his card's credentials. In particular, we require that no coalition of users, even with the collaboration of the RA, can forge a proof Π_G that $\mathsf{VerifyGuilt}(pk_{\mathsf{U}}, S, \Pi_G)$ accepts where pk_{U} is an honest user U's public key who did not double-used a credential with the serial number S.
- *Forward secrecy*, even if IDC is compromised, no entity or collaboration of entities, except for the IDC's owner can learn his memberships or activities.

Because our system is intended for real-world use, we also have the requirement of *Deployability*. This is partially expressed in our threat model and security requirements, which are made to reflect a real world environment. Furthermore, we require that our protocols scale for large systems.

4 A Centralized Privacy-Preserving Credential system

In this section we first describe the building blocks we made use of and we proceed with the details of our protocols.

4.1 Building Blocks

Electronic cash An E-Cash [CHL05,JCL06] system consists of three types of players: the *bank*, *users* and *merchants*. The input and output specifications of the basic operations are as follows. For convenience, we will assume that the operations take place between a merchant M, a user U and the Bank B.

- $(pk_{\mathsf{B}}, sk_{\mathsf{B}}) \leftarrow \mathsf{EC.BKeyGen}(1^k, params)$ and $(pk_{\mathsf{U}}, sk_{\mathsf{U}}) \leftarrow \mathsf{EC.UKeyGen}(1^k, params)$, which are the key generation algorithm for the bank and the users respectively.
- $\langle W, \top \rangle \leftarrow \mathsf{EC.Withdraw}(pk_{\mathsf{B}}, pk_{\mathsf{U}}, n)$ $[\mathsf{U}(sk_{\mathsf{U}}), \mathsf{B}(sk_{\mathsf{B}})]$. In this interactive procedure, U withdraws a wallet W of n coins from B.
- $\langle W', (S, \pi) \rangle \leftarrow \mathsf{EC.Spend}(pk_{\mathsf{M}}, pk_{\mathsf{B}}, n)$ $[\mathsf{U}(W), \mathsf{M}(sk_{\mathsf{M}})]$. In this interactive procedure, U spends a digital coin with serial S from his wallet W to M. When the procedure is over, W is reduced to W', M obtains as output a coin (S, π), where π is a proof of a valid coin with a serial number S.
- $\langle \top/\bot, L' \rangle \leftarrow \mathsf{EC.Deposit}(pk_{\mathsf{M}}, pk_{\mathsf{B}})$ $[\mathsf{M}(sk_{\mathsf{M}}, S, \pi),\ \mathsf{B}(sk_{\mathsf{B}}, L)]$. In this interactive procedure, M deposits a coin (S, π) into its account in the bank. If this procedure is successful, M's output will be $\top$ and the bank's list L of the spent coins will be updated to L'.
- $(pk_{\mathsf{U}}, \Pi_G) \leftarrow \mathsf{EC.Identify}(params, S, \pi_1, \pi_2)$. When the bank receives the two coins with the same serial number S and validity proofs π_1 and π_2, it executes this procedure, to reveal the public key of the violator accompanied with a violation proof Π_G.
- $\top/\bot \leftarrow \mathsf{EC.VerifyGuilt}(params, S, pk_{\mathsf{U}}, \Pi_G)$. This algorithm, given Π_G publicly verifies the violation of pk_{U}.
- $\{(S_i, \Pi_i)\}_i \leftarrow \mathsf{EC.Trace}(params, S, pk_{\mathsf{U}}, \Pi_G, D, n)$. This algorithm provides the list of serials S_i of the ecoins a violator pk_{U} has issued, with the corresponding ownership proofs Π_i.
- $\top/\bot \leftarrow \mathsf{EC.VerifyOwnership}(params, S, \Pi, pk_U, n)$. This algorithm allows to publicly verify the proof Π that a coin with serial number S belongs to a user with public key pk_U.

[JCL06] is a money-laundering prevention version of [CHL05], where anonymity is revoked when the spender spends more coins to the same merchant than a spending limit. In this case ecoins are upgraded to

$$C = (S, V, \pi),$$

where V is a merchant-related locator, while EC.Identify and EC.VerifyGuilt procedures are upgraded to the DetectViolator and VerifyViolation to support the extended violation definition.
Security Properties: (a) Correctness. (b) Balance. No collection of users and merchants can ever spend more coins than they withdrew. *(c) Identification of Violators.* Given a violation and the corresponding proofs of guilt, the violator's public pk_{U} key is revealed such that EC.VerifyViolation accepts. *(d) 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. *(e) Exculpability.* An honest user U cannot be accused of conducting a forgery attempting to fool EC.VerifyViolation. (f) Violators' Traceability. Given a violator Uwith a proof of violation Π_G, this property guarantees that EC.Trace will output the serial numbers of all coins that belong to U along with the corresponding proofs of ownership, such that for each one of them VerifyOwnership accepts.

Blacklistable Anonymous Credentials Blacklistable credential systems BAC serve user authentication purposes, i.e., provide the means for a user to prove that he is a member of a group multiple times anonymously and unlinkably, nevertheless accountably. A misbehaving person will not be able to use his credential any more. The entities in BAC [TAKS07] are the Group Manager GM, a set of service providers SPs (or verifiers) and users. The procedures supported are the following:

- $\langle \mathsf{gpk}, \mathsf{gsk} \rangle \leftarrow \mathsf{BAC.Setup}[\mathsf{GM}(1^k)]$. This algorithm generates a group public key gpk and the GM's secret group information gsk.

- $\langle \mathsf{cred}_{\mathsf{U}}, \mathsf{JLog}_{\mathsf{U}} \rangle \leftarrow \mathsf{BAC.Register}(\mathsf{gpk})[\mathsf{U}, \mathsf{GM}(\mathsf{gsk})]$. When this interactive registration ends, U has obtained his membership credential $\mathsf{cred}_{\mathsf{U}}$.
- $\langle \top/\bot \rangle \leftarrow \mathsf{BAC.Authenticate}(\mathsf{gpk})\ [\mathsf{U}(\mathsf{cred}_{\mathsf{U}}),\ \mathsf{SP}(\mathrm{BL})]$. In this interactive procedure, U proves to SP that he is a valid (non-blacklisted) member of the group.
- $\langle BL' \rangle \leftarrow \mathsf{BAC.BLAdd}[\mathsf{SP}(BL)]$, where a service provider ads a credential (ticket) to the blacklist BL.
- $\langle tick \rangle \mathsf{BAC.BLExtract}[\mathsf{SP}(BL)]$, where SP extracts an element from the blacklist.
- $\langle BL' \rangle \leftarrow \mathsf{BAC.BLRemove}[\mathsf{SP}(BL)]$, where SP removes a credential from the blacklist.

Security Properties: (a) Correctness. (b) Mis-authentication Resistance. No unregistered user or collection of unregistered users should be able to authenticate themselves. *(c) Blacklistability.* SPs may blacklist any misbehaving user of the system and restrict him from any ability of authenticating himself. *(d) Anonymity.* SPs may only learn whether a user is blacklisted or not; no identification information may be leaked. *(e) Non-framability.* An honest user should never be blocked from access.

Anonymous and Unlinkable Credential System, Pseudonym Systems Pseudonym systems have three types of players: *users, organizations, and verifiers.* Users are entities that receive credentials. Organizations are entities that grant and verify the credentials of users. Finally, verifiers are entities that verify credentials of the users. See [LRSW99,CL01] for more details. The standard operations supported are the following:

- $(pk_O, sk_O) \leftarrow \mathsf{PS.OKeyGen}(1^k)$. This procedure generates a public/secret key pair for an organization. We denote a key pair for an organization O by (pk_O, sk_O).
- $(pk_U, sk_U) \leftarrow \mathsf{PS.UKeyGen}(1^k)$. This procedure generates a public/secret key pair for a user. We denote a key pair for a user U by (pk_U, sk_U). Sometimes we refer to the secret key of a user as a master secret key for the user.
- $\langle (N, \mathsf{NSecr}_N), (N, \mathsf{NLog}_N) \rangle \leftarrow \mathsf{PS.FromNym}(pk_O)\,[U(pk_U, sk_U),\ O(sk_O)]$. This interactive procedure between a user and an organization generate a pseudonym (or simply nym). The common input is the public key of the organization O. The output for the user is a nym N and some secret information NSecr_N, and for the organization the nym N and some secret information NLog_N.
- $\langle \mathsf{cred}_N, \mathsf{CLog}_{\mathsf{cred}_N} \rangle \leftarrow \mathsf{PS.GrantCred}(N, pk_O)\ [U(pk_U, sk_U, \mathsf{NSecr}_N),\ O(sk_O, \mathsf{NLog}_N)]$. This interactive procedure between a user and an organization generate a credential for a nym N. The common input is N and pk_O. The output for the user is the credential cred_N for the nym N. The output for the organization is some secret information $\mathsf{CLog}_{\mathsf{cred}_N}$ for the credential.
- $\langle \top, \top \rangle / \langle \bot, \bot \rangle \leftarrow \mathsf{PS.VerifyCred}(pk_O)\ [U(N, \mathsf{cred}_N), V]$. In this interactive procedure between a user and a verifier, the user proves that he has a credential on the nym N issued by the organization O.
- $\langle \top, \top \rangle / \langle \bot, \bot \rangle \leftarrow \mathsf{PS.VerifyCredOnNym}\ (N, pk_O, pk_{O_1})\ [U(N_1, \mathsf{cred}_{N_1}), O(\mathsf{NLog}_N)]$. In this interactive procedure between a user and the organization O, the user proves that N is his valid nym of the organization O and that he has a credential cred_{N_1} on the nym N_1 issued by the organization O_1.

Security Properties. (a) Unique User for Each Pseudonym. Even though the identity of a user who owns a nym must remain unknown, the owner should be unique. *(b) Unlinkability of Pseudonyms.* Nyms of a user are not linkable at any time better than by random guessing. *(c) Unforgeability of Credentials.* A credential may not be issued to a user without the organization's cooperation. *(d) 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 could not have gotten. *(e) Non-Transferability.* Whenever Alice discloses some information that allows Bob to use her credentials or nyms, she is effectively disclosing her master secret key to him.

Group Signature Scheme In a typical GSS [NF06,CKL05,KY05,BSZ05,CS97] with member revocability, there is a group manager (GM), the group-members, who act as signers (let each be S) and produce signatures on behalf of the group. If necessary, the group manager can remove members from the group. The procedures supported are the following:

- $(\mathsf{gpk}, \mathsf{gsk}) \leftarrow \mathsf{GS.Setup}(1^k)$. This algorithm generates a group public key $\mathsf{gpk}$and the GM's secret group information gsk.
- $\langle \mathsf{bgusk_S}, \mathsf{JLog_S} \rangle \leftarrow \mathsf{GS.Join}(\mathsf{gpk})[\mathsf{S}, \mathsf{GM}(\mathsf{gsk})]$. When this interactive join procedure ends, an S obtains a secret signing key $\mathsf{bgusk_S}$, and the GM (group manager) logs the join transcript in the database D.
- $\sigma \leftarrow \mathsf{GS.Sign}(\mathsf{gpk}, \mathsf{bgusk_S}, m)$. This algorithm generates a group signature on a message m.
- $\langle \top/\bot \rangle \leftarrow \mathsf{GS.Verify}(\mathsf{gpk}, m, \sigma)$. This is a verification algorithm.
- $\langle \mathsf{gpk}/\bot \rangle \leftarrow \mathsf{GS.MembershipRevoke}(\mathsf{gpk}, \mathsf{gsk}, \mathsf{JLog_S})$. In this operation the group manager GM removes the member with group-membership transcript $\mathsf{JLog_S}$.
- $\mathsf{ms} \leftarrow \mathsf{GS.Open}(\mathsf{gsk}, \sigma, D)$. With this algorithm the GM determines the identity of the group member who generated the signature σ.

Security Properties: *(a) Anonymity.* Given a signature and two members, one of whom is the originator, the adversary can identify its originator among the group members no better than randomly. *(b) Unforgeability.* The adversary or group of adversaries cannot produce a valid group signature without owning current group membership information. *(c) Non-framability.* The adversary cannot create a valid group signature that opens to another group member. *(d) Unlinkability.* Given two signatures, it is infeasible that anyone except the signers and the group manager to determine whether they were produced by the same person.

Blind Signature Scheme In a typical BSS, there are signers (let each be S) who produce blind signatures on messages of users (let each be U). The procedures supported are the following:

- $(pk_\mathsf{S}, sk_\mathsf{S}) \leftarrow \mathsf{BS.KeyGen}(1^k)$. This is a key-generation algorithm that outputs a public/secret key-pair $(pk_\mathsf{S}, sk_\mathsf{S})$.
- $\langle \top/\bot,\ \sigma/\bot \rangle \leftarrow \mathsf{BS.Sign}(pk_\mathsf{S})[\mathsf{S}(sk_\mathsf{S}),\ \mathsf{C}(m)]$. At the end of this interactive procedure, the output of the S is either *completed* or *not-completed* and the output of U is either the signature (σ) or a failure sign $(\bot)$.
- $\langle \top/\bot \rangle \leftarrow \mathsf{BS.Verify}(m, \sigma, pk_\mathsf{S})$ is a verification algorithm.

Security Properties: Apart from *Unforgeability, Blindness* is the most important security property of blind signature schemes: S does not learn any information about the message m on which it generates a signature σ.

4.2 The Protocols in Depth

Moving on to the details of our protocols, for the system **setup**, the RA generates through $\mathsf{PS.OKeyGen}$ a digital signature key-pair $(pk^s_\mathsf{RA}, sk^s_\mathsf{RA})$ to identify itself. RA also generate a blind signature key-pair $(pk^b_\mathsf{RA}, sk^b_\mathsf{RA}))$ and runs $\mathsf{EC.Setup}$ multiple times: once for each digital cash-based credential system (perm, cred, rev credentials) and once for each attribute supported. In addition, it runs $\mathsf{BAC.Setup}$ and for the setup of users' RA-registration credentials ($\mathsf{regtick}$).

All the organizations of the system generate a digital signature key-pair to be identified in the system. Let $(pk_\mathsf{O}, sk_\mathsf{O})$ denote the O-identification key-pair. Organizations' setup depends entirely on the level of accountability they want to enforce:

– Important Financial Organizations, i.e., banks, employers, require high level of both security and privacy. It is thus reasonable to assume that (if adopting privacy) they are implemented as the credential systems of [CL01,CL02a], which combines strong transaction privacy (complete transaction unlinkability of honest individuals) with strong accountability in case of misbehavior (recovery of misbehaving user's master secret). See section 4.1 for more details. Under this context, each organization O runs the $\mathsf{PS.OKeyGen}$ procedure to generate its identification key-pair $(sk_\mathsf{O}, pk_\mathsf{O})$.

– Non financial Organizations, i.e., online magazines, which do not require any level of security, user may register multiple times, but whose privilege of extending the prescription can be revoked if the user misbehaves, i.e., he uses bad language in a forum. In this case, the identity of the users is not required for their registration. Organizations of this type may be implemented as group signature systems with membership revocation enabled, and — thus — run $\mathsf{GS.Setup}$ to generate group administration information: $(\mathsf{gpk}^{\mathsf{O}}, \mathsf{gsk}^{\mathsf{O}})$.
– Other Financial Institutions, who do not involve large monetary amounts and require a reasonable degree of accountability, i.e, any type of online service providers, may choose any of the aforementioned systems: blacklistable anonymous credentials, pseudonymous systems or group signature schemes.

In what follows we will use the following notation:

– $\mathsf{Sig}_{\mathsf{E}}(\mathsf{M})$ ($\mathsf{Sig}^{\mathsf{H}}_{\mathsf{E}}(\mathsf{M})$) for the signature of entity E on M ($\mathsf{H}(\mathsf{M})$).
– $\mathsf{GSig}^{(\mathsf{H})}_{\mathsf{g},\mathsf{E}}(\mathsf{M})$ for the g-group-signature of entity E on M ($\mathsf{H}(\mathsf{M})$).
– $\{\mathsf{M}\}_K$ for the encryption of M under key K. For efficiency, we induct every asymmetric encryption a symmetric one. Therefore, $\{\mathsf{M}\}_{PK}$ denotes $\{K\}_{PK}||\{\mathsf{M}\}_K$ for a random K.

RA **Registration** This procedure takes place between the RA and the user U, who requests to enter the system. After providing strong identification credentials, i.e., birth certificates, passports, etc., U runs EC.UKeyGen or PS.UKeyGen to generate his master secret ms_{U} and engages with the RA to the following series of interactions:

1. ms_{U}-*Validation.* U $\leftrightarrow$ RA: run PS.FormNym and PS.GrantCred procedure, to generate a pseudonym and a credential for U which is blindly linked to ms_{U}. The RA stores in $\mathsf{DB}_{\mathsf{RA}}$ all the public ms_{U}-information, $\mathsf{pub}_{\mathsf{U}}$.
2. *Registration Credential's Issue.* U $\leftrightarrow$ RA: run BAC.Register, to grant U a registration credential, regtick, which will serve blacklistability purposes.
3. *Credentials' Issue.* U $\leftrightarrow$ RA: run EC.Withdraw operation twice to generate wallets of perm-credentials and cred-credentials. They both serve authentication purposes for user registrations to organizations: the perm-credentials are realized as accountable ecash [JCL06], where the constant N is set to two (see section 4.1) and aim for organizations where users may have at most one entry; the cred-credentials, are implemented as any ecash scheme and may be used for any entity or organization. The key attributes of these credentials are that they are non-transferable, enable anonymous user-authentication, while through their serials provide a degree of traceability to their owner. The latter is particularly useful when their owner loses his card and wants to trace his previous activities.
4. *Recovery Mechanism Setup*, where U creates and stores some credential and master secret recovery information:
   (a) U generates his recovery encryption key pair $(\mathsf{ek}_{\mathsf{U}}, \mathsf{dk}_{\mathsf{U}})$.
   (b) U encrypts the serials of both types of credentials into

$$\mathsf{RegInfo}_{\mathsf{U}} = \{\mathsf{ms}_{\mathsf{U}}, \text{perm-serials}, \text{cred-serials}, \text{att-serials}, \text{date}\}_{\mathsf{ek}_{\mathsf{U}}},$$

   where att-serials are related to attribute-credentials (see section 4.2).
   (c) U$\rightarrow$ RA: $\mathsf{RegInfo}_{\mathsf{U}}$; both entities agree on a hash H and exchange proofs of the final form of $\mathsf{RegInfo}_{\mathsf{U}}$: $\mathsf{Sig}^{H}_{\mathsf{RA}}(\mathsf{RegInfo}_{\mathsf{U}}||\text{date})$ and $\mathsf{Sig}^{H}_{\mathsf{U}}(\mathsf{RegInfo}_{\mathsf{U}}||\text{date})$.
   (d) U shares in a shared secret fashion[S79] his ms_{U} and dk_{U}.

Organization Registration This procedure takes place between an organization O_i and a user U, and allows U to obtain membership in O_i. Depending on its setting, O_i may restrict user-registrations to one per user. Thus, depending on the case, membership pre-requisites of O_i may include proof of perm or cred credentials, U's identity or ownership of attributes. We emphasize on the fact that as perm-credentials have the form of accountable ecash [JCL06] (see, 4.1) if more than two perm- credentials of U are used for the same organization-merchant, U's identity is revealed.

1. U *Authentication.* $\mathsf{U} \leftrightarrow \mathsf{O}_i$:
   (a) run BAC.Authenticate for U to prove that he is among the valid users of the system. Let SID be the transcript of this demonstration.
   (b) engage in an EC.Spend procedure for U to bind one credential of his to O_i. As mentioned before, if there is a restriction regarding how many registrations a user should maintain in O_i, a perm-credential should be used; otherwise, a cred -credential may be used. Let $\mathsf{S}^{\mathsf{perm/cred}}_{\mathsf{U}\rightarrow\mathsf{O}_i}$ denote the credential's serial. It is apparent that if U uses the same credential twice or more than two perm-credentials for the same organization, his identity will be revealed.
2. *Actual Registration.* As mentioned before, the exact procedures that take place in this step depend on the type of the organization and on the level of accountability the latter wants to enforce. For strong accountable systems, U and O_i run PS.FormNym and PS.VerifyCredOnNym for U to create a pseudonym to O_i, which is blindly connected to his actual master secret. In the general case, U and O_i interact so that the former obtains his secret membership information $\mathsf{MemSec}^{\mathsf{O}_i}_{\mathsf{U}}$ and O_i the corresponding public information $\mathsf{MemPub}^{\mathsf{O}_i}_{\mathsf{U}}$. Henceforth, for simplicity, we assume that U is known to O_i as P_{U}.
3. *Second Level Authentication Mechanism.* In this phase, the user creates and stores information locally which will enable him to authenticate himself and manage his organization credentials in the case where he loses his card and all his organization membership information:
   (a) U creates an O_i-specific recovery encryption key pair: $(\mathsf{ek}^{\mathsf{O}_i}_{\mathsf{U}}, \mathsf{dk}^{\mathsf{O}_i}_{\mathsf{U}})$, computes
   $$\epsilon^{\mathsf{dk}}_{\mathsf{P}_{\mathsf{U}}} = \{\mathsf{dk}^{\mathsf{O}_i}_{\mathsf{U}}\}_{\mathsf{ek}_{\mathsf{U}}}.$$
   (b) Ugenerates secret and computes $H_{\mathsf{O}_i}(\mathsf{secret})$, and
   $$\epsilon^{\mathsf{sec}}_{\mathsf{P}_{\mathsf{U}}} = \{\mathsf{secret}\}_{\mathsf{ek}^{\mathsf{O}_i}_{\mathsf{U}}}, \quad \sigma^{\mathsf{sec}}_{\mathsf{P}_{\mathsf{U}}} = \mathsf{Sig}^{H\mathsf{O}_i}_{\mathsf{P}_{\mathsf{U}}}(\mathsf{secret}),$$
   where H_{O_i} is an O_i-specific hash.
   (c) $\mathsf{U} \rightarrow \mathsf{O}_i$: $\mathsf{AuthData} = \{\epsilon^{\mathsf{dk}}_{\mathsf{P}_{\mathsf{U}}}, \epsilon^{\mathsf{sec}}_{\mathsf{P}_{\mathsf{U}}}, \sigma^{\mathsf{sec}}_{\mathsf{P}_{\mathsf{U}}}\}$. U is the only one who knows secret. AuthData will be used to authenticate U, when the latter loses his credentials.

   Finally, O_i creates U's entry in his $\mathsf{DB}_{\mathsf{O}_i}$, where he stores the following:
   $$\mathsf{Entry}^{\mathsf{O}_i}_{\mathsf{U}} = \{\mathsf{S}^{\mathsf{perm/cred}}_{\mathsf{U}\rightarrow\mathsf{O}_i}, \mathsf{MemPub}^{\mathsf{O}_i}_{\mathsf{U}}, \mathsf{AuthData}\}.$$

Attributes Credentials Issue This is the procedure by which a user obtains attribute credentials (e.g., proof of age, medical status, marital status, etc.). We assume that for each possible attribute there is a separate service-group that U can visit right after his RA-registration. For each of these, RA runs GS.Setup during **setup**. After U proves to RA that he corresponds to $\mathsf{pub}_{\mathsf{U}}$ and that he is entitled of attribute att_i, the following take place:

1. RA$\leftrightarrow$ U: run EC.Withdraw for U to obtain one ecoin-token $\mathsf{AttTick}_i$, using the att_i-related ecash-setting. RA updates U's entry in $\mathsf{DB_{RA}}$ accordingly.
2. U$\leftrightarrow$ att_i-Service:
   (a) U-*Authentication.* U runs BAC.Authenticate to prove that he has registered to RA and that he is not among the blacklisted users in $\mathsf{BL_{RA}}$. Let AttTrans be the authentication transcript.
   (b) U-*Membership to* att_i-Service-*group.* U $\leftrightarrow$ att_i-Service: run GS.Join, for U to obtain membership to the corresponding group of attribute-possessors. Thus, U issues group membership key-pairs $(\mathsf{gsk}_{\mathsf{U}}^{\mathsf{att}_i}, \mathsf{gpk}_{\mathsf{U}}^{\mathsf{att}_i})$.
   (c) *Registration Recovery Mechanism.* Similar to the organization registration procedure, att_i-RecData is generated for U to be able to authenticate himself and invalidate his membership when his master secret is compromised.

At the end of this procedure, att_i-Service stores the U-attribute related info:

$$(\mathsf{AttTrans},\ \mathsf{AttTick}_i,\ \mathsf{gpk}_{\mathsf{U}}^{\mathsf{att}_i},\ \mathsf{att}_i\text{-RecData}).$$

Attributes Demonstration This is the procedure by which a user U proves ownership of an attribute att to a verifier V. V may be either a person or an organization.

1. Both parties participate to generate a challenge R.
2. V downloads the updated att-group public information: $\mathsf{gpk}^{\mathsf{att}}$.
3. U uses his $\mathsf{gsk}^{\mathsf{att}}$ membership (GS.Sign) and sign the dated challenge:

$$\sigma_{\mathsf{att}} = \mathsf{GSig}_{\mathsf{att,U}}(R \mid\mid \text{timestamp}).$$

4. V runs GS.Verify to validate the signature produced.

It is apparent that in case someone misbehaves, V may collaborate with the att-Service in GS.Open procedure to (locally) identify and remove the misbehaving group member from the group. In addition, att-Service, may blacklist that user's regtick using the AttTrans associated with the misbehaving $\mathsf{gpk}_{\mathsf{U}}^{\mathsf{att}_i}$.

Master Secret compromise - IDC Content Recovery This procedure has three phases. In the first phase, a user U contacts the RA to report the loss or compromise of his IDC. To authenticate himself, provides strong identification credentials, i.e., birth certificates, passports, etc. Then U using RA-issued permissions contacts an external database $\mathsf{DB_{MS}}$ to recover confidential information regarding his registrations, i.e., in which organizations he has registered to, so that he eventually contacts the latters to update his memberships. More specifically, the following take place:

1. U– RA*interaction.*
   (a) U $\rightarrow$ RA: authenticated report for IDC-loss or IDC-compromise.
   (b) U recovers his master secret $\mathsf{ms_U}$ and recovery $\mathsf{dk_U}$ using a shared secret recovery protocol[s79].
   (c) RAruns BAC.BLAdd to blacklist the old regtick. In this way, RA aims to prevent the attacker from registering using the card elsewhere.
   (d) RA($\mathsf{DB_{RA}}$) $\rightarrow$ U: $\mathsf{RegInfo_U}$; U then uses the recovered $\mathsf{dk_U}$ to recover the serials of his credentials (perm, cred and AttTick).
   (e) U $\leftrightarrow$ RA: run EC.Withdraw once more to issue one-time-use revocation credentials, rev-credentials. Attribute specific rev-credentials, are issued according to the information in U's entry in $\mathsf{DB_{RA}}$, for U to invalidate his old attribute credentials.

(f) $\mathsf{U} \leftrightarrow \mathsf{att}_i$-Service: U anonymously authenticates himself to att_i-Service (through att_i-specific rev-credentials) as the owner of a compromised master secret. After having recovered his $\mathsf{AttTick}_i$ serial number, U uses RecData, to authorize himself to order the blacklisting of the corresponding att_i-group membership information. Both parties collaborate so that U obtains a att_i-Service blind confirmation of their interaction, aimed for RA.

(g) $\mathsf{U} \leftrightarrow \mathsf{RA}$: run the RA Registration procedure again for U to generate a temporary ms_U and IDC. RA updates U's entry in DB_RA.

2. U– DB_MS *interaction.* In this procedure, U recovers the list of his registrations, i.e., which organizations he has registered with his old ms_U. This will be covered extensively in the following subsection.
3. U– O_i *interaction.* For each organization O_i, U has registered to, the following series of procedures take place:

(a) $\mathsf{U} \leftrightarrow \mathsf{O}_i$: run EC.Spend procedure, for U to bind one of his rev-credentials to O_i. It is important to note that because of their ecash nature, rev-credentials are also unlinkable to U's identity, nevertheless non-transferable. U shows the serial $\mathsf{S}^{\mathsf{perm/cred}}$, which he used to register to O_i.

(b) O_i: checks rev-credential's validity; if valid, O_i looks up $\mathsf{S}^{\mathsf{perm/cred}}$ up in its $\mathsf{DB}_{\mathsf{O}_i}$.

(c) $\mathsf{O}_i \rightarrow \mathsf{U}$: ϵ^{sec}.

(d) U: uses the recovered dk_U to decrypt his secret and demonstrates knowledge of it to O_i.

(e) $\mathsf{U} \leftrightarrow \mathsf{O}_i$: run BAC.Authenticate procedure for the temporary membership of U. The transcript of this interaction SID' will serve blacklistability purposes in cases U misbehaves. Note that the blacklist corresponding to the temporary master secrets is considerably smaller than the regular one.

(f) O_i blacklists all the credentials issued for the $\mathsf{MemPub}_\mathsf{U}^\mathsf{O}$ — using a technique similar to [CL02a,TAKS07] or [NF06], depending on its setting.

(g) U contacts O_i using his new credentials to open a new privacy preserving account.

4. U – RA *interaction.* U collaborates with RA in a BAC.Authenticate procedure for all the recently blacklisted items for both his old and temporary RA-registrations. Assuming that organizations report accounts that have not been accessed for a month, in this way, we want to avoid cases where the user falsely claimed loss of his IDC to erase accounts of his without being traced. After being cleared, both repeat a procedure similar to the RA registration for U to issue his new credentials.

User Registrations' Recovery In this stage, a user U who has already reported the loss of his IDC to the RA, recovers his registrations, i.e., information regarding which organizations he has become a member to using his old ms_U. The registrations' recovery procedure is subjected to many caveats. First of all, aiming to maintain U's privacy towards the RA, the serial numbers of the withdrawn credentials are only visible to U and — when eventually used — to the organization they are used for. Revealing the serials to the RA would enable the latter with the collaboration of all organizations to trace U's activities. However, there should be measures to prevent U from claiming ownership of false numbers and causing DoS to honest users who own these numbers. Therefore, we introduce an external database DB_MS used strictly by authorized organizations or users to upload registration related information. We assume that users maintain anonymous accounts with DB_MS and can read DB_MS-data from their accounts only through rev-credentials, while they The registrations' recovery protocol includes two phases, one which takes place after each registration,

and the actual recovery procedure. For the purposes of these protocols, we assume that there are two serial-number–specific hash functions $\mathsf{H}_{\mathsf{serial}}$ and $\mathsf{H}_{\mathsf{intserial}}$ and that the user chooses a OWF function F with id number FID from a public pool of OWFs. The series of actions are the following:

1. *Recovery Setup.* It takes place at the end of each user U- organization O_i registration. Let $\mathsf{S}^{\mathsf{perm/cred}}$ be the serial number of the credential U used (see subsection 4.2). The following take place:
   (a) $\mathsf{U} \leftrightarrow \mathsf{O}_i$: choose two hash functions F_1 and F_2 from the pool of hashes with id numbers $FID1$ and $FID2$ respectively.
   (b) $\mathsf{U} \rightarrow \mathsf{O}_i$:

$$\epsilon^{\mathsf{ser}} = \{\mathsf{S}^{\mathsf{perm/cred}}\}^{\mathsf{H}_{\mathsf{serial}}}_{\mathsf{K_U}},\ \epsilon^{\mathsf{ser},f} = \{\mathsf{H}_{\mathsf{intserial}}(\epsilon^{\mathsf{ser}}) \mid\mid FID,\ FID1,\ FID2\}_{\mathsf{K_U}},\ \text{and } FID,$$

   where $\mathsf{K_U}$ is a $\mathsf{ek_U}$-generated symmetric key.
   (c) O_i, U: compute $\epsilon^{\mathsf{org}}_{\mathsf{O}_i} = \{\mathsf{O}_i\}_{\mathsf{K}_f}$, where $\mathsf{K}_f = F(\mathsf{S}^{\mathsf{perm/cred}})$ is a $\mathsf{S}^{\mathsf{perm/cred}}$-generated key.
   (d) O anonymously uploads to $\mathsf{DB_{MS}}$:

$$\mathsf{RecData} = \epsilon^{\mathsf{ser}} \mid\mid \epsilon^{\mathsf{ser},f} \mid\mid \epsilon^{\mathsf{org}}_{\mathsf{O}_i}.$$

2. *Serial LookUp* phase, which takes place after the IDC loss declaration phase. In particular
   (a) $\mathsf{U} \leftrightarrow \mathsf{DB_{MS}}$: collaborate in $\mathsf{BAC.Authenticate}$ and $\mathsf{EC.Spend}$ procedures for U to prove that he is a valid user who has reported the loss of his IDC. U makes use of his temporary $\mathsf{ms_U}$ for these purposes. Let SID be the transcript of it.
   (b) For each serial $\mathsf{S}^{\mathsf{perm/cred}}$ U has recovered, he computes the corresponding ϵ^{ser}, which he then uses as a lookup key in $\mathsf{DB_{MS}}$.
   (c) U decrypts the corresponding $\epsilon^{\mathsf{org}}_{\mathsf{O}_i}$ to recover the name of the organization O_i related to that serial.$FID1, 2$ are used for avoiding DoS attacks by users who try to blacklist credentials they never owned.

U may occasionally decide to publish through $\mathsf{DB_{MS}}$ the $\mathsf{S}^{\mathsf{perm/cred}}$ he has recovered. In fact, $\mathsf{DB_{MS}}$ publishes with U's collaboration (ϵ^{ser}, $F_1(\mathsf{S}^{\mathsf{perm/cred}})||F_2(\mathsf{S}^{\mathsf{perm/cred}})$, SID) records. After its regular check O, freezes automatically the credentials that correspond to $\mathsf{S}^{\mathsf{perm/cred}}$'s registration. If U fails to contact O within a prefixed time, the freezing stops and SID is sent to RA, who depending on the type of O and its policy may blacklist U.

5 Discussion

In this section we will illustrate how privacy and security are achieved in our system, while we discuss deployability issues.

5.1 Privacy-Security

The following theorem states the correctness, privacy and security of our general scheme:

Theorem. if the underlying primitives (anonymous credential system, e-cash system, blind signatures) are secure, then our scheme satisfies *correctness*, *user anonymity*, *user activity unlinkability*,

credential unforgeability, *credential non transferability*, *mis-authentication resistance*, *user unframability*, *forward secrecy* and *accountability*.

We use prove this theorem with the following lemmas.

Lemma 1. If the underlying primitives (anonymous credential system, e-cash system, group signatures) are secure, then our scheme satisfies *Correctness*.

Lemma 2. If the underlying primitives (blacklistable anonymous credential system, ecash system) are secure, then our scheme satisfies *user-anonymity* and *user-activity unlinkability*.

Proof. Let that a user U has registered with the RA and obtained a registration membership $\mathsf{regtick}$ and wallets of $\mathsf{perm/cred}$ credentials $\mathsf{W}_{\mathsf{U}}^{\mathsf{perm/cred}}$ respectively. In addition, U has obtained a wallet of att-related credentials and has been granted attributes $\mathsf{att}_1, \mathsf{att}_2$. U has also registered to organizations O_1 and O_2, with memberships $\mathsf{MemPub}_{\mathsf{U}}^{\mathsf{O}_1}$ and $\mathsf{MemPub}_{\mathsf{U}}^{\mathsf{O}_2}$ respectively.

Anonymity property of the blacklistable anonymous credential and ecash scheme used guarantee that $\mathsf{regtick}$ ownership demonstration(SID) and the honest use of $\mathsf{W}_{\mathsf{U}}^{\mathsf{perm/cred}}$ will not reveal U's identity, which implies that $\mathsf{MemPub}_{\mathsf{U}}^{\mathsf{O}}$s are also unlinkable to U. In a similar way, the *anonymity* property of ecash schemes guarantee that U's identity is not known to att-service, while the same property of group signatures guarantees that demonstration of possession of any of $\mathsf{att}_1, \mathsf{att}_2$ ($\mathsf{Att\text{-}Trans}$) will not be linked to U's identity. *User Anonymity* in case of card loss is achieved through the blacklistable anonymous credential nature of the temporary registration credentials and the ecash nature of the revocation authorizations.

In a similar way, *unlinkability of user-activity* is satisfied through the unlinkability properties of the underlying ecash and blaclistable anonymous credentilas' schemes.

Lemma 3. If the underlying primitives (blacklistable anonymous credential system, ecash system) are secure, then our scheme satisfies *credential unforgeability*, *card non transferability* and *mis-authentication resistance*.

Proof. Credential unforgeability is directly satisfied through the *unforgeability* property of the underlying ecash and blacklistable anonymous credential system, while *mis-authentication resistance* is achieved through the *mis-authentication resistance* property of the blacklistable anonymous credentials, according to which the transcript of a demonstration of $\mathsf{regtick}$ ownership is enough to effectively blacklist a user. Non transferability property is implicitly achieved in our system. In particular, we adopt an "all or nothing" method according to which for a user U_1 to be able to lend his credentials to another user U_2, U_1 should reveal his master secret to U_2, which is a property supported by the underlying ecash schemes used.

Lemma 4. If the underlying primitives (blacklistable anonymous credential system, ecash system) are secure, then our scheme satisfies *accountability* and *user-unframability*.

Proof. It is directly achieved through the *Identification of Violators* and *exculpubility* properties of the anonymous ecash scheme, which guarantee that a user cannot be framed by any other party as, while if a user U uses the same credential twice or uses more that one perm credentials to the same organization, U's identity is revealed.

Lemma 5. If the underlying primitives (blacklistable anonymous credential system, ecash system) are secure, then our scheme satisfies *forward secrecy* and *resistance to DoS attacks.*
Sketch Proof. Accountability and Credential non- transferability provides a degree of protection against DoS attacks at the IDC-registration recovery phase. Users trying to blacklist credential serials of other users, will fail to identify the corresponding organization and FIDs; will thus be reported and blacklisted.

Forward secrecy in case of IDC-loss or compromise has been discussed in the previous section. Because of the *anonymity* property of ecash (rev-credentials) and anonymous credentials (TempPub), the RA, even when collaborating with organizations or $\mathsf{DB_{MS}}$, cannot link a particular serial to a user. FID-based user authentication for the temporary serial organization-membership blacklisting does not reveal also any information leakage regarding U.

5.2 Deployability

It consists of applicability and scalability. Regarding applicablity, as shown in previous sections, we have taken in consideration "real world" in our threat model and architecture. Scalability is achieved through (a) the tree-structure of the suggested credential system, and (b) the scalability of the underlying primitives. As demonstrated before, the credential-issuing procedure of each user may be parallelized with a tree, whose root is the user's regtick and whose leaves are attribute or organization credentials. Each organization maintains local blacklists according to its policies and notifies the RA-blacklist when necessary. In this way, user-authentication does not create bottlenecks: for regtick-authentication the user-perceived delay for a blacklist of 1600 entries is 4 seconds[TAKS07]. User-authentication against the global RA-blacklist, i.e., in cases when a user applies for a passport or a visa, does not create a bottleneck either since these procedures currently take much longer than a week. For attribute demonstration, [NF06,CKL05] group signature schemes offer the possibility to prove group-membership in $O(1)$ time regardless of the size of the group. These schemes also provide the possibility for the attribute service to activate a user's group-membership at a later time from his request without the need of that user to update his card. This is particularly useful in cases of age credentials. In the recovery protocol case, the detection of the credential serials and their blacklisting are done immediately through the secret sharing protocol and the use of the hashes at the $\mathsf{DB_{MS}}$.

6 Related Work

There has been some work indicating the problem of online privacy. Brands [B00] and Camenisch and Lysyanskaya [CL01] were the first to provide a big overview of privacy issues caused by the extended online use of PKI and provided a series of constructions of privacy preserving credentials, tickets and certificates based on blind signatures and zero knowledge proofs. There has been some work on blacklistable anonymous credentials [CL02b,TAKS07,AMO08]. Although the privacy provided in the aforementioned schemes is very strong, they do not refer to systems with multiple operations each requiring a different privacy level.

Centralized identity management systems applying the primitives of [B00,CL01] have been suggested in the past. In Idemix [CVH02] , Camenisch and Herreweghen developed additional functionality for service providers and credential issuers to configure and enforce resource access control and credential issuing decisions. Higgins [F], OpenID [FPVFOGTOIT] and iCard [JLLP03] Foundation are examples offrameworks handling many identities of the same user across different websites.

The PRIME project [G] is a European initiative for privacy preserving identity management for online commercial interactions. Although the existing work in the field refer to multiple types of user-interactions, they do not provide accountability when the user misbehaves, or consider real world issues deriving from master identity compromise, i.e., the complete recovery of the user's online subscriptions, automatic invalidation of the corresponding compromised credentials, advanced user-authentication to manage these operations etc.

7 Conclusion

In this report we presented a centralized, card-based identity management system which addresses many online and offline activities of individuals achieving different levels of privacy. It thus constitutes the core of the system presented in this dissertation. As opposed to most existing credential-based identity management systems, in our system the card is recoverable: when lost or compromised the card's legal owner may recover its content completely — his master identity and the subscriptions he has obtained through the latter — or even invalidate it. Including the cases of card loss or compromise, this card will protect an honest individuals anonymity when applicable as well as ensure his activity is known only to authorized users.

References

[AMO08] N. Akagi, Y. Manabe, and T. Okamoto. An efficient anonymous credential system. pages 272–286, 2008.

[B00] S. A. Brands. *Rethinking public key infrastructures and digital certificates: building in privacy.* Cambridge, Mass. : MIT Press, 2000.

[BSZ05] M. Bellare, H. Shi, and C. Zhang. Foundations of group signatures: The case of dynamic groups. In *Topics in Cryptology - CT-RSA 2005*, volume 3376 of *Lecture Notes in Computer Science*, pages 136–153. Springer-Verlag, 2005.

[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.

[CKL05] E. Y. Choi, H.-J. Kim, and D. H. Lee. Efficient member revocation in group signature schemes. In *TrustBus*, pages 195–205, 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.

[CL02a] J. Camenisch and A. Lysyanskaya. A signature scheme with efficient protocols. In *International Conference on Security in Communication Networks – SCN*, volume 2576 of *Lecture Notes in Computer Science*, pages 268–289. Springer Verlag, 2002.

[CL02b] J. Camenisch and A. Lysyanskaya. Dynamic accumulators and application to efficient revocation of anonymous credentials. In *CRYPTO '02: Proceedings of the 22nd Annual International Cryptology Conference on Advances in Cryptology*, pages 61–76, London, UK, 2002. Springer-Verlag.

[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.

[CVH02] J. Camenisch and E. Van Herreweghen. Design and implementation of the idemix anonymous credential system. In *CCS '02: Proceedings of the 9th ACM conference on Computer and communications security*, pages 21–30, New York, NY, USA, 2002. ACM.

[F] T. E. Foundation. Higgins: Open source identity framework.

[FPVFOGTOIT] O. Foundation and I. C. F. publish vision for open government through open identity technologies. Openid foundation.

[G] P. R. Group. Prime project.

[JCL06] S. H. Jan Camenisch and A. Lysyanskaya. Balancing accountability and privacy using e-cash (extended abstract). In *Security and Cryptography for Networks*, 2006.

[JLLP03] Z. Jiang, H. Luo, Y.-N. Li, and H. P. icard - foundation for a new ubiquitous computing architecture. In *ICC '03: Proceedings of the IEEE International Conference on Communications, 2003.*, pages 1211– 1217, 2003.

[KY05] A. Kiayias and M. Yung. Group signatures: Provable security, efficient constructions and anonymity from trapdoor-holders. In *Progress in Cryptology - Mycrypt 2005*, volume 3715 of *Lecture Notes in Computer Science*, pages 151–170. Springer-Verlag, 2005.

[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.

[NF06] T. Nakanishi and N. Funabiki. Group signature schemes with membership revocation for large groups. *IEICE Transactions*, 89-A(5):1275–1283, 2006.

[S79] A. Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, 1979.

[TAKS07] P. P. Tsang, M. H. Au, A. Kapadia, and S. W. Smiths. Blacklistable anonymous credentials: blocking misbehaving users without ttps. In *CCS '07: Proceedings of the 14th ACM conference on Computer and communications security*, pages 72–81, New York, NY, USA, 2007. ACM.