RZ 3526 (# 99540) 01/19/04
Computer Science 53 pages

Research Report

Designing Anonymous Applications with Accountability Using *idemix* Anonymous Credentials

Els Van Herreweghen

IBM Research GmbH
Zurich Research Laboratory
8803 Rüschlikon
Switzerland

Designing Anonymous Applications With Accountability Using *idemix* Anonymous Credentials

Els Van Herreweghen

Abstract

Anonymous credential systems [11, 10, 12, 13, 19] allow anonymous yet authenticated and accountable transactions between users and service providers. As such, they represent a powerful technique for protecting users' privacy when conducting Internet transactions. In this report, we show how to design privacy-friendly yet secure and accountable applications using the *idemix* anonymous credential system introduced in [9], based on protocols developed in [7]. In order to facilitate such design, we describe authentication, accountability and linkability features of the various *idemix* protocols as assertions on their in- and output parameter values. Using these assertions, we demonstrate the design of an application using *idemix* credentials based on the application's authentication, accountability and unidentifiability requirements.

1 Introduction

The protection of users' privacy when performing Internet or web-based transactions is an important factor in the acceptance and use of Internet and web services.

Solutions for minimizing release of personal information can be based on one of many proposed techniques for anonymizing the transport medium used between users and service providers, e.g., [20, 15, 21]. This may anonymize the user towards outsiders and, if desired, towards the service provider.

Service providers may require authentication (e.g., for controlling access to resources) or accountability of users' actions, in which case users need to prove their identity, or at least possession of a certificate or capability of a certain type. Such a certificate may contain a pseudonymous identity of the user, or contain only the necessary attributes required for accessing a certain service. However, when using certificates as defined by X.509 [17] or SPKI [14], or even certificates specifically constructed for conveying policy or authorization information as in [4], different uses of the same certificate still remain linkable to each other. They can eventually identify a user through a combination of context and addressing information from one or a series of transactions.

This linkability can be avoided by using an anonymous credential system (also called pseudonym system) [11, 10, 12, 13, 19]. In such a system, the organizations (service providers and credential issuers) know the users only by pseudonyms. Different pseudonyms of the same user cannot be linked. Yet, an organization can issue a credential to a pseudonym, and the corresponding user can prove possession of this credential to another organization (who knows him by a different pseudonym), without revealing anything more than the fact that the user owns such a credential.

[9] describes the design and implementation of the *idemix* credential system, based on protocols developed in [7]. The *idemix* system is described in terms of high-level primitives; these allow reasoning about security and privacy features, while hiding the complexity of the cryptographic protocols.

In this report, we describe authentication, accountability and unlinkability features of the various *idemix* protocols in terms of assertions on their in- and output parameter values. This representation facilitates the design of *idemix*-based applications based on authentication (authorization), accountability and unidentifiability requirements of the application.

The report is structured as follows. In Section 2, we give an overview of terminology used. In Section 3, we describe the *idemix* interfaces in more detail. These are the basic primitives described in [9] as well as signed variants of nym registration. Signed nym registration procedures allow a user to sign a nym using a (non-*idemix*) signature key. They provide the basis for user accountability by generating a proof of linking between a nym and the *idemix*-external certificate of a user.

Section 4 captures the result of each of the interactive protocols (nym registration, credential issuing and showing) in an 'assertion' defining a relationship between both participating parties' input and output values. E.g., an assertion about nym registration captures the following: "After a successful run of a nym registration protocol between a user and an organization, their respective nym values are related to each other as well as to the user's secret". These assertions about the system's building blocks allow us to concisely describe an application consisting of *idemix* interactive protocol invocations. We also specify which assertions are provable. Section 5 then specifies the result of local operations on transcripts (deanonymization and double-spending detection) as a function of the assertions about the transcripts they are invoked on. We capture relationships such as: "If a transcript is the result of a credential show protocol invocation with a local deanonymization parameter set, then deanonymization with the correct deanonymization key will reveal the nym the credential shown was issued on". These local operations can produce transcripts with which the results of their correctness can be proved; we can thus define provable assertions also about the results of deanonymization and double-spending detection. The definitions in Sections 4 and 5 thus allow us to reason about authentication (which credential was shown?) and provability (what can be proved with the various signatures and transcripts produced?) As a user's identity can be provably linked to a nym through the signed nym registration procedures, the provability of actions based on such a nym and on the credentials issued on it allows to fulfill requirements of user accountability.

Section 6 re-visits the various protocols from the viewpoint of the user's unidentifiability and the unlinkability of his actions; it examines which values (identities, nyms, credentials) become linkable by a user executing a certain protocol. By associating linkability assertions with individual *idemix* protocol executions, we can then reason about the effect of a sequence of protocol steps on the linkability of a user's actions, and on the linkability between his actions and his identity.

Section 7 completes the detailed description of the *idemix* system by defining additional functions needed or helpful in describing real applications. These are additional implementations of global deanonymization, as well as functions such as revocation and certification.

In Section 8, the previous detailed description of *idemix* functionality is used to design an application. The example shows how the assertions described in Sections 3 to 6 can be used be used to design an application based on authentication and accountability requirements from organizations on the one hand, and unidentifiability and unlinkability requirements from users on the other hand.

In Section 9, we discuss issues of trust related to user accountability and unidentifiability, accountability and liability of organizations, and certification. Section 10 concludes the report.

2 Terminology

In this section, we introduce some terminology and concepts related to unidentifiability and accountability as they will be used in this report.

2.1 Identity-Based vs. Attribute-Based Authentication and Authorization

Authentication is a service related to identification. Entity authentication as well as message authentication corroborate the identity of an entity (e.g., a person, a computer, etc.) associated with a communication channel or with a specific piece of information [22].

In traditional systems for access control, an entity's authorization to perform an action or to act under a certain role is typically derived from such *identity authentication.* This identity may have a global meaning, or it may only have a meaning to some participants in the system; it may also be a *pseudonym* as defined in the next section. In *identity-based authorization*, the authorization decision is thus based on the authenticated identity or pseudonym; the verifying (and authorizing) party derives necessary access rights from this identity or pseudonym.

Identity authentication of an authenticating party A to a verifying party (also called *relying party*) V can be achieved using a digital certificate certifying the linking between A's public key and its identity; this linking is certified (signed) by a trusted entity, such as a certification authority CA. A can now convince V of its identity by proving knowledge of the associated private key.

In this report, we will also discuss attribute authentication and attribute-based authorization. With *attribute authentication*, an authenticating party A convinces a verifying party V of the fact that A owns certain attributes; these attributes may but need not be unique to A and need not correspond to an identity. Examples of attributes are the right to access a certain resource or the age of the attribute holder. We will use the term 'proving ownership of an attribute' both for proving the exact value of an attribute (access right, age) as for proving a property of the attribute (e.g., age ≥ 18).

As is the case with identity authentication, the fact that A owns an attribute needs to be certified by a trusted authority. Using conventional certificates, this certification is realized by including the attributes in A's certificate. Using *credentials* as defined in Section 2.2, it is realized in a similar way, i.e., by the certificate issuer signing a piece of information including the attributes and a public value associated with A's secret. As with certificates, A can then prove ownership of an attribute certified in a credential by proving knowledge of this secret.

Attribute-based authorization will then allow A to perform an action, such as accessing a resource, based on the ownership of one or more attributes, rather than on its identity or on access rights derived from it by a relying party.

In the context of attribute-based authorization, the term ‘attribute’ covers more than only the fields in a certificate or credential carrying that name. It may stand for any property of the certificate or credential, other than the identity of the certificate holder, from which the relying party can derive necessary privileges. E.g., the fact that the certificate or credential used for authentication is signed by a certain issuer (i.e., can be verified using a specific public key) may be considered to be an attribute. Also, attribute-based authorization does not exclude that the certificate or credential may contain an identity or pseudonym; only, the authorization decision is not based on it. E.g., in the Secure Electronic Transactions (SET) [23] protocol for credit-card payments, the customer’s account number is not visible to the merchant receiving the payment; if the merchant’s acceptance of the payment is based only on the certificate being valid and having been issued by a trusted bank, the merchant’s authorization decision (in this case, the decision to grant access to a paid service or to deliver the purchased goods) is attribute-based.

2.2 Certificates and Credentials

A public-key certificate (short: *certificate*) is a piece of information signed by a trusted entity such as a certification authority CA. It binds the identity or name of a certified entity (A) and possibly additional relevant information (attributes) pertaining to A to the public key corresponding to a private key owned by A and known only to A. The certified key can be a public encryption key: in this case, information encrypted with A’s public key can be decrypted only using A’s private decryption key. Most often, we will talk about certificates certifying public signature keys: in this case, A authenticates to V (convinces a verifier V of its identity and/or of owning certain attributes) by proving ownership of the private signature key, e.g., by digitally signing a piece of information with that private key. V verifies the authentication using the signed information and the public key in A’s certificate. In order to verify the authenticity of this certificate (and of A’s public key), V needs an authentic copy of CA’s public key.

Rather than a real name or globally meaningful identity, public-key certificates may contain a local identity. A local identity is an identity which has a meaning only to specific parties in the system or is valid only for a short period of time. Some examples of local identities are: an employee number in a certificate issued by an employer; an account number in a certificate issued by a bank; or a temporary login name assigned to an employee allowing the employee to fill out this year’s employee opinion survey.

When a local identity is used with the goal to hide the authenticating entity’s real name or global identity from certain parties in the system or from outsiders, we will call the local identity a *pseudonym* (see also Section 2.4); a certificate certifying it is called a *pseudonym certificate.* The public key in a pseudonym certificate can be considered to be the pseudonym; a pseudonym certificate may but need not carry an additional identifier.

A pseudonym certificate can be used for identity-based as well as for attribute-based authorization. In the first case, authorization is based on the pseudonym; this requires that the relying party can link the pseudonym to another identity or to privileges. In the latter case, authorization is based on accompanying attributes in the certificate, and the relying party need not have any prior knowledge about the pseudonym. Note that the relying party can link all the transactions with the same pseudonym certificate to each other (and to the pseudonym), even if it cannot link the pseudonym to an identity.

Digital credentials can generally be defined as the digital equivalent of paper documents or other objects traditionally used to establish a person’s identity, attributes and privileges [5]. As such, a public-key certificate is a special type of digital credential; we will, however, use the term *credential* in a more restricted sense. The credentials discussed in this report belong to the class of *anonymous credentials* [11, 10, 12, 13, 19, 7]. Like a certificate, an anonymous credential is a certified piece of information allowing its owner to prove ownership of an identity and/or attributes contained in the certified data. Unlike a certificate, an anonymous credential allows its owner, A, to provide such a proof without the need for the verifier, V, to see or obtain (a copy of) the credential; A can authenticate (provide such a proof) multiple times without V being able to link the various authentications to each other.

2.3 Belief, Provability, Non-Repudiation, Accountability and Liability

According to definitions by Kailar [18], an individual is said to *believe* a statement if he is convinced of that statement; and a *proof* of a statement x is a set of statements that can collectively convey the validity of x to an audience. *Accountability* is the property whereby the association of a unique originator with an object or an action can be proved to a third party (i.e., a party other than the originator and the prover).

We illustrate these definitions with some examples. A may authenticate (sign) a message to V using the private key corresponding to a public-key certificate issued by CA. In order to believe that the message is signed by A, V needs to successfully verify the signature on the message with the public key in A's certificate, and also has to verify the authenticity of A's certificate; the latter verification is done using an authentic copy of CA's public key. For V to trust that a message signed with A's private key (more precisely: verified using A's public key) can indeed be attributed to A, V also needs to trust CA as well as the overall certificate infrastructure.

The message signed by A may also constitute proof to a third party if this party also trusts CA and the certificate infrastructure. I.e., using the signed message, V is able to prove to a third party that A made the statement contained in the message, or: A is accountable for the signed message.

The notion of accountability as defined here is based on and related to the property of *non-repudiation*, which is a service or property preventing the denial of previous commitments or actions [22]. Public-key based digital signatures are often attributed this property. However, in order for a digital signature to be considered non-repudiable and potentially legally binding, it should be established beyond reasonable doubt that the person to whom the public key is claimed to belong is indeed the only entity who could have made (or triggered the making of) the signature; that this person could verify and understood the contents and meaning of the data being signed; and that he was aware of possible consequences and legal interpretations of signing these data with this key. For this to be the case, many requirements have to be fulfilled safeguarding the security of the various procedures in the public-key infrastructure, including the generation of public/private keys, the generation of signatures, and the certification and registration of public keys by certification and registration authorities.

Liability, or obligation by law, assumes accountability; in our interpretation, the term liability implies a quantification of consequences for certain accountable facts. An individual is liable to perform a certain quantifiable action (such as paying an amount of money) if he can be held accountable for a certain fact, and if the quantifiable action is the *liability value* associated with this fact. The association between the accountable fact and the liability value needs to be known to and accepted by the individual in order for the individual to be held liable. The acceptance may be enforced by society, laws or the judicial system; e.g., a detention sentence is a generally accepted liability value for certain criminal behavior. In other cases, the acceptance needs to be more explicit. When registering a digital signature key and obtaining a public-key certificate, a person may accept a certain liability for data or transactions signed with that key (we will use *liability* also as a short form for *liability value*). E.g., when the public-key certificate allows the user to sign electronic payments, the user may accept a maximum liability for payments made with this certificate, thereby protecting himself against the consequences of the private key being stolen. When the public-key certificate allows the user to sign electronic contracts, the user's liability upon digitally signing a contract with that key may consist in being subject to the same dispute resolution procedure in court as is applied for paper-based contracts.

Liability applies to *legal entities*. In our definition, a legal entity is any entity that can be held accountable and liable before a court, and can be enforced to honor its liabilities. This can be a natural person, an organization or a company.

2.4 Pseudonyms and Pseudonymity, Anonymity, Linkability, (Un)identifiability

A *pseudonym* is an identifier with a local meaning. A user may choose or create his own pseudonym(s); or, organizations issuing certificates or credentials may create pseudonyms for users. Some pseudonyms are created based on inputs both from the user who will act under the pseudonym as from an organization registering the pseudonym [7].

A transaction carried out under a pseudonym (e.g., using a pseudonym certificate) is a *pseudonymous transaction.* As previously discussed, the use of pseudonyms assumes that it is not trivial, for at least some participants in the system or for outsiders, to derive a real identity from the pseudonym. According to the definition of anonymity in the following paragraph, the user in a pseudonymous transaction is anonymous towards the party or parties that cannot map the pseudonym used to the user's real identity.

An entity can be said to be *anonymous* towards another entity in a particular transaction if his identity in that transaction is concealed from that other entity. Anonymity of an entity A is thus always considered and specified with respect to one or more specific other entities in the transaction. E.g., in an electronic payment transaction, a user may act under a pseudonym under which he is known by his bank but which has no meaning to the merchant; in the transaction, which is pseudonymous, the user is anonymous towards the merchant but not towards the bank.

Factors other than the transaction protocol and its use of pseudonyms influence whether a user's identity indeed remains concealed from a relying party, i.e., whether the user remains *unidentifiable* to the relying party. In the above payment example, one can describe many scenarios through which the merchant could obtain the user's real name associated with a pseudonym. The user may inadvertently fill out his phone number in an optional field of an online form presented by the merchant during the transaction; or, the merchant may be able to derive the user's real identity from network or addressing information obtained during the transaction. Also, the merchant may recognize the pseudonym from a previous transaction where such identification occurred; or, the bank may collude with the merchant and reveal the real name associated with a pseudonym used. In some cases, the relying party has access to the list of real names of users owning a certain certificate or credential, without being able to map an authentication using such a certificate or credential to a specific user. An example is a voting procedure where the voting server knows the list of voters but another, trusted, entity has issued the anonymous voting credentials to individual users. A user's vote is of course anonymous only within the (potentially small) set of voters; in the extreme case of the number of voters being only one, the unique user's vote is fully identifiable despite the use of the anonymous voting process.

In the following, we will use the term *anonymity* towards an entity to indicate an intrinsic property of a transaction or protocol; if a transaction is anonymous towards a certain party then it allows the user to remain unidentifiable towards that party in the absence of identifying factors. Examples of identifying factors are a bad choice of parameters (e.g., the user filling out his phone number), possible identification through linking (e.g., the same pseudonym is used in another, identifiable transaction), parties colluding and sharing information (e.g., the bank revealing a customer's real name to the merchant), or system parameters such as the size of the set of users within which anonymity is achieved (e.g., a very small number of voters in the voting process). In this definition, the voting process with one voter remains anonymous even if the unique user's vote is identifiable.

Linkability between anonymous actions may thus have a high impact on the identifiability of each of the individual actions. In the payment example, the merchant may not only identify a user through linking with another, identifiable, transaction under the same pseudonym; also, the correlation of items purchased in both transactions together gives the merchant more information about the user's buying pattern and preferences; when combined with yet more purchasing transactions, such a detailed profile may ultimately identify the user.

Identifiability is thus a property of a transaction which can evolve over time and takes these transaction parameters, linkabilities, participants' behaviors and system parameters into account. Thus, an anonymous transaction can be identifiable because of transaction or system parameters; it can also become identifiable at a later point in time due to linking with other transactions or information, or because of information sharing between other entities. We cannot define identifiability as a property with a boolean value, as identifying factors may merely increase a *chance* of identification; also, we will not claim to have an exact measure of identifiability as we cannot quantify and take into account all of the potential identifying factors associated with a transaction. A transaction's *(level of) (un)identifiability* will be discussed only on a relative scale and implies a comparison with other transactions regarding intrinsic anonymity and taking into account a set of identifying factors which may have a varying impact on the chance of identification. E.g., a transaction allowing identity escrow by a trusted party has a lower level of unidentifiability than a similar transaction without identity escrow. Or, the payment transaction where a user simply has to trust his bank not to reveal the user's name to the merchant can be claimed to have a lower level of unidentifiability than a similar transaction where only a dedicated trusted entity can deanonymize a payment. The latter claim is true if that entity is trusted more than the bank with respect to deanonymization; it certainly holds if the deanonymization is verifiable and accountable (i.e., the fact that deanonymization occurred can be proved, and the conditions for deanonymization are concrete and verifiable). Also, the payment transaction where the user fills out his street name and postal code has a lower level of unidentifiability than the same transaction without the use of this parameter.

3 Primitives and Parameters of the Anonymous Credential System

3.1 Basic primitives

In this section, we discuss in more detail the *idemix* primitives introduced in [9]. Figure 1 presents an overview of the protocol primitives.

The various credential protocols and methods are illustrated using four participants: user (U), credential issuer (I), credential verifier (V), and deanonymizer (D). These participants represent the four possible roles that entities can take in the system (a special type of issuer, the Root Authority, is introduced in Section 3.3). An actual operational system may of course have multiple instantiations of each role; and some entities may take two roles at a time. E.g., the same organization can both issue and verify credentials.

Nym registration, credential issuing and credential showing are interactive protocols, and are represented in Figure 1 by double-headed arrows with the main protocol output for each participant drawn above the corresponding end of the arrow (minor output parameters such as error or success codes are not represented). Deanonymization and double-spending detection are non-interactive, local operations (by D and I, respectively); the information (transcripts) on which they operate is obtained from the respective verifiers through any synchronous or asynchronous communication medium, represented by broken single-headed arrows. The protocols are briefly described in the next paragraphs; differences with the specifications in [9] are pointed out and motivated subsequently.

Nym registration between U and I consists of U and I calling the respective primitives $\mathsf{URegNym}()$ and $\mathsf{ORegNym}()$, resulting in I obtaining OrgNym_{UI} and U obtaining $\mathrm{UserNym}_{UI}$. I issuing a credential to U consists of U and I calling $\mathsf{UGetCred}()$ and $\mathsf{OIssueCred}()$, respectively, which results in U obtaining $\mathrm{UserCred}_{UI}$; I has no real protocol output but a successful protocol execution provides him with the information that a credential of type $\mathrm{CredInfo}_{UI}$ has been issued to OrgNym_{UI}. When U shows a credential to V, U and V call $\mathsf{UShowCred}()$ and $\mathsf{OVerifyCred}()$, respectively, resulting in V obtaining a $\mathrm{Transcript}_{UV}$ of the show protocol. If the local or global deanonymization option, with deanonymizer D, was used as an option in the credential show, the verifier can send the transcript $\mathrm{Transcript}_{UV}$ for deanonymization to D. D then calls $\mathsf{DODeAnonLocal}()$, revealing the nym OrgNym_{UI} the credential was issued on, respec-

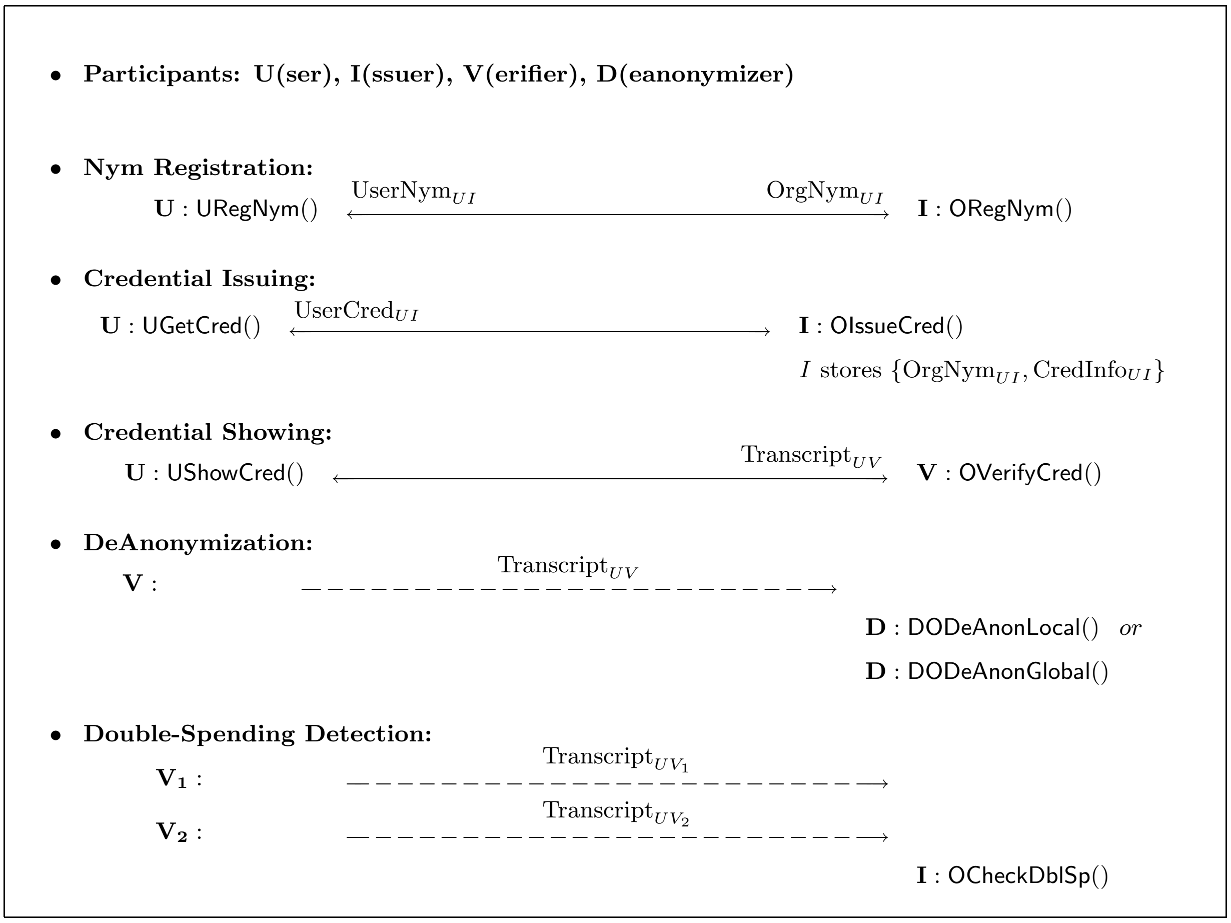

Figure 1: Anonymous Credential System: Overview of Basic Primitives

tively DODeAnonGlobal(), revealing the user's nym OrgNym_{UR} with a Root Authority (in case of global deanonymization).

When calling OCheckDblSp() on two transcripts (from V_1 and V_2, which can be the same or different verifiers) which result from showing the same one-show credential, the result reveals the nym on which the credential was issued. [7] describes both on-line and off-line double-spending detection. With on-line double-spending detection, every credential verification by V is checked for double-spending by I during the show protocol execution, i.e. the execution of the show protocol fails if the credential is double-spent. With off-line double-spending detection, the actual show protocol execution succeeds even if the credential is double-spent, and I verifies transcripts after the fact. Here, we have represented only off-line double-spending detection and assumed an interface where two transcripts are checked against each other. In a setting where every new transcript needs to be verified against a database of existing transcripts, it is possible to optimize the process by first searching for potentially matching transcripts and only call the OCheckDblSp() primitive on the resulting candidates.

Figures 2 and 3 then show the key material (user secret, public and private keys) used in the system, as well as the parameters and parameter contents of each primitive. For ease of reading, parameter instance and type names are distinguished only by subscripts, e.g., UserSecret_U is U's parameter of type UserSecret; Comm_{UI} is a parameter of type Comm used by U and I in the current protocol instance. $[\text{Par}_{XY}]$ is an optional parameter of type Par used in a protocol execution between X and Y. DSKey, DPKey are a

- **Key Material: U:** UserSecret_U, **I:** IssuerKeys_I, **V:** VerifierKeys_V, **D:** DeAnOrgKeys_D

- **Parametrized Primitives**

  URegNym($\text{Comm}_{UI}, \text{UserSecret}_U$) returns UserNym_{UI}

  ORegNym(Comm_{UI}) returns OrgNym_{UI}

  UGetCred($\text{Comm}_{UI}, \text{UserSecret}_U, \text{UserNym}_{UI}, \text{CredInfo}_{UI}$) returns UserCred_{UI}

  OIssueCred($\text{Comm}_{UI}, \text{IssuerKeys}_I, \text{OrgNym}_{UI}, \text{CredInfo}_{UI}$) returns `void`

  UShowCred($\text{Comm}_{UV}, \text{UserSecret}_U, \text{UserCred}_{UI}, \text{CredShowInfo}_{UV}$,
  $\text{CredShowFeatures}_{UV}$, [$\text{UserNym}_{UV}$], [$\text{Msg}_{UV}$]) returns `void`

  OVerifyCred($\text{Comm}_{UV}, \text{VerifierKeys}_V, \text{CredShowInfo}_{UV}, \text{CredShowFeatures}_{UV}$,
  [OrgNym_{UV}], [Msg_{UV}]) returns Transcript_{UV}

  DODeAnonLocal($\text{DeAnOrgKeys}_D, \text{Transcript}_{UV}$) returns OrgNym_{UI}

  DODeAnonGlobal($\text{DeAnOrgKeys}_D, \text{Transcript}_{UV}$) returns OrgNym_{UR}

  OCheckDblSp($\text{IssuerKeys}_I, \text{Transcript}_{UV_1}, \text{Transcript}_{UV_2}$) returns OrgNym_{UI}

Figure 2: Anonymous Credential System: Key Material and Parametrized Primitives

deanonymization private and public key; ISKey, IPKey are an issuing private and public key; and VSKey, VPKey represent a verification private and public key.

When a credential is issued, it is verified by the user receiving it; when it is shown, it is verified by another organization. In both cases, the public key of the issuing organization is necessary for verification; it is, however, not visible as an individual parameter as it is part of a larger parameter structure. In UGetCred(), the IPKey field in the CredInfo parameter contains the public credential issuing key needed by the user to verify the credential received. Likewise, in OVerifyCred(), the IPKey field in the CredShowInfo parameter (the use of this new parameter is explained in subsequent paragraphs) contains the public credential issuing key needed by the verifier to verify the credential shown. In both cases, it is up to the verifying party (the user receiving the credential, respectively the organization verifying the credential) to ensure that the public key in CredInfo or CredShowInfo indeed belongs to the issuer he expects the credential to be issued by; this can be done by matching the public key against the issuer's public issuing key certificate (see Section 7.3).

Similarly, the user showing a credential needs the public verification key of the organization verifying the credential. This public key is part of the CredShowFeatures parameter of the UShowCred() primitive; it is the user's responsibility to verify the correctness of this key before invoking UShowCred().

The primitives' parameters and their contents are fundamentally the same as discussed in [9]. Following are some minor differences:

- In [9], we discussed that expiration times or attributes that are (dynamically) shown or proved about a credential may include ranges of values, while the (static) expiration times or attributes of a credential are point values. Here, we have made this fact more explicit by the introduction of a new parameter type, CredShowInfo, which is used instead of CredInfo to express which of a credential's features are actually shown or verified in the UShowCred() and OVerifyCred() primitives. From an implementation perspective, we consider CredInfo to extend the CredShowInfo type by restricting ranges to point values; we can thus assign a variable of type CredInfo to a variable of type CredShowInfo.

- **Contents of Composed Parameter Types**

UserNym	{OrgNym, UserNymSecret}
CredInfo	{IPKey, MultiShow, [Expiration], [CredAttrs]}
CredAttrs	{{AttrName, AttrValue}, ...}
Transcript	{CryptoTrscr, CredShowInfo, CredShowFeatures, [OrgNym], [Msg]}
CredShowInfo	{IPKey, MultiShow, [Expiration], [CredAttrs]}
CredShowFeatures	{RelNym, VPKey, [LocalDeAnData], [GlobalDeAnData]}
LocalDeAnData	{DPKey, DeAnCondition}
GlobalDeAnData	{DPKey, DeAnCondition}
IssuerKeys	{ISKey, IPKey}
VerifierKeys	{VSKey, VPKey}
DeAnOrgKeys	{DSKey, DPKey}

- **Non-Cryptographic Primitive Types**

Expiration, AttrName, AttrValue, DeAnCondition, Msg	*String* (`null`: none)
MultiShow, RelNym	*Boolean*

Figure 3: Anonymous Credential System: Parameter Types and Contents

- In [9], no explicit distinction was made between the key types for issuing and verification ('OrgKeys' in the parameter lists of the respective primitives). Here, we make the description more general by introducing different key types; this does not exclude the use of the same key for both purposes by an organization both issuing and verifying credentials.

- As discussed in [9], the credential show protocol can be used to produce a signature, optionally over an additional message. In the following, we will assume this use of the show protocol, which will allow the verifier to prove the showing of a credential after the fact. We have provided for the signing of an additional message by modifying UShowCred() and OVerifyCred() to include an optional Msg argument, representing the message being signed. We will sometimes refer to the signing of a message as part of showing a credential as 'signing the message with the credential'.

- In order to hide implementation details, we do not explicitly mention system parameters. It is assumed that all the parties in the system call the various primitives (including the primitives to initialize key material, which are not discussed here) using the same system-wide set of system parameters. Therefore, previous parameters UserNymSysData, OrgNymSysData, DeAnOrgNymSysData, which included entities' keys as well as system parameters, were renamed UserSecret, IssuerKeys, DeAnOrgKeys.

- As discussed in [9], the primitives involving interactive protocols (nym registration, credential issuing and showing) rely on a communication channel being set up between the parties involved in the protocol, based on addressing information obtained at application level. In order to reason about the results of a protocol execution, we now explicitly include these communication channels in the primitives' parameter list. A communication channel Comm is a global entity with a finite duration, and with a client (originator) and a server (receiver) interface; intuitively, a user and an issuer calling URegNym(), respectively ORegNym(), with the same Comm parameter, are communicating with each other.

Whereas anonymity of credentials and of credential shows preserves the anonymity of the user at the application level, this anonymity has to be supported by anonymity of the user at the communication level. Without this 'sender anonymity', the organization the user is communicating with, as well as an external observer, could derive the identity of the user by tracing the communication. Many methods have been proposed for anonymizing communication; some examples are Crowds [21], anonymizers [1], Onion Routing [15] and Mix networks [16, 20]. When designing *idemix* applications, we assume the presence of sender anonymity using one of these methods.

- In [9], we did not make a distinction between application-level transcripts and cryptographic transcripts. Here, we want to more strictly separate information used and generated by the interactive protocols from application-level information the calling application may want to store; we therefore modified return value types of some of the operations as follows. Our definition of protocol transcript is now limited to the transcript Transcript_{UV} returned by credential verification OVerifyCred(). This transcript contains the cryptographic information CryptoTrscr generated by the credential verification; it also contains protocol parameters (CredShowInfo_{UV}, $\text{CredShowFeatures}_{UV}$, UserNym_{UV} and Msg_{UV}) that may be needed by an after-the fact verifier of the transaction, or a transaction deanonymizer, to extract public keys, deanonymization conditions and other arguments. ORegNym() now only returns an OrgNym - the issuer application may of course choose to store additional information (called X in [9]) in an application-level transcript. OIssueCred() has no return value; the issuer may store application-level information indicating what type of credential was issued on which nym (the $\{\text{OrgNym}_{UI}, \text{CredInfo}_{UI}\}$ information in Figure 1, called 'OrgCred' in [9]), or referring to additional cryptographic and/or application-level transcripts related to this credential issuing (e.g., the transcripts of the credential shows that were necessary in order for the issuer to issue the new credential).

3.2 Signed Nym Registration

During registration of a nym, a user U can additionally authenticate to the nym issuer using a signature. This signature uses a signature key the public part of which is certified by certification authority CA in a certificate Cert_{CA-U} external to the *idemix* system. The signature provides the issuer of the nym with a provable linking between the nym and the external certificate. The protocol in Figure 4 shows such a signed nym registration procedure between a user U and an issuer I. The provable linking between the nym and the external certificate is represented by U's signature SIG_{UI} on OrgNym_{UI}.

The details of the signed nym registration protocol are described in [6]. In the version presented here, we add an optional Msg argument to these nym registration primitives. This allows the user to sign (and the nym issuer to verify) the additional message and link it to Cert_{CA-U} and OrgNym_{UI}. This can be trivially realized by adding a hash of the Msg as argument to the user's signature and the organization's signature verification function in the signed nym registration protocol.

Signed nym registration is most typically applied for the registration of a specially formed 'root nym' with the Root Authority. The registration of such a root nym is a prerequisite for global deanonymization; it is introduced in the following section and its use is illustrated in the example application in Section 8. Signed nym registration can, however, be requested by any organization other than the Root Authority in order to obtain a provable linking between a 'normal' nym and an external certificate.

3.3 Root Nym Registration

Root nym registration uses signed nym registration to register a *root nym*, which is a nym with special features. Root nym registration is a prerequisite for enabling global deanonymization, as discussed in Sec-

- **Participants and Key Material:**

  User (**U**): UserSecret_U, SSKey_U, Cert_{CA-U}

  Issuer (**I**): IssuerKeys_I

- **Signed Nym Registration:**

$$\mathbf{U} : \mathsf{URegSignedNym}() \quad \overset{\text{UserNym}_{UI} \qquad\qquad \text{OrgNym}_{UI}, \text{SIG}_{UI}}{\longleftrightarrow} \quad \mathbf{I} : \mathsf{ORegSignedNym}()$$

I stores $\{\text{OrgNym}_{UI}, \text{SIG}_{UI}, \text{Cert}_{CA-U}, [\text{Msg}_{UI}]\}$

- **Parametrized Primitives**

$\mathsf{URegSignedNym}(\text{Comm}_{UI}, \text{UserSecret}_U, \text{SSKey}_U, \text{Cert}_{CA-U}, [\text{Msg}_{UI}])$ returns UserNym_{UI}
$\mathsf{ORegSignedNym}(\text{Comm}_{UI}, \text{Cert}_{CA-U}, [\text{Msg}_{UI}])$ returns $\text{OrgNym}_{UI}, \text{SIG}_{UI}$

- **Additional Inputs and Outputs**

$\text{SSKey}_U, \text{Cert}_{CA-U}$	U's signature secret key and public-key certificate (certifying the public key SPKey_U) in an external PKI
SIG_{UI}	A signature (using SSKey_U) on OrgNym_{UI} [and Msg_{UI}], allowing I to prove the linking between Cert_{CA-U} and OrgNym_{UI} [and Msg_{UI}]

Figure 4: Anonymous Credential System: Signed Nym Registration

tion 5.2. If root nym registration is applied, it is provided by a dedicated organization, which we will call the Root Authority R; R enforces that every user registers exactly one root nym. A root nym can be used in the same way as an ordinary nym; e.g., R can issue a credential on it. A credential issued by R on a root nym is called a *root credential.*

Root nym registration enables global deanonymization because of following features of the root nym, the external certificate and the registration process:

- The root nym is formed in a special way such that it is a component of all the other nyms of the same user (related to the same UserSecret_U). This feature is a result of the root nym depending only on the user's secret UserSecret_U and public system parameters. It ensures that global deanonymization of any credential show transaction by the same user (of any credential issued on any nym) will reveal the user's root nym. Of course, mechanisms have to be in place to ensure that a user's nyms indeed are formed correctly, i.e., that they are properly linked to the user's root nym. The mechanism to verify that a new nym is correctly linked to a user's root nym is the verification of that user's root credential relative to this new nym. Showing a credential relative to a nym is discussed in Section 4.6; how it is applied to global deanonymization is discussed in Section 5.2.

- The use of the signed nym registration procedure (as part of root nym registration) ensures that R has a provable linking between the root nym and an external certificate Cert_{CA-U}. This external certificate is issued by a Certificate Authority CA trusted with guaranteeing a unique linking between Cert_{CA-U} and a legal entity. (In further discussion, we often assume that U is a human user; note, however, that U could be another type of legal entity, such as a company.). This guarantee is necessary to provide a real means of recourse in case of, e.g., criminal misuse of a credential.

Figure 5 shows the root nym registration process. Inputs and outputs of the root nym registration primi-

- **Participants and Key Material:**

User (**U**):	UserSecret_U, SSKey_U, Cert_{CA-U}
Root Authority (**R**):	IssuerKeys_R

- **Root Nym Registration:**

$$\mathbf{U} : \mathsf{URegRootNym}() \quad \xleftrightarrow{\text{UserNym}_{UR} \qquad\qquad \text{OrgNym}_{UR}, \text{SIG}_{UR}} \quad \mathbf{R} : \mathsf{ORegRootNym}()$$

R stores $\{\text{OrgNym}_{UR}, \text{SIG}_{UR}, \text{Cert}_{CA-U}, [\text{Msg}_{UR}]\}$

- **Parametrized Primitives**

$\mathsf{URegRootNym}(\text{Comm}_{UR}, \text{UserSecret}_U, \text{SSKey}_U, \text{Cert}_{CA-U}, [\text{Msg}_{UR}])$ returns UserNym_{UR}

$\mathsf{ORegRootNym}(\text{Comm}_{UR}, \text{Cert}_{CA-U}, [\text{Msg}_{UR}])$ returns $\text{OrgNym}_{UR}, \text{SIG}_{UR}$

- **Additional Inputs and Outputs**

$\text{SSKey}_U, \text{Cert}_{CA-U}$	U's signature secret key and public-key certificate (certifying the public key SPKey_U in an external PKI
SIG_{UR}	A signature (using SSKey_U) on OrgNym_{UR} [and Msg_{UR}], allowing R to prove the linking between Cert_{CA-U} and OrgNym_{UR} [and Msg_{UR}]

Figure 5: Anonymous Credential System: Root Nym Registration

tives ($\mathsf{URegRootNym}()$ and $\mathsf{ORegRootNym}()$) are identical to the ones of their counterparts for signed nym registration ($\mathsf{URegSignedNym}()$ and $\mathsf{ORegSignedNym}()$). The fact that UserNym_{UR} and OrgNym_{UR} now represent a root nym will, however, result in a different assertion about the result of the protocol execution; this is shown in Section 4.3.

The goal of the external authentication in signed and root nym registration is a (provable) linking of a nym with a legal entity. One could also achieve this using a paper passport or other means of identification (as opposed to Cert_{CA-U}) combined with a handwritten signature. On the one hand, this would obviate the need for prior registration of U with an external CA; but, on the other hand, it would require a more elaborate registration procedure by the organization (I or R) registering the nym. When designing applications using signed and root nym registration, we will always assume that the external authentication is done as depicted in Figures 4 and 5, i.e., using Cert_{CA-U} and a signature (SIG_{UI} or SIG_{UR}) with SSKey_U.

4 Assertions on Nyms, Credentials and Transcripts Resulting from Interactive Protocols

After successful execution of an interactive protocol, both participants' protocol outputs are related; e.g., UserNym and OrgNym are related. As each participant's output depends on both participants' invocation parameters, such a relationship is global; it typically cannot be expressed as a relationship between parameters of one participant's primitive invocations alone.

Here, we try to express these global relationships between, on the one hand, instances of primitive invocations and their parameters and, on the other hand, the contents of the nyms, credentials and transcripts established.

They take the form of postconditions, called *assertions*, on protocol inputs and outputs from successful primitive invocations. The choice of representing protocol executions using such a global view will enable us to represent an interactive protocol execution with one statement or assertion; this will facilitate the description of applications.

We will also specify which of the assertions are provable; e.g., the signature resulting from a signed nym registration procedure proves a relationship between an external certificate and a nym; the transcript from a credential show with certain parameters proves that a credential was shown with these parameters.

When the provability of an assertion is important in describing applications, we will make it explicit by its naming. E.g., after signed nym registration, the issuing organization can prove the linking between the newly registered nym and the user's external certificate; the resulting assertion is called `SignedNymProof`().

Provability, together with the possibility to link nyms (and actions based on them) to external certificates, and thus to 'real users', form the basis for user accountability in the *idemix* system.

4.1 Nym Registration

$\mathsf{URegNym}(\mathrm{Comm}_{UI}, \mathrm{UserSecret}_{U})$ returns $\mathrm{UserNym}_{UI}$,
$\mathsf{ORegNym}(\mathrm{Comm}_{UI})$ returns OrgNym_{UI}
$\rightarrow \mathtt{Nym}(\mathrm{UserSecret}_{U}, \mathrm{OrgNym}_{UI}, \mathrm{UserNym}_{UI})$.

If a user and an issuer invoke $\mathsf{URegNym}()$ and $\mathsf{ORegNym}()$ on the same Comm_{UI} channel with inputs as indicated, then the respective outputs $\mathrm{UserNym}_{UI}$ and OrgNym_{UI} are linked by the resulting `Nym` assertion. `Nym` is an assertion about a global relationship between values. It can informally be expressed as: "OrgNym_{UI} and $\mathrm{UserNym}_{UI}$ are the issuer's and user's representation of the same nym, and are related to the user's $\mathrm{UserSecret}_{U}$".

4.2 Signed Nym Registration

$\mathsf{URegSignedNym}(\mathrm{Comm}_{UI}, \mathrm{UserSecret}_{U}, \mathrm{SSKey}_{U}, \mathrm{Cert}_{CA-U}, \mathrm{Msg}_{UI})$ returns $\mathrm{UserNym}_{UI}$,
$\mathsf{ORegSignedNym}(\mathrm{Comm}_{UI}, \mathrm{Msg}_{UI})$ returns $\mathrm{OrgNym}_{UI}, \mathrm{Cert}_{CA-U}, \mathrm{SIG}_{UI}, \mathrm{Msg}_{UI}$
$\rightarrow \mathtt{Nym}(\mathrm{UserSecret}_{U}, \mathrm{OrgNym}_{UI}, \mathrm{UserNym}_{UI})$,
$\rightarrow \mathtt{SignedNymProof}(\mathrm{SIG}_{UI}, \mathrm{Cert}_{CA-U}, \mathrm{OrgNym}_{UI}, \mathrm{Msg}_{UI})$.

A signed nym registration results in a `Nym`() assertion as well as a `SignedNymProof`() assertion. `SignedNymProof`() expresses the existence of a proof, in SIG_{UI}, that OrgNym_{UI} and Msg_{UI} are signed with the private key the public counterpart of which is certified in Cert_{CA-U}. As the values in `SignedNymProof`() are all known to I, I can prove this signed linking.

Msg_{UI} in the above may be `null`; this will, in general be the case for parameters indicated as being optional in Figures 2 to 5, unless their value is explicitly used in any of the preconditions.

4.3 Root Nym Registration

$\mathsf{URegRootNym}(\mathrm{Comm}_{UR}, \mathrm{UserSecret}_{U}, \mathrm{SSKey}_{U}, \mathrm{Cert}_{CA-U}, \mathrm{Msg}_{UR})$ returns $\mathrm{UserNym}_{UR}$,
$\mathsf{ORegRootNym}(\mathrm{Comm}_{UR}, \mathrm{Msg}_{UR})$ returns $\mathrm{OrgNym}_{UR}, \mathrm{Cert}_{CA-U}, \mathrm{SIG}_{UR}, \mathrm{Msg}_{UR}$
$\rightarrow \mathtt{RootNym}(\mathrm{UserSecret}_{U}, \mathrm{OrgNym}_{UR}, \mathrm{UserNym}_{UR})$,
$\rightarrow \mathtt{SignedNymProof}(\mathrm{SIG}_{UR}, \mathrm{Cert}_{CA-U}, \mathrm{OrgNym}_{UR}, \mathrm{Msg}_{UR})$.

The `RootNym`() assertion indicates that the nym is of this special root nym type; the `SignedNymProof`()

assertion, as with signed nym registration, expresses the provable linking between the nym and the message with the external public key and certificate.

Moreover, a root nym is also a nym:

$\texttt{RootNym}(\text{UserSecret}_U, \text{OrgNym}_{UR}, \text{UserNym}_{UR})$
$\rightarrow \texttt{Nym}(\text{UserSecret}_U, \text{OrgNym}_{UR}, \text{UserNym}_{UR})$.

4.4 Credential Issuing

$\texttt{Nym}(\text{UserSecret}_U, \text{OrgNym}_{UI}, \text{UserNym}_{UI})$,
$\text{CredInfo}_{UI}.\text{IPKey} = \text{IssuerKeys}_I.\text{IPKey}$,
$\mathsf{UGetCred}(\text{Comm}_{UI}, \text{UserSecret}_U, \text{UserNym}_{UI}, \text{CredInfo}_{UI})$ returns UserCred_{UI},
$\mathsf{OIssueCred}(\text{Comm}_{UI}, \text{IssuerKeys}_I, \text{OrgNym}_{UI}, \text{CredInfo}_{UI})$
$\rightarrow \texttt{Cred}(\text{OrgNym}_{UI}, \text{UserCred}_{UI}, \text{CredInfo}_{UI})$.

If OrgNym_{UI} and UserNym_{UI} are U's nym with I related to UserSecret_U, and U and I invoke UGetCred() and OIssueCred() on the same Comm_{UI} channel to issue a credential on OrgNym_{UI}, and the IPKey in the requested CredInfo is indeed the public credential issuing key IPKey_I in the IssuerKeys used by I for credential issuing, then the invocation by U and I of UGetCred() and OIssueCred() with inputs as indicated results in a UserCred_{UI} about which we can make the resulting `Cred` assertion: "UserCred_{UI} is a valid credential with features CredInfo_{UI} (including IPKey_I) on OrgNym_{UI}".

Note that the condition $\text{CredInfo}_{UI}.\text{IPKey} = \text{IssuerKeys}_I.\text{IPKey}$ is actually a precondition for the success of the respective UGetCred() and OIssueCred() invocations; this is not captured in our notation. As our focus is on representing the relationship between successful invocations (with specific parameter values) and resulting assertions, this does not pose a problem.

4.5 Showing a Credential - Not Relative to a Nym

$\texttt{Cred}(\text{OrgNym}_{UI}, \text{UserCred}_{UI}, \text{CredInfo}_{UI})$,
$\texttt{Nym}(\text{UserSecret}_U, \text{OrgNym}_{UI}, \text{UserNym}_{UI})$,
$\texttt{Fulfills}(\text{CredInfo}_{UI}, \text{CredShowInfo}_{UV})$,
$\text{CredShowFeatures}_{UV}.\text{RelNym} = \texttt{false}$,
$\text{CredShowFeatures}_{UV}.\text{VPKey} = \text{VerifierKeys}_V.\text{VPKey}$,
$\mathsf{UShowCred}(\text{Comm}_{UV}, \text{UserSecret}_U, \text{UserCred}_{UI}, \text{CredShowInfo}_{UV}, \text{CredShowFeatures}_{UV}, \texttt{null}^1, \text{Msg}_{UV})$,
$\mathsf{OVerifyCred}(\text{Comm}_{UV}, \text{VerifierKeys}_V, \text{CredShowInfo}_{UV}, \text{CredShowFeatures}_{UV}, \texttt{null}^1, \text{Msg}_{UV})$
returns Transcript_{UV}
$\rightarrow \texttt{ShowTranscript}(\text{Transcript}_{UV}, \text{UserCred}_{UI}, \text{CredShowInfo}_{UV}, \text{CredShowFeatures}_{UV}, \texttt{null}, \text{Msg}_{UV})$.

If UserCred_{UI} is a valid credential with features CredInfo_{UI} (including IPKey_I) on UserNym_{UI}, UserNym_{UI} is related to UserSecret_U and OrgNym_{UI}, the credential features CredInfo_{UI} fulfill the CredShowInfo_{UV} features that U wants to show to V in this particular UShowCred() invocation, and the verification key of V matches the verification key VPKey in $\text{CredShowFeatures}_{UV}$, then an invocation by U and V of UShowCred() and OVerifyCred() on the same Comm_{UV} channel with inputs as indicated results in a Transcript_{UV} about which the resulting `ShowTranscript` assertion holds. As the credential show is not relative to a pseudonym shared with V (CredShowFeatures.RelNym = `false`), UserNym in UShowCred(), OrgNym in OVerifyCred(), and OrgNym in the `ShowTranscript` assertion are `null`[1].

[1]The value of the UserNym (OrgNym) parameter in UShowCred() (OVerifyCred()) should be `null`; one can however assume that even non-`null` input values are not taken into account in the protocol if CredShowFeatures.RelNym = `false`.

`ShowTranscript` is a global assertion linking UserCred_{UI} to the show protocol parameters and the transcript.

In Section 3.1, we assumed that the credential show protocol results in a proof. We can state this as follows:

$\texttt{ShowTranscript}(\text{Transcript}_{UV}, \text{UserCred}_{UI}, \text{CredShowInfo}_{UV}, \text{CredShowFeatures}_{UV}, \texttt{null}, \text{Msg}_{UV})$
$\rightarrow \texttt{SigProof}(\text{Transcript}_{UV}, \text{CredShowInfo}_{UV}, \text{CredShowFeatures}_{UV}, \texttt{null}, \text{Msg}_{UV})$.

The `SigProof`() assertion expresses that, with Transcript_{UV}, V can prove that:

- the credential specified in CredShowInfo_{UV} was shown with $\text{CredShowFeatures}_{UV}$;
- the optional Msg_{UV} was signed together with it ('was signed with the credential').

4.6 Showing a Credential - Relative to a Nym

$\texttt{Cred}(\text{OrgNym}_{UI}, \text{UserCred}_{UI}, \text{CredInfo}_{UI})$,
$\texttt{Nym}(\text{UserSecret}_{U}, \text{OrgNym}_{UI}, \text{UserNym}_{UI})$,
$\texttt{Fulfills}(\text{CredInfo}_{UI}, \text{CredShowInfo}_{UV})$,
$\text{CredShowFeatures}_{UV}.\text{RelNym} = \texttt{true}$,
$\text{CredShowFeatures}_{UV}.\text{VPKey} = \text{VerifierKeys}_{V}.\text{VPKey}$,
$\texttt{Nym}(\text{UserSecret}_{U}, \text{OrgNym}_{UV}, \text{UserNym}_{UV})$,
$\mathsf{UShowCred}(\text{Comm}_{UV}, \text{UserSecret}_{U}, \text{UserCred}_{UI}, \text{CredShowInfo}_{UV}, \text{CredShowFeatures}_{UV}, \text{UserNym}_{UV}, \text{Msg}_{UV})$,
$\mathsf{OVerifyCred}(\text{Comm}_{UV}, \text{VerifierKeys}_{V}, \text{CredShowInfo}_{UV}, \text{CredShowFeatures}_{UV}, \text{OrgNym}_{UV}, \text{Msg}_{UV})$
    returns Transcript_{UV}
$\rightarrow \texttt{ShowTranscript}(\text{Transcript}_{UV}, \text{UserCred}_{UI}, \text{CredShowInfo}_{UV}, \text{CredShowFeatures}_{UV}, \text{OrgNym}_{UV}, \text{Msg}_{UV})$.

As the credential is shown relative to a pseudonym shared with V (CredShowFeatures.RelNym = `true`), $\mathsf{UShowCred}()$, $\mathsf{OVerifyCred}()$ and the `ShowTranscript`() assertion have UserNym_{UV}, respectively OrgNym_{UV} as parameters; in addition, UserNym_{UV} and OrgNym_{UV} have to be linked by a `Nym` assertion in order for the credential show protocol to succeed.

We can again state an assertion about provability:

$\texttt{ShowTranscript}(\text{Transcript}_{UV}, \text{UserCred}_{UI}, \text{CredShowInfo}_{UV}, \text{CredShowFeatures}_{UV}, \text{OrgNym}_{UV}, \text{Msg}_{UV})$
$\rightarrow \texttt{SigProof}(\text{Transcript}_{UV}, \text{CredShowInfo}_{UV}, \text{CredShowFeatures}_{UV}, \text{OrgNym}_{UV}, \text{Msg}_{UV})$.

The `SigProof`() assertion now expresses that, with Transcript_{UV}, V can prove that:

- the credential specified in CredShowInfo_{UV} was shown with $\text{CredShowFeatures}_{UV}$;
- the optional Msg_{UV} was signed together with it;
- OrgNym_{UV} was the signer's nym with V.

5 Local Operations on Transcripts

Using assertions on nyms, credentials and transcripts as preconditions, we can now describe the results of the methods for deanonymization and double-spending detection in terms of the inputs and outputs of the interactive protocols.

5.1 Local Deanonymization

$\texttt{Nym}(\text{UserSecret}_{U}, \text{OrgNym}_{UI}, \text{UserNym}_{UI})$,
$\texttt{Cred}(\text{OrgNym}_{UI}, \text{UserCred}_{UI}, \text{CredInfo}_{UI})$,

ShowTranscript$(\text{Transcript}_{UV}, \text{UserCred}_{UI}, \text{CredShowInfo}_{UV}, \text{CredShowFeatures}_{UV}, \text{OrgNym}_{UV}, \text{Msg}_{UV})$,
$\text{CredShowFeatures}_{UV}$.LocalDeAnData.DPKey = DeAnOrgKeys_{D}.DPKey
$\rightarrow$ LDeanonymizable$(\text{DeAnOrgKeys}_{D}, \text{Transcript}_{UV}, \text{OrgNym}_{UI})$.

If a transcript Transcript_{UV} was generated during a credential show with local deanonymization using a deanonymizer's DPKey_{D}, then the LDeanonymizable() assertion holds. The LDeanonymizable() assertion expresses the following relationship between DeAnOrgKeys_{D}, Transcript_{UV} and OrgNym_{UI}: if DODeAnonLocal() is called on Transcript_{UV} with DeAnOrgKeys_{D}, it will return OrgNym_{UI}, the nym on which the credential was issued.

Note that the deanonymization condition LocalDeAnData.DeAnCondition is not taken into account. The approach taken, for local as well as for global deanonymization, is that the invocation, not the outcome of the deanonymization operation, should depend on the deanonymization condition being fulfilled. Thus, D can (but should not) successfully invoke DODeAnonLocal() even if the deanonymization condition is not fulfilled.

By showing UserCred_{UI} with local deanonymization using DPKey_{D}, the resulting Transcript_{UV} contains an encryption $EV_D(\text{OrgNym}_{UI})$ of U's nym with I, verifiably encrypted with D's public deanonymization key DPKey_{D} (actually, the encrypted value is not the nym itself but a validating tag which I can associate with OrgNym_{UI}; in this discussion, we will not make this distinction). V can verify that the encrypted nym is indeed the nym the credential was issued on. V trusts D to decrypt this information if the deanonymization condition is fulfilled; as OrgNym_{UI} has no meaning to V, V has to rely on I to take appropriate action or provide other information associated with OrgNym_{UI}.

5.1.1 Local Deanonymization With Proof of Correctness

The protocols described in [6] allow a deanonymizing organization to prove the correctness of a deanonymization, i.e., to prove that the resulting OrgNym is indeed the one encrypted in the deanonymized transcript. This can be done interactively or non-interactively. As this proof is a valuable feature in building secure applications, we will define also specific primitives for deanonymizations producing such a proof.

For local deanonymization, we define a new operation DODeAnonLocalWProof() which returns a deanonymization transcript DeAnTranscript rather than only a nym. The deanonymization transcript contains the nym as well as a (non-interactive) proof of correctness of the deanonymization. We also define an assertion LDeAnProof() capturing what is proved by the deanonymization transcript.

LDeanonymizable$(\text{DeAnOrgKeys}_{D}, \text{Transcript}_{UV}, \text{OrgNym}_{UI})$,
DODeAnonLocalWProof$(\text{DeAnOrgKeys}_{D}, \text{Transcript}_{UV})$ returns $\text{DeAnTranscript}_{D_{UV}}$
$\rightarrow$ LDeAnProof$(\text{DeAnTranscript}_{D_{UV}}, \text{Transcript}_{UV}, \text{OrgNym}_{UI})$.

If the above LDeanonymizable() assertion holds, then the transcript $\text{DeAnTranscript}_{D_{UV}}$ returned by the DODeAnonLocalWProof() invocation satisfies the LDeAnProof() assertion. The latter implies that, with $\text{DeAnTranscript}_{D_{UV}}$, any party can be convinced that Transcript_{UV} indeed contained the encryption of OrgNym_{UI} contained in Transcript_{UV}. The double-subscript notation in Transcript_{UV} indicates that D deanonymizes a transcript Transcript_{UV}.

5.2 Global Deanonymization

Global deanonymization works quite differently than local deanonymization. It is based on the fact that U carried out a root nym registration with Root Authority R. We assume that the identity (or, at least, the credential issuing public key) of R is known to all the organizations in the system, e.g., by making it part of the system parameters. Alternatively, R's issuing public key can be made an additional parameter of the GlobalDeAnData structure.

Consider the case where V verifies a credential Cred_{UI} issued by I with global deanonymization enabled.

If the verification is successful, the resulting transcript Transcript_{UV} should contain $EV_D(\text{OrgNym}_{UR})$, a verifiable encryption of U's registered root nym with D's public deanonymization key DPKey_D. If U's nym with R is indeed a root nym of which R knows a provable mapping to an external certificate, and U's nym with I is correctly linked to that root nym (is related to the same UserSecret_U), this is indeed the case.

We can represent this as follows:

$\texttt{RootNym}(\text{UserSecret}_U, \text{OrgNym}_{UR}, \text{UserNym}_{UR})$,
$\texttt{Nym}(\text{UserSecret}_U, \text{OrgNym}_{UI}, \text{UserNym}_{UI})$,
$\texttt{Cred}(\text{OrgNym}_{UI}, \text{UserCred}_{UI}, \text{CredInfo}_{UI})$,
$\texttt{ShowTranscript}(\text{Transcript}_{UV}, \text{UserCred}_{UI}, \text{CredShowInfo}_{UV}, \text{CredShowFeatures}_{UV}, \text{OrgNym}_{UV}, \text{Msg}_{UV})$,
$\text{CredShowFeatures}_{UV_I}.\text{GlobalDeAnData.DPKey} = \text{DeAnOrgKeys}_D.\text{DPKey}$
$\rightarrow \texttt{GDeanonymizable}(\text{DeAnOrgKeys}_D, \text{Transcript}_{UV}, \text{OrgNym}_{UR})$.

The $\texttt{GDeanonymizable}()$ assertion expresses that, if $\textsf{DODeAnonGlobal}()$ is called on Transcript_{UV} with DeAnOrgKeys_D, it will return OrgNym_{UR}.

Of course, in order to then derive an external identity from OrgNym_{UR}, the root nym also should have been registered using the external authentication:

$\texttt{SignedNymProof}(\text{SIG}_{UR}, \text{Cert}_{CA-U}, \text{OrgNym}_{UR}, \text{Msg}_{UR})$

The above set of assertions could, in principle, be realized by the corresponding protocol executions: root nym registration ($\texttt{RootNym}()$ and $\texttt{SignedNymProof}()$ assertions), nym registration with I ($\texttt{Nym}()$), credential issuing by I ($\texttt{Cred}()$) and showing of the latter credential ($\texttt{ShowTranscript}()$). However, the result of $\textsf{DODeAnonGlobal}()$ depends on OrgNym_{UI} being formed correctly w.r.t OrgNym_{UR}; neither I nor V can rely on this relationship without the issuing and verification of a root credential.

The following series of exchanges then realizes the above assertions while also verifying the relationship between OrgNym_{UR} and OrgNym_{UI}.

In this example, we directly represent the execution of an interactive protocol by means of the resulting assertion, and introduce shorthand notations for individual assertions. E.g., $\texttt{RootNym}_{UR}$ is a shorthand notation for a $\texttt{RootNym}$ assertion resulting from a root nym registration between U and R.

$$\begin{aligned}
U \leftrightarrow R: \quad & \texttt{RootNym}_{UR} = \texttt{RootNym}(\text{UserSecret}_U, \text{OrgNym}_{UR}, \text{UserNym}_{UR});\\
& \texttt{SignedNymProof}(\text{SIG}_{UR}, \text{Cert}_{CA-U}, \text{OrgNym}_{UR}, \text{Msg}_{UR});\\
& R \text{ stores } \{\text{OrgNym}_{UR}, \text{SIG}_{UR}, \text{Cert}_{CA-U}, \text{Msg}_{UR}\}\\
U \leftrightarrow R: \quad & \texttt{Cred}_{UR} = \texttt{Cred}(\text{OrgNym}_{UR}, \text{UserCred}_{UR}, \text{CredInfo}_{UR})\\
U \leftrightarrow I: \quad & \texttt{Nym}_{UI} = \texttt{Nym}(\text{UserSecret}_U, \text{OrgNym}_{UI}, \text{UserNym}_{UI})\\
U \leftrightarrow I: \quad & \texttt{ShowTranscript}_{UI_R} = \texttt{ShowTranscript}(\text{Transcript}_{UI_R}, \text{UserCred}_{UR}, \text{CredShowInfo}_{UI_R},\\
& \qquad \text{CredShowFeatures}_{UI_R}, \text{OrgNym}_{UI}, \text{Msg}_{UI_R}),\\
& \text{with CredShowFeatures}_{UI_R}.\text{RelNym} = \texttt{true}\\
& \text{and CredShowInfo}_{UI_R}.\text{IPKey} = \text{IPKey}_R\\
U \leftrightarrow I: \quad & \texttt{Cred}_{UI} = \texttt{Cred}(\text{OrgNym}_{UI}, \text{UserCred}_{UI}, \text{CredInfo}_{UI})\\
U \leftrightarrow V: \quad & \texttt{ShowTranscript}_{UV_I} = \texttt{ShowTranscript}(\text{Transcript}_{UV_I}, \text{UserCred}_{UI}, \text{CredShowInfo}_{UV_I},\\
& \qquad \text{CredShowFeatures}_{UV_I}, \texttt{null}, \text{Msg}_{UV_I}),\\
& \text{with CredShowFeatures}_{UV_I}.\text{GlobalDeAnData.DPKey} = \text{DPKey}_D
\end{aligned}$$

In a first exchange with R, U registers a root nym with R, resulting in the $\texttt{RootNym}()$ and $\texttt{SignedNymProof}()$ assertions; R stores the $\{\text{SIG}_{UR}, \text{Cert}_{CA-U}, \text{OrgNym}_{UR}, \text{Msg}_{UR}\}$ relationship. In a second exchange with R, U then establishes a root credential $\texttt{Cred}_{UR}$ that will allow him to convince I of the linking between a valid root nym and the nym it will later share with I. In the following two exchanges with I, U establishes a nym $\texttt{Nym}_{UI}$ with I and shows his root credential relative to the newly established $\texttt{Nym}_{UI}$ (the double-subscript notation $\texttt{ShowTranscript}_{UI_R}$ captures the fact that U shows to I a credential issued by R). This convinces I that $\texttt{Nym}_{UI}$ is related to the nym on which the root credential was issued; if it was a correctly formed root nym, then $\texttt{Nym}_{UI}$ is derived from it. In the third exchange with I, U then obtains a credential $\texttt{Cred}_{UI}$ on $\texttt{Nym}_{UI}$. U finally shows the latter credential to V, using global deanonymization with DPKey_D.

In the above scenario, V only directly verifies the credential $\texttt{Cred}_{UI}$; V cannot verify that U has a well-formed root nym with R and that the nym that U shares with I is correctly derived from that root nym. V thus has to rely on I and R for the following:

- V relies on R for issuing a root credential only on a well-formed root nym (of which it knows a provable mapping with an external legal entity); in case R issues credentials other than such root credentials, one expects R to use a dedicated key $iskey_R$ for issuing root credentials (on root nyms) and (an)other key(s) ISKey'_R for other credentials;
- V relies on I to verify that U has registered a nym with R, and that the nym $\texttt{Nym}_{UI}$ on which I will issue $\texttt{Cred}_{UI}$ is derived from that (root) nym; I does this by verifying the root credential relative to $\texttt{Nym}_{UI}$ in $\texttt{ShowTranscript}_{UI_R}$.

Alternatively, V can choose not to trust I for verifying the linking between the nyms, and to verify the linking itself. This now requires that U share a nym with V, and that U shows both credentials ($\texttt{Cred}_{UR}$ and $\texttt{Cred}_{UI}$) to V relative to this nym:

$$\begin{aligned}
U \leftrightarrow R:\quad & \texttt{RootNym}_{UR} = \texttt{RootNym}(\text{UserSecret}_U, \text{OrgNym}_{UR}, \text{UserNym}_{UR});\\
& \texttt{SignedNymProof}(\text{SIG}_{UR}, \text{Cert}_{CA-U}, \text{OrgNym}_{UR}, \text{Msg}_{UR});\\
& R \text{ stores } \{\text{OrgNym}_{UR}, \text{SIG}_{UR}, \text{Cert}_{CA-U}, \text{Msg}_{UR}\}\\
U \leftrightarrow R:\quad & \texttt{Cred}_{UR} = \texttt{Cred}(\text{OrgNym}_{UR}, \text{UserCred}_{UR}, \text{CredInfo}_{UR})\\
U \leftrightarrow I:\quad & \texttt{Nym}_{UI} = \texttt{Nym}(\text{UserSecret}_U, \text{OrgNym}_{UI}, \text{UserNym}_{UI})\\
U \leftrightarrow I:\quad & \texttt{Cred}_{UI} = \texttt{Cred}(\text{OrgNym}_{UI}, \text{UserCred}_{UI}, \text{CredInfo}_{UI})\\
U \leftrightarrow V:\quad & \texttt{Nym}_{UV} = \texttt{Nym}(\text{UserSecret}_U, \text{OrgNym}_{UV}, \text{UserNym}_{UV})\\
U \leftrightarrow V:\quad & \texttt{ShowTranscript}_{UV_R} = \texttt{ShowTranscript}(\text{Transcript}_{UV_R}, \text{UserCred}_{UR}, \text{CredShowInfo}_{UV_R},\\
& \qquad \text{CredShowFeatures}_{UV_R}, \text{OrgNym}_{UV}, \text{Msg}_{UV_R})\\
U \leftrightarrow V:\quad & \texttt{ShowTranscript}_{UV_I} = \texttt{ShowTranscript}(\text{Transcript}_{UV_I}, \text{UserCred}_{UI}, \text{CredShowInfo}_{UV_I},\\
& \qquad \text{CredShowFeatures}_{UV_I}, \text{OrgNym}_{UV}, \text{Msg}_{UV_I}),\\
& \text{with CredShowFeatures}_{UV_I}.\text{GlobalDeAnData.DPKey} = \text{DPKey}_D
\end{aligned}$$

We have assumed that I does not have its own requirement to verify a root credential before issuing $\texttt{Cred}_{UI}$; if such a verification is a condition, an additional show of $\texttt{Cred}_{UR}$ to I is necessary.

By verifying both credentials relative to OrgNym_{UV}, V verifies that the nyms on which the credentials are issued are related. V now only has to rely on R to issue root credentials only on correctly formed root nyms.

5.2.1 Global Deanonymization With Proof of Correctness

Similar as for local deanonymization, we can define a new operation DODeAnonGlobalWProof() and a new assertion GDeAnProof():

$\texttt{GDeanonymizable}(\text{DeAnOrgKeys}_D, \text{Transcript}_{UV}, \text{OrgNym}_{UR})$,
$\text{DODeAnonGlobalWProof}(\text{DeAnOrgKeys}_D, \text{Transcript}_{UV})$ returns $\text{DeAnTranscript}_{D_{UV}}$
$\rightarrow \texttt{GDeAnProof}(\text{DeAnTranscript}_{D_{UV}}, \text{Transcript}_{UV}, \text{OrgNym}_{UR})$.

The reasoning is the same as for the case of local deanonymization: if the GDeanonymizable() assertion holds, then the transcript $\text{DeAnTranscript}_{D_{UV}}$ returned by the DODeAnonGlobalWProof() invocation will satisfy the GDeAnProof() assertion.

The GDeAnProof() assertion implies that $\text{DeAnTranscript}_{D_{UV}}$ not only contains the result of the deanonymization (OrgNym_{UR}) but also a proof of correctness of the deanonymization.

5.3 Double-Spending Detection

$\texttt{ShowTranscript}(\text{Transcript}_{UV}, \text{UserCred}_{UI}, \text{CredShowInfo}_{UV}, \text{CredShowFeatures}_{UV}, \text{OrgNym}_{UV}, \text{Msg}_{UV})$,
$\texttt{Cred}(\text{OrgNym}_{UI}, \text{UserCred}_{UI}, \text{CredInfo}_{UI})$,
$\text{CredInfo}_{UI}.\text{MultiShow} = \texttt{false}$,
$\rightarrow \texttt{DblSpDetectable}(\text{OrgNym}_{UI}, \text{UserCred}_{UI}, \text{Transcript}_{UV})$.

The showing of a one-show credential results in a DblSpDetectable() assertion; this assertion implies that a second show of the same credential UserCred_{UI} causes OCheckDblSp(), when called on both transcripts with the correct issuer keys, to reveal OrgNym_{UI} on which UserCred_{UI} is issued.

5.3.1 Double-Spending Detection With Proof of Correctness

Also a result of double-spending detection can be proved correct, in this case by the issuer who detects double-spending, using the primitive OCheckDblSpWProof():

$\text{CredInfo}_{UI}.\text{IPKey} = \text{IssuerKeys}_I.\text{IPKey}$,
$\texttt{DblSpDetectable}(\text{OrgNym}_{UI}, \text{UserCred}_{UI}, \text{Transcript}_{UV_1})$,
$\texttt{DblSpDetectable}(\text{OrgNym}_{UI}, \text{UserCred}_{UI}, \text{Transcript}_{UV_2})$,
$\text{OCheckDblSpWProof}(\text{IssuerKeys}_I, \text{Transcript}_{UV_1}, \text{Transcript}_{UV_2})$ returns $\text{DblSpTranscript}_{I_{UI}}$
$\rightarrow \texttt{DblSpProof}(\text{DblSpTranscript}_{I_{UI}}, \text{Transcript}_{UV_1}, \text{Transcript}_{UV_2}, \text{OrgNym}_{UI})$.

$\text{DblSpTranscript}_{I_{UI}}$ is the transcript of the double-spending detection; it includes OrgNym_{UI}. The double-subscript notation in $\text{DblSpTranscript}_{I_{UI}}$ implies that I verifies double-spending of a credential $\texttt{Cred}_{UI}$.

6 Assertions on Linkability

6.1 Anonymity, Linkability and Identifiability

In the previous sections, we have focused on a description and features of the *idemix* protocols which allows to show fulfillment of an organization's requirements. Assertions on credential shows can fulfill requirements of authentication (what was proved?); provable assertions about credential shows, deanonymization and double-spending detection, together with the provability of the relationship between a nym and a user through signed or root nym registration, can fulfill requirements of user accountability.

When designing applications, an important requirement by the user is also unidentifiability, realized through anonymity and unlinkability of his actions. If a user requires that a certain action requiring a credential show be unconditionally unidentifiable, this clearly excludes a credential show with a global deanonymization option.

In Section 2.4, we defined anonymity as being a protocol-intrinsic property, and unidentifiability to result from anonymity in the absence of factors causing linkability with an identity. However, when discussing a specific system such as *idemix*, we have to concretely define anonymity properties of its protocols. E.g., signed nym registration can certainly be considered to be an identifiable action, if the key and certificate used for signing are identifiable. However, even if our discussions of accountability assume Cert_{CA-U} to be identifiable, one can consider scenarios where signed nym registration uses a pseudonym certificate. Do we then define the signed nym registration protocol to be anonymous or not?

For the sake of clarity and consistency, we define all the *idemix* interactive protocols to be intrinsically anonymous. According to this definition, also signed and root nym registration are anonymous; identifiability of the user's certificate and signature key then makes the registered nym identifiable by introducing a linking between the nym and the certificate. Similarly, the showing of a credential, even with the global deanonymization option set, is anonymous; identifiability results if global deanonymization is applied and the user's external certificate is identifiable.

In this section, we describe assertions about linkability of nyms, credentials and credential shows related to executions of the various primitives; these allow to reason about the identifiability of a user's actions as a consequence of a linkability to identifiable items. For each of the *idemix* protocols in Section 4, we define its linkability characteristics by expressing whether nyms or the user's identity are linkable to the result of the protocol execution; this linkability can be towards the organization involved in the protocol (e.g., issuing or verifying organization) or towards another organization (e.g., a deanonymizing organization).

We define a new assertion:

$$\texttt{Linked}(\text{Org}, \text{Cond}, \text{Value}_1, \text{Value}_2)$$

expressing that Org can link the values Value_1 and Value_2. Cond is a condition that may restrict Org's right to do so. E.g., Org may be a deanonymizing organization that is able to link a transcript to a nym but should only do so under a certain condition.

Intuitively, if two organizations each have their own linkability knowledge, then a collaborating set of both organizations combines that knowledge:

$\texttt{Linked}(\text{Org}_1, \text{Cond}_1, \text{Value}_{1_1}, \text{Value}_{1_2})$,
$\texttt{Linked}(\text{Org}_2, \text{Cond}_2, \text{Value}_{2_1}, \text{Value}_{2_2})$
$\rightarrow \texttt{Linked}(\{\text{Org}_1 \text{ AND } \text{Org}_2\}, \text{Cond}_1, \text{Value}_{1_1}, \text{Value}_{1_2})$,
$\rightarrow \texttt{Linked}(\{\text{Org}_1 \text{ AND } \text{Org}_2\}, \text{Cond}_2, \text{Value}_{2_1}, \text{Value}_{2_2})$.

Also, linkability is symmetric:

$\texttt{Linked}(\text{Org}, \text{Cond}_1, \text{Value}_1, \text{Value}_2)$
$\rightarrow \texttt{Linked}(\text{Org}, \text{Cond}_1, \text{Value}_2, \text{Value}_1)$.

and transitive:

$\texttt{Linked}(\text{Org}, \text{Cond}_1, \text{Value}_1, \text{Value}_2)$,
$\texttt{Linked}(\text{Org}, \text{Cond}_2, \text{Value}_2, \text{Value}_3)$
$\rightarrow \texttt{Linked}(\text{Org}, \{\text{Cond}_1 \text{ AND } \text{Cond}_2\}, \text{Value}_1, \text{Value}_3)$.

The linkabilities we can express are the ones related to the choice of protocol and its options. Some examples:

- A credential show relative to a nym obviously links the transcript to this nym;
- A deanonymizable credential show makes a later linking possible of the transcript to a nym by the deanonymizing organization;
- Showing of a one-show credential makes a linking possible of the transcript to a nym by the credential issuer, if a second show transcript of the same credential exists;
- As part of showing a credential, a user may sign a message which is unique; this message may identify the user or make the credential show linkable to another transaction linked to the same message;
- A credential may have a unique attribute which the issuer can link to the nym it was issued on.

The showing of a credential, even if not relative to a nym and not deanonymizable, may thus become fully or partially identifiable if only one or a small set of users has this type of credential. One reason can be that the issuer purposely gave it some unique attribute, as discussed above. Another reason can be that the set of users owning and using a credential of the same type is (still) too small. In the first case, the identifiability is feature inherent in the credential and protocol definition and we will define its effect on linkability. In the second case, the linkability is dynamic and is a result of how the system is actually used; this cannot be predicted by knowledge of the protocols used.

Thus, there are hidden linkabilities which we cannot capture with our reasoning. The small user set owning a specific type of credential is one example; another example is traceability of communication which may identify a user in an otherwize unidentifiable *idemix* transaction. A hidden linkability may also exist if a user signs an identifying message which he believes not to be identifying.

The linkability factors that are not related to the choice of *idemix* protocols and their parameters have to be taken into account by good system design and deployment as well as user education. Anonymous communication should be available and used; a credential of a specific type should not be used unless the group of users owning and using such a credential is large enough; and user agents as well as organizations should help users to avoid including identifiable information in messages if this identifiability is not a design feature of the system; if, on the contrary, it is an intended feature, this should be visible to users.

Thus, using good system design and deployment, hidden linkabilities may be excluded. In such a system, linkability assertions as we discuss here are a meaningful way of expressing or discovering potential identifiability factors.

In the following paragraphs, we now derive linkability assertions for each of the *idemix* protocols. When designing a specific application, we can then translate unidentifiability requirements into requirements about the absence of linkabilities. We will illustrate this with our example application in Section 8.

We do not attempt to develop a logic of linkability; the linkability assertions as we define them are merely intended to help us expressing, in an intuitive way, direct traceability of a user's identity or nyms through various *idemix* protocol executions. Also, we do not attempt to make any probabilistic statements about linkability; e.g., a credential attribute is either unique or not; in the former case, its showing can be linked to its issuing; in the latter case, it cannot (however large or small the set of users using a credential with this attribute). A statement about 'unconditional unidentifiability' of an action or set of actions thus only claims that according to the associated linkability assertions as defined in the following, no linkability to an identifiable item exists.

Note that the `Linked`() assertion does not distinguish between linkabilities which are provable and linkabilities which are not. Although most `Linked`() assertions will be related to provable assertions, we also want to capture linkabilities which are not provable. E.g., if a credential was shown relative to a nym, then the fact that the show transcript is linked to this nym is provable. But, if an organization can link a credential to a nym because the credential type (CredInfo) is unique, this linkability is not provable according to any of our provable assertions. Whether or not the linking can be proved to a third party depends on the organization being able to convince a third party that no other credential with the same CredInfo was issued.

6.2 Nym Registration

From establishment of the $\texttt{Nym}()$ assertion as a result of the protocol execution in Section 4.1, I's only result is OrgNym_{UI} and no linkability towards I results:

$\texttt{Nym}(\mathrm{UserSecret}_U, \mathrm{OrgNym}_{UI}, \mathrm{UserNym}_{UI})$
$\rightarrow -.$

6.3 Signed Nym Registration

Signed nym registration as in Section 4.2 results in a $\texttt{Nym}$ as well as a $\texttt{SignedNymProof}()$ assertion. The $\texttt{SignedNymProof}()$ assertion creates a linkability, by I, between the established nym and the certificate used for signing:

$\texttt{SignedNymProof}(\mathrm{SIG}_{UI}, \mathrm{Cert}_{CA-U}, \mathrm{OrgNym}_{UI}, \mathrm{Msg}_{UI})$
$\rightarrow \texttt{Linked}(I, \texttt{null}, \mathrm{OrgNym}_{UI}, \mathrm{Cert}_{CA-U}).$

Cert_{CA-U} here stands for the identity of the user: when something is linked to Cert_{CA-U}, we will say that it is linked to the user.

If Msg_{UI} is a generic message signed by all the users obtaining this type of nym with I, this is the only resulting linkability. However, if Msg is a message unique to this instance of the nym registration protocol, an additional linkability results. The assertion $\texttt{Unique}()$ expresses the fact that a value is unique:

$\texttt{SignedNymProof}(\mathrm{SIG}_{UI}, \mathrm{Cert}_{CA-U}, \mathrm{OrgNym}_{UI}, \mathrm{Msg}_{UI}),$
$\texttt{Unique}(\mathrm{Msg}_{UI})$
$\rightarrow \texttt{Linked}(I, \texttt{null}, \mathrm{OrgNym}_{UI}, \mathrm{Cert}_{CA-U}),$
$\rightarrow \texttt{Linked}(I, \texttt{null}, \mathrm{Msg}_{UI}, \mathrm{Cert}_{CA-U}).$

By transitivity, also a linking between Msg_{UI} and OrgNym_{UI} can be derived.

6.4 Root Nym Registration

Root nym registration results in $\texttt{RootNym}$ and $\texttt{SignedNymProof}$ assertions. The linkability effect of $\texttt{SignedNymProof}$ is the one described in Section 6.3; $\texttt{RootNym}$, as $\texttt{Nym}$, does not result in any linkability:

$\texttt{RootNym}(\mathrm{UserSecret}_U, \mathrm{OrgNym}_{UR}, \mathrm{UserNym}_{UR})$
$\rightarrow -.$

6.5 Issuing of a Non-Unique Credential

Anonymous credential issuing assumes that $\mathrm{CredInfo}_{UI}$ is a generic credential type issued on a large set of different nyms. In this case, it is still important for I to know he issued a credential of type $\mathrm{CredInfo}_{UI}$ on OrgNym_{UI} (e.g., for revocation purposes); however, as there are many credentials of the same type, the credential issuing does not result in any linkability information, i.e., a later use of the credential cannot be linked back to the specific issuing instance (and the nym OrgNym_{UI}) it was issued on:

$\texttt{Cred}(\mathrm{OrgNym}_{UI}, \mathrm{UserCred}_{UI}, \mathrm{CredInfo}_{UI}),$
$\neg\texttt{Unique}(\mathrm{CredInfo}_{UI})$
$\rightarrow -.$

6.6 Issuing of a Unique Credential

If the CredInfo type of a credential is unique, then the linking between the nym and the type (CredInfo) is relevant:

$\texttt{Cred}(\text{OrgNym}_{UI}, \text{UserCred}_{UI}, \text{CredInfo}_{UI})$,
$\texttt{Unique}(\text{CredInfo}_{UI})$
$\rightarrow \texttt{Linked}(I, \texttt{null}, \text{OrgNym}_{UI}, \text{CredInfo}_{UI})$.

6.7 Unconditionally Unidentifable Showing of a Credential

We refer to the protocol discussed in Sections 4.5. If the credential is not shown relative to a nym, then Transcript_{UV} may not contain any identifiable parameters. However, linkability can still be introduced by the use of either:

- local or global deanonymization (Section 6.10);
- a unique parameter such as a unique CredShowInfo, allowing to link the show transcript to a specific credential with a matching CredInfo; or a unique Msg, allowing to link the show transcript to another exchange related to Msg (Section 6.8).

If neither CredShowInfo_{UV} nor Msg_{UV} are unique, a credential can be shown in an unconditionally unidentifable way if the show is neither relative to a nym nor deanonymizable:

$\texttt{ShowTranscript}(\text{Transcript}_{UV}, \text{UserCred}_{UI}, \text{CredShowInfo}_{UV}, \text{CredShowFeatures}_{UV}, \texttt{null}, \text{Msg}_{UV})$,
$\neg\texttt{Unique}(\text{CredShowInfo}_{UV}), \neg\texttt{Unique}(\text{Msg}_{UV})$,
$\text{CredShowFeatures}_{UV} = \{\texttt{false}, \texttt{null}, \texttt{null}\}$
$\rightarrow -$.

6.8 Showing a Credential With Unique Parameters

If either CredShowInfo_{UV} or Msg_{UV} is unique, this has to be taken into account by an additional linkability. However, this is relevant only if the unique feature in CredShowInfo_{UV} (or Msg_{UV}) also occurs in another linkability statement.

$\texttt{ShowTranscript}(\text{Transcript}_{UV}, \text{UserCred}_{UI}, \text{CredShowInfo}_{UV}, \text{CredShowFeatures}_{UV}, \text{OrgNym}_{UV}, \text{Msg}_{UV})$,
$\texttt{Unique}(\text{CredShowInfo}_{UV})$
$\rightarrow \texttt{Linked}(V, \texttt{null}, \text{Transcript}_{UV}, \text{CredShowInfo}_{UV})$.

$\texttt{ShowTranscript}(\text{Transcript}_{UV}, \text{UserCred}_{UI}, \text{CredShowInfo}_{UV}, \text{CredShowFeatures}_{UV}, \text{OrgNym}_{UV}, \text{Msg}_{UV})$,
$\texttt{Unique}(\text{Msg}_{UV})$
$\rightarrow \texttt{Linked}(V, \texttt{null}, \text{Transcript}_{UV}, \text{Msg}_{UV})$.

This linkability effect has to be combined with any other linkablities resulting from showing a credential relative to a nym, showing a credential in a deanonymizable way, or showing a one-show credential.

6.9 Showing a Credential Relative to a Nym

The showing of a transcript relative to a nym makes the nym and the transcript linkable towards the verifier:

$\texttt{ShowTranscript}(\text{Transcript}_{UV}, \text{UserCred}_{UI}, \text{CredShowInfo}_{UV}, \text{CredShowFeatures}_{UV}, \text{OrgNym}_{UV}, \text{Msg}_{UV})$,
$\text{CredShowFeatures}_{UV}.\text{RelNym} = \texttt{true}$
$\rightarrow \texttt{Linked}(V, \texttt{null}, \text{Transcript}_{UV}, \text{OrgNym}_{UV})$.

6.10 Showing a Credential With Deanonymizaton

For all previous types of showing a credential, additional linking assertions result from the use of a deanonymization option:

For local deanonymization:

$\texttt{LDeanonymizable}(\text{DeAnOrgKeys}_D, \text{Transcript}_{UV}, \text{OrgNym}_{UI})$,
Transcript_{UV}.CredShowFeatures.LocalDeAnData.DPKey = $\text{DeAnOrgKeys}_D.\textit{dpkey}$,
Transcript_{UV}.CredShowFeatures.LocalDeAnData.DeAnCondition = $\text{DeAnCondition}_{UV}$
$\rightarrow \texttt{Linked}(D, \text{DeAnCondition}_{UV}, \text{OrgNym}_{UI}, \text{Transcript}_{UV})$.

For global deanonymization:

$\texttt{GDeanonymizable}(\text{DeAnOrgKeys}_D, \text{Transcript}_{UV}, \text{OrgNym}_{UR})$,
Transcript_{UV}.CredShowFeatures.GlobalDeAnData.DPKey = $\text{DeAnOrgKeys}_D.\textit{dpkey}$,
Transcript_{UV}.CredShowFeatures.GlobalDeAnData.DeAnCondition = $\text{DeAnCondition}_{UV}$
$\rightarrow \texttt{Linked}(D, \text{DeAnCondition}_{UV}, \text{OrgNym}_{UR}, \text{Transcript}_{UV})$.

6.11 Showing a One-Show Credential

When a one-show credential is shown, the transcript is conditionally linked to the nym on which the credential was issued - the condition being double-spending of the credential:

$\texttt{DblSpDetectable}(\text{OrgNym}_{UI}, \text{UserCred}_{UI}, \text{Transcript}_{UV_1})$,
CredInfo_{UI}.IPKey = IssuerKeys_I.IPKey
$\rightarrow \texttt{Linked}(I, \texttt{dblspent}, \text{OrgNym}_{UI}, \text{Transcript}_{UV})$.

The `dblspent` condition indicates that the presence of another transcript $\text{Transcript}'_{UV}$ satisfying the `DblSpDetectable`() assertion will materialize the linking towards I.

7 Additional Procedures and Functionality

7.1 More On Global Deanonymization

In Section 5.2, we described how global deanonymization can be realized using a specific option in the credential show protocol together with the showing of the root credential. Here, we discuss two additional possibilities for achieving a similar functionality by using only the 'local deanonymization' protocol option.

We refer again to an exchange such as the one in Section 5.2. In order for a verifier V to have a guarantee of obtaining Cert_{CA-U} under condition $\text{DeAnCondition}_{UV_I}$, other possiblities exist:

- It can be achieved by cooperation between V and I both requiring local deanonymization in subsequent show protocols, as shown in the exchange in Figure 6.

  When verifying $\texttt{Cred}_{UI}$ in the last exchange, V requires only local deanonymization (with DPKey_D and $\text{DeAnCondition}_{UV_I}$), and trusts I to have verified $\texttt{Cred}_{UR}$ before issuing $\texttt{Cred}_{UI}$. This verification ($\texttt{ShowTranscript}_{UI_R}$) was also done using local deanonymization, probably with the same DPKey_D and a related condition $\text{DeAnCondition}_{UI_R}$; and was relative to OrgNym_{UI}.

  If $\text{DeAnCondition}_{UV_I}$ is fulfilled, $\texttt{ShowTranscript}_{UV_I}$ can be (locally) deanonymized by D to reveal OrgNym_{UI}:

$$\mathsf{DODeAnonLocal}(\text{DeAnOrgKeys}_D, \text{Transcript}_{UV_I}) \text{ returns } \text{OrgNym}_{UI}$$

$$\begin{array}{ll}
U \leftrightarrow R: & \texttt{RootNym}_{UR} = \texttt{RootNym}(\mathrm{UserSecret}_U, \mathrm{OrgNym}_{UR}, \mathrm{UserNym}_{UR}); \\
 & \texttt{SignedNymProof}(\mathrm{SIG}_{UR}, \mathrm{Cert}_{CA-U}, \mathrm{OrgNym}_{UR}, \mathrm{Msg}_{UR}); \\
 & R \text{ stores}\{\mathrm{OrgNym}_{UR}, \mathrm{SIG}_{UR}, \mathrm{Cert}_{CA-U}, \mathrm{Msg}_{UR}\} \\
U \leftrightarrow R: & \texttt{Cred}_{UR} = \texttt{Cred}(\mathrm{OrgNym}_{UR}, \mathrm{UserCred}_{UR}, \mathrm{CredInfo}_{UR}) \\
U \leftrightarrow I: & \texttt{Nym}_{UI} = \texttt{Nym}(\mathrm{UserSecret}_U, \mathrm{OrgNym}_{UI}, \mathrm{UserNym}_{UI}) \\
U \leftrightarrow I: & \texttt{ShowTranscript}_{UI_R} = \texttt{ShowTranscript}(\mathrm{Transcript}_{UI_R}, \mathrm{UserCred}_{UR}, \mathrm{CredShowInfo}_{UI_R}, \\
 & \qquad \mathrm{CredShowFeatures}_{UI_R}, \mathrm{OrgNym}_{UI}, \mathrm{Msg}_{UI_R}) \\
 & \text{with } \mathrm{CredShowFeatures}_{UI_R}.\mathrm{LocalDeAnData} = \{\mathrm{DPKey}_D, \mathrm{DeAnCondition}_{UI_R}\} \\
 & \text{and } \mathrm{CredShowFeatures}_{UI_R}.\mathrm{RelNym} = \texttt{true}; \\
 & I \text{ stores } \{\mathrm{OrgNym}_{UI}, \mathrm{Transcript}_{UI_R}\} \\
U \leftrightarrow I: & \texttt{Cred}_{UI} = \texttt{Cred}(\mathrm{OrgNym}_{UI}, \mathrm{UserCred}_{UI}, \mathrm{CredInfo}_{UI}) \\
U \leftrightarrow V: & \texttt{ShowTranscript}_{UV_I} = \texttt{ShowTranscript}(\mathrm{Transcript}_{UV_I}, \mathrm{UserCred}_{UI}, \mathrm{CredShowInfo}_{UV_I}, \\
 & \qquad \mathrm{CredShowFeatures}_{UV_I}, \texttt{null}, \mathrm{Msg}_{UV_I}), \\
 & \text{with } \mathrm{CredShowFeatures}_{UV_I}.\mathrm{LocalDeAnData} = \{\mathrm{DPKey}_D, \mathrm{DeAnCondition}_{UV_I}\}
\end{array}$$

Figure 6: Global Deanonymization Using Local Deanonymization: V and I Cooperating

The fact that $\texttt{ShowTranscript}_{UI_R}$ was relative to OrgNym_{UI} ensures to I(and allows I to prove) that $\mathrm{Transcript}_{UI_R}$ is related to OrgNym_{UI}.

$$\texttt{SigProof}(\mathrm{Transcript}_{UI_R}, \mathrm{CredShowInfo}_{UI_R}, \mathrm{CredShowFeatures}_{UI_R}, \mathrm{OrgNym}_{UI}, \mathrm{Msg}_{UI_R})$$

Thus, with cooperation from I, who can link OrgNym_{UI} to $\mathrm{Transcript}_{UI_R}$,$\mathrm{Transcript}_{UI_R}$ can then be (locally) deanonymized by D to reveal OrgNym_{UR}:

$$\mathsf{DODeAnonLocal}(\mathrm{DeAnOrgKeys}_D, \mathrm{Transcript}_{UI_R}) \text{ returns } \mathrm{OrgNym}_{UR}$$

For the latter deanonymization to be enabled upon V's deanonymization condition being fulfilled, $\mathrm{DeAnCondition}_{UI_R}$ and $\mathrm{DeAnCondition}_{UV_I}$ should be related: $\mathrm{DeAnCondition}_{UI_R}$ has to be fulfilled if $\mathrm{DeAnCondition}_{UV_I}$ is fulfilled.

- It can be realized by V alone: V itself demands a locally deanonymizable show $\texttt{ShowTranscript}_{UV_R}$ of $\texttt{Cred}_{UR}$ in addition to a show $\texttt{ShowTranscript}_{UV_I}$ of $\texttt{Cred}_{UI}$. For V to verify the linking, $\texttt{ShowTranscript}_{UV_R}$ then has to be relative to a nym shared between U and V. Consequently, this requires an additional nym registration between U and V. Figure 7 illustrates the procedure.

  By verifying that both $\texttt{Cred}_{UR}$ and $\texttt{Cred}_{UI}$ can be shown relative to OrgNym_{UV}, V verifies that the nyms on which both credentials are issued are related to the same UserSecret. Consequently, V is assured that decryption by D of $\texttt{ShowTranscript}_{UV_R}$ wil reveal the OrgNym_{UR} of the user who showed both credentials. When deanonymization of $\texttt{ShowTranscript}_{UV_I}$ is required, V now only has to ask D for (local) deanonymization of $\texttt{ShowTranscript}_{UV_R}$:

$$\mathsf{DODeAnonLocal}(\mathrm{DeAnOrgKeys}_D, \mathrm{Transcript}_{UV_R}) \text{ returns } \mathrm{OrgNym}_{UR}$$

$$
\begin{array}{ll}
U \leftrightarrow R: & \texttt{RootNym}_{UR} = \texttt{RootNym}(\mathrm{UserSecret}_{U}, \mathrm{OrgNym}_{UR}, \mathrm{UserNym}_{UR}); \\
 & \texttt{SignedNymProof}(\mathrm{SIG}_{UR}, \mathrm{Cert}_{CA-U}, \mathrm{OrgNym}_{UR}, \mathrm{Msg}_{UR}); \\
 & R \text{ stores}\{\mathrm{OrgNym}_{UR}, \mathrm{Cert}_{CA-U},, \mathrm{Msg}_{UR}\} \\
U \leftrightarrow R: & \texttt{Cred}_{UR} = \texttt{Cred}(\mathrm{OrgNym}_{UR}, \mathrm{UserCred}_{UR}, \mathrm{CredInfo}_{UR}) \\
U \leftrightarrow I: & \texttt{Nym}_{UI} = \texttt{Nym}(\mathrm{UserSecret}_{U}, \mathrm{OrgNym}_{UI}, \mathrm{UserNym}_{UI}) \\
U \leftrightarrow I: & \texttt{Cred}_{UI} = \texttt{Cred}(\mathrm{OrgNym}_{UI}, \mathrm{UserCred}_{UI}, \mathrm{CredInfo}_{UI}) \\
U \leftrightarrow V: & \texttt{Nym}_{UV} = \texttt{Nym}(\mathrm{UserSecret}_{U}, \mathrm{OrgNym}_{UV}, \mathrm{UserNym}_{UV}) \\
U \leftrightarrow V: & \texttt{ShowTranscript}_{UV_R} = \texttt{ShowTranscript}(\mathrm{Transcript}_{UV_R}, \mathrm{UserCred}_{UR}, \mathrm{CredShowInfo}_{UV_R}, \\
 & \qquad\qquad\qquad\qquad\qquad \mathrm{CredShowFeatures}_{UV_R}, \mathrm{OrgNym}_{UV}, \mathrm{Msg}_{UV_R}) \\
 & \text{with } \mathrm{CredShowFeatures}_{UV_R}.\mathrm{LocalDeAnData} = \{\mathrm{DPKey}_{D}, \mathrm{DeAnCondition}_{UV_R}\} \\
U \leftrightarrow V: & \texttt{ShowTranscript}_{UV_I} = \texttt{ShowTranscript}(\mathrm{Transcript}_{UV_I}, \mathrm{UserCred}_{UI}, \mathrm{CredShowInfo}_{UV_I}, \\
 & \qquad\qquad\qquad\qquad\qquad \mathrm{CredShowFeatures}_{UV_I}, \mathrm{OrgNym}_{UV}, \mathrm{Msg}_{UV_I})
\end{array}
$$

Figure 7: Global Deanonymization Using Local Deanonymization: V Verifying Correct Linking of Nyms

7.2 Revocation

Traditional methods for certificate revocation are based on the fact that the relying party (the signature verifier) can compare a certificate's serial number or public key with items in a list of revoked (or, alternatively, valid) certificates. In an anonymous credential system, this type of revocation is impossible: the relying party has no way of distinguishing one user's credential from another user's credential of the same type.

Two methods can be considered for realizing revocation in the anonymous credential system. Using the first method (described in [3, 2]), when I wants to revoke a credential it issued on a nym OrgNym_{IU}, I simply changes its issuing public key IPKey_I to IPKey'_I, and advertises this change to relying parties and owners of credentials based on IPKey_I. All the credentials based on the old IPKey_I are now invalid, and each owner of a credential based on IPKey_I has to go through an interactive re-issuing protocol with the issuer to have their credential re-issued with the new key IPKey'_I. I simply does not re-issue the revoked credential. Revocation of a credential thus involves that the issuer revokes and replaces his issuing key; and information of a relying party does not consist of revocation lists but of the latest issuing organization's IPKey. In terms of efficiency for relying parties, this is quite advantageous, as they only need to keep track of this valid IPKey. In terms of efficiency for the issuer and other users in the system, the interactive re-issuing is quite inefficient. One can optimize the process for the other users. E.g., if U's credential based on IPKey_I was revoked, the other users need not be notified but are made aware of the revocation at the moment they want to show their own (invalidated) credential. At that point, they obtain the new issuer key (either from the verifier or from the issuer) and initiate the re-issuing process.

A second method, based on *dynamic accumulators*, is described in [8]. Using this method, an issuer revoking a credential also changes its public key, by changing a dynamic accumulator list of valid credentials, which is part of the public key. The new public key is again distributed to users and relying parties; every user can update (re-validate) their (invalidated) credential in a non-interactive computation, based on the information in the accumulator. As the revoked credential is not anymore part of the accumulator, the revoked credential cannot be re-validated.

Both methods assume that each credential issuer keeps track of which credentials it issued with which key and on which OrgNym. The second method can be realized more efficiently than the first method, especially because it does not involve an interactive re-issuing of credentials. In following sections, when discussing applications based on anonymous credentials, we will assume that one of the two mechanisms for revocation is in place. We assume the existence of a high-level primitive.

$$\mathsf{ORevokeCred}(\text{IssuerKeys, OrgNym, CredInfo})$$

As mentioned in Section 3, a party relying on verification of a credential is responsible for verifying the correctness of the issuer's public key IPKey_I in the CredShowInfo parameter of the $\mathsf{OVerifyCred}()$ invocation; the relying party should also verify that IPKey_I has not been revoked. This implies the existence of the appropriate mechanisms for relying parties to verify an issuer key's 'good' (non-revoked) state.

7.3 Certification

In Sections 3.2 and 3.3, we described how users can use an external certificate to securely link their root or other nym to an externally validated identity.

Organizations, as well, need to link their *idemix* keys to an external certification infrastructure. For organizations, several levels of certification exist: the organization, as a business entity (e.g., an on-line merchant), may have an externally certified public signature key, the private key of which it uses to issue self-certified *idemix* certificates for the keys needed in various transaction and communication roles. E.g., the on-line merchant may self-certify *idemix* public issuing and/or verification keys (IPKey and/or VPKey) corresponding to the ISKey/VSKey with which he issues/verifies *idemix* credentials; he may self-certify another signature key to realize authentication and key exchange for protecting communication channels used in *idemix* as well as non-*idemix* exchanges.

Certification of organizations' *idemix* keys and other keys in the system is necessary in order for relying parties (users and other organizations) to trust their authenticity and origin; it is also necessary for relying parties to be able to derive liabilities or accountability based on transactions and credentials with these keys. In the following section, we introduce an example application based on anonymous credentials; Section 9 illustrates the details of its certificate infrastructure and how it defines the liabilities and accountabilities of the various parties.

8 Designing an Application

When designing an actual application scenario, we start with various sets of requirements. Requirements of authentication related to an organization's authorization decisions (called *Rules* in [9]), can be translated into requirements on (assertions on) 'show transcripts'; by their relationship with the protocol primitives, these will in part determine which *idemix* credentials need to be shown in order to fulfill organizations' authentication requirements. Requirements of user accountability determine various other options such as signed or root nym registrations and deanonymization.

Users' unidentifiability requirements then further narrow down the set of acceptable solutions. Of course, a user may have unidentifiability requirements which collide with an organization's accountability requirements; in practice, we will assume that a user using a organization's service accepts this service's accountability requirements. Fulfilling the user's requirements then consists of making sure that these accountability requirements are fulfilled with 'minimal linkabilities'.

This section describes an application using the various *idemix* protocols. We show the design of the application based on its authentication, accountability and unidentifiability requirements using the assertions

- **Key Material:**
  - U (User): $\{\text{SSKey}_U, \text{Cert}_{CA-U}\}$, UserSecret_U
  - R (Root): $\{\text{SSKey}_R, \text{Cert}_{CA-R}\}$, $\{\text{ISKey}_R, \text{CSKey}_R, \text{Cert}_{R-R}\}$
  - K (Kiosk): $\{\text{SSKey}_K, \text{Cert}_{CA-K}\}$, $\{\text{ISKey}_K, \text{VSKey}_K, \text{CSKey}_K, \text{Cert}_{K-K}\}$
  - L (LostFound): $\{\text{SSKey}_L, \text{Cert}_{CA-L}\}$, $\{\text{VSKey}_L, \text{CSKey}_L, \text{Cert}_{L-L}\}$
  - D (DOrg): $\{\text{SSKey}_D, \text{Cert}_{CA-D}\}$, $\{\text{DSKey}_D, \text{CSKey}_{D_c}, \text{Cert}_{D-D}\}$

- **Root Nym Registration and Obtaining Root Credential:**

$$\mathbf{U} : \mathsf{URegRootNym()} \xleftrightarrow{\text{Comm}_{U\underline{R}_1} : \texttt{RootNym}_{UR}, \texttt{SignedNymProof}_{UR}} \mathbf{R} : \mathsf{ORegRootNym()}$$

$$R \text{ stores}\{\text{OrgNym}_{UR}, \text{SIG}_{UR}, \text{Cert}_{CA-U}\}$$

$$\mathbf{U} : \mathsf{UGetCred()} \xleftrightarrow{\text{Comm}_{U\underline{R}_2} : \texttt{Cred}_{UR}} \mathbf{R} : \mathsf{OIssueCred()}$$

$$R \text{ updates}\{\text{OrgNym}_{UR}, \text{SIG}_{UR}, \text{Cert}_{CA-U}, \text{CredInfo}_{UR}\}$$

- **Payment and Obtaining Kiosk Credential:**

$$\mathbf{U} : \mathsf{URegNym()} \xleftrightarrow{\text{Comm}_{U\underline{K}_1} : \texttt{Nym}_{UK}} \mathbf{K} : \mathsf{ORegNym()}$$

$$\mathbf{U} : \mathsf{UShowCred()} \xleftrightarrow{\text{Comm}_{U\underline{K}_2} : \texttt{ShowTranscript}_{UK_R}} \mathbf{K} : \mathsf{OVerifyCred()}$$

$$K \text{ stores } \{\text{OrgNym}_{UK}, \text{Transcript}_{UK_R}\}$$

$$\mathbf{U} : \text{Send()} \xleftrightarrow{\text{Comm}_{U\underline{K}_3} : \text{Payment}} \mathbf{K} : \text{Receive()}$$

$$\mathbf{U} : \mathsf{UGetCred()} \xleftrightarrow{\text{Comm}_{U\underline{K}_3} : \texttt{Cred}_{UK}} \mathbf{K} : \mathsf{OIssueCred()}$$

$$K \text{ updates } \{\text{OrgNym}_{UK}, \text{Transcript}_{UK_R}, \text{CredInfo}_{UK}\}$$

Figure 8: Example Application: Key Material and Registration

developed in Sections 4 to 6. Details of the linking of organizations' and users' key material to an external public-key infrastructure are discussed in Section 9 as part of a more general discussion on trust and certification.

8.1 Introducing the Application

LostFound is an online service where items can be traded on-line. Users can use LostFound to advertise and search for items, and can remain unidentifiable while doing so. All items should be traded for free, and users are not allowed to use the service for commercial purposes. Users pay a small fee for accessing the service during a certain amount of time; we allow for this fee to be paid by a non-anonymous bank transfer as long as the actual use of the service later on is unidentifiable.

It is assumed that users cannot misuse the service when only searching for items. Thus, as long as users only retrieve information, access can be unconditionally unidentifiable. When users advertise (post) items, however, they can misuse the service by posting commercial or even illegal (e.g., drugs) advertisements.

- **Showing Credential To LostFound and Retrieving Data:**

$$\mathbf{U} : \mathsf{UShowCred}() \xleftrightarrow{\text{Comm}_{U\underline{L}} : \texttt{ShowTranscript}_{UL_K(Get)}} \mathbf{L} : \mathsf{OVerifyCred}()$$

$$\mathbf{U} : \text{Get}() \xleftrightarrow{\text{Comm}_{U\underline{L}} : \texttt{RetrievedData}_1} \mathbf{L} : \text{Process}()$$

$$\mathbf{U} : \text{Get}() \xleftrightarrow{\text{Comm}_{U\underline{L}} : \texttt{RetrievedData}_2} \mathbf{L} : \text{Process}()$$

- **Showing Credential To LostFound and Posting Data:**

$$\mathbf{U} : \mathsf{UShowCred}() \xleftrightarrow{\text{Comm}_{U\underline{L}} : \texttt{ShowTranscript}_{UL_K(Post)}} \mathbf{L} : \mathsf{OVerifyCred}()$$

$$L \text{ stores } \text{Transcript}_{UL_K(Post)}$$

Figure 9: Example Application: Accessing LostFound

In the latter case, the misuse is considered criminal and the real identity of the user has to be provided to authorities. In the former case, LostFound simply wants to be able to revoke the user's subscription credential.

Deanonymization is thus a clear requirement in the system: revealing an identity behind a transaction implies the need for global deanonymization in posting transactions; revoking a credential implies local deanonymization. We want to protect both users as well as organizations or authorities against uncorrect or unfair functioning of the deanonymization process. Thus, we should give deanonymization rights to a dedicated entity, a deanonymizing organization, which is independent as well from users as from organizations or authorities in the system.

From the above description of high-level requirements, we can already derive which are the organizations playing a role in the system, and what are the main interactions between users and organizations, as well as between organizations.

LostFound is providing the actual service; a Root Authority (Root) ensures the external certification of users and issues the root credentials necessary for supporting the global deanonymization process; a deanonymization organization (DOrg) performs local and global deanonymization. We also introduce a Kiosk organization implementing the function of verifying a bank payment or transfer and in return issuing an anonymous subscription credential for accessing LostFound. We make this choice in order to more clearly separate this functionality from the functionality of actually delivering the LostFound service; however, Kiosk and LostFound can be operated by the same business entity, and the unidentifiability of users should be preserved also if they share their user data.

Figure 8 introduces the participants in the protocols with their various keys and certificates, and shows the protocol flows for registering with Root and Kiosk; Figure 9 shows the protocol flows related to accessing LostFound. Figure 10 shows detailed assertions and primitive invocation parameters related to the exchanges in Figures 8 and 9.

The reasoning behind the design of the actual protocol flows will be discussed in Section 8.4. First, we discuss the various organizations' key material and certificates, and the general security requirements of the communication channels.

$\texttt{RootNym}_{UR}$	$\texttt{RootNym}(\text{UserSecret}_U, \text{OrgNym}_{UR}, \text{UserNym}_{UR})$
$\texttt{SignedNymProof}_{UR}$	$\texttt{SignedNymProof}(\text{SIG}_{UR}, \text{Cert}_{CA-U}, \text{OrgNym}_{UR}, \texttt{null})$
$\texttt{Cred}_{UR}$	$\texttt{Cred}(\text{OrgNym}_{UR}, \text{UserCred}_{UR}, \text{CredInfo}_{UR})$
$\texttt{Nym}_{UK}$	$\texttt{Nym}(\text{UserSecret}_U, \text{OrgNym}_{UK}, \text{UserNym}_{UK})$
$\texttt{ShowTranscript}_{UK_R}$	$\texttt{ShowTranscript}(\text{Transcript}_{UK_R}, \text{UserCred}_{UR}, \text{CredShowInfo}_{UK_R},$ $\text{CredShowFeatures}_{UK_R}, \text{OrgNym}_{UK}, \texttt{null})$
$\texttt{Cred}_{UK}$	$\texttt{Cred}(\text{OrgNym}_{UK}, \text{UserCred}_{UK}, \text{CredInfo}_{UK})$
$\texttt{ShowTranscript}_{UL_K(Get)}$	$\texttt{ShowTranscript}(\text{Transcript}_{UL_K(Get)}, \text{UserCred}_{UK}, \text{CredShowInfo}_{UL_K},$ $\text{CredShowFeatures}_{UL_K(Get)}, \texttt{null}, \texttt{null})$
$\texttt{ShowTranscript}_{UL_K(Post)}$	$\texttt{ShowTranscript}(\text{Transcript}_{UL_K(Post)}, \text{UserCred}_{UK}, \text{CredShowInfo}_{UL_K},$ $\text{CredShowFeatures}_{UL_K(Post)}, \texttt{null}, \texttt{PostedMessage})$
CredInfo_{UR}	$\{\text{IPKey}_R, \texttt{true}, \text{Expiration}_{UR}, \texttt{null}\}$
$\text{CredShowInfo}_{UK_R}$	$\{\text{IPKey}_R, \texttt{any}, \text{Expiration}_{UK_R}, \texttt{any}\}$
$\text{CredShowFeatures}_{UK_R}$	$\{\texttt{true}, \text{VPKey}_K, \{\text{DPKey}_D, \texttt{CriminalMisuseCond}\}, \texttt{null}\}$
CredInfo_{UK}	$\{\text{IPKey}_K, \texttt{true}, \text{Expiration}_{UK}, \texttt{null}\}$
$\text{CredShowInfo}_{UL_K}$	$\{\text{IPKey}_K, \texttt{any}, \text{Expiration}_{UL_K}, \texttt{any}\}$
$\text{CredShowFeatures}_{UL_K(Get)}$	$\{\texttt{false}, \text{VPKey}_L, \texttt{null}, \texttt{null}\}$
$\text{CredShowFeatures}_{UL_K(Post)}$	$\{\texttt{false}, \text{VPKey}_L, \{\text{DPKey}_D, \texttt{CriminalMisuseCond} \mid \texttt{OtherMisuseCond}\}, \texttt{null}\}$

Figure 10: Example Application: Parameters of Assertions and Primitives

8.2 Key Material, External Certificates and *idemix* Certificates

Root (R) and Kiosk (K) will issue *idemix* credentials and thus need IssuerKeys; Kiosk and LostFound (L) will verify credentials and need VerifierKeys; DOrg (D) needs DeAnOrgKeys; and all organizations need keys for secure communication (see also Section 8.3). The user (U), of course, has a UserSecret.

As discussed in Section 7.3, all parties (users as well as organizations) have signature private keys SSKey_X and corresponding public-key certificates Cert_{CA-X} in an external Public-Key Infrastructure certified by a Certification Authority *CA*. U uses this certificate, Cert_{CA-U}, for root nym registration with R; the organizations use their signature key pairs and certificates to self-certify their *idemix* certificates (Cert_{R-R}, Cert_{K-K}, Cert_{L-L} and Cert_{D-D}) containing their *idemix* issuing, verification and deanonymization public keys as well as communication public keys. Detailed contents of these certificates will be discussed and motivated in Section 9.3.

8.3 Security of the Communication Channels

8.3.1 Authentication, Integrity and Confidentiality

In Figures 8 and 9, a communication channel between parties X and Y is represented as $\text{Comm}_{XY_{[i]}}$; the optional subscript indicates the sequence number in case X and Y execute multiple exchanges over potentially different communication channels.

All communication channels are at least one-way authenticated (Y to X, indicated by underlining Y in $\text{Comm}_{X\underline{Y}_i}$) and integrity-protected. This can be realized by an SSL-type server authentication (with

CSKey_Y) and session key establishment, and consecutive integrity-protection of the communication using the established session key.

The authentication and integrity-protection allow the originator X to make sure it is communicating with the correct (issuing, verifying or deanonymizing) organization Y. We state this requirement in order to avoid denial-of-service situations such as a user establishing a nym with another organization than the intended one, or a user showing a credential to another organization than the one from which he expects a service.

The need for confidentiality-protection depends on the data being exchanged on the communication channel. The communication channels transporting only *idemix* protocols carry the cryptographic *idemix* protocol exchanges, as well as meta-information exchanged by the state machines driving those cryptographic protocols. The cryptographic exchanges themselves do not exchange any information from which secrets can be derived. Meta-information consists of information such as which credential is requested and issued on which nym, or which credential is shown relative to which nym. The following lists, for the various *idemix* protocols between a user and an organization, the information which can be derived by an observer of the exchange. The use of subscripts to indicate parameter instances is the same as in Figures 1, 4 and 5.

- URegNym() - ORegNym(): OrgNym_{UI}
  The exchange during a nym registration procedure allows any observer to derive the resulting OrgNym;
- URegSignedNym() - ORegSignedNym(): $\mathrm{OrgNym}_{UI}, \mathrm{Cert}_{CA-U}, \mathrm{SIG}_{UI}, [\mathrm{Msg}_{UI}]$
  When registering a signed nym, the user additionally sends the external certificate, the signature, and the message to be signed; this meta-information is visible also to an observer;
- URegRootNym() - ORegRootNym(): $\mathrm{OrgNym}_{UR}, \mathrm{Cert}_{CA-U}, \mathrm{SIG}_{UR}, [\mathrm{Msg}_{UR}]$
  With a root nym registration, an observer acquires the same information as with a 'normal' signed nym registration;
- UGetCred() - OIssueCred(): $\mathrm{OrgNym}_{UI}, \mathrm{CredInfo}_{UI}$
  During a credential issuing exchange, the user has to send meta-information to the issuer indicating on which nym he wants which type of credential. This meta-information is visible to an observer of the communication;
- UShowCred() - OVerifyCred(): $\mathrm{CredShowInfo}_{UV}, \mathrm{CredShowFeatures}_{UV}, [\mathrm{OrgNym}_{UV}], [\mathrm{Msg}_{UV}]$
  When showing a credential, the user sends meta-information to the issuer indicating what he wants to show about the credential, the specific protocol options and possible parameters for deanonymization, (optionally) the nym relative to which the credential is shows, and (optionally) a message to be signed. This meta-information is visible to observers.

Thus, with exception of the 'potential' linkabilities towards deanonymizers or credential issuers resulting from deanonymization or showing a one-show credential (see Sections 6.10 and 6.11), an observer acquires the same linkability information as the organization(s) participating in the exchanges. This is not a problem, as the *idemix* protocols are designed exactly to protect a user's unidentifiability also towards these organizations. The communication channels carrying *idemix* protocols need thus not be confidentiality-protected as long as the communication channel provides sender anonymity.

The only reason, then, for protecting the confidentiality of data transmitted over these communication channels would be that the organization considers the exchanges to be business-sensitive information: e.g., the organization does not want observers to know how many credentials of a certain type he issued, or how many times he verified a certain type of credential to authorize service access.

We can thus conclude that the user's unidentifiability does not depend on the confidentiality-protection of his *idemix* exchanges; again, as long as the sender anonymity of the communication channel is protected.

As to the need for confidentiality-protection of communication channels transporting application data, we can state the following:

- The data retrieved from and posted to L is publicized at some point. Thus, as long as the user retrieving or posting it remains unidentifiable, encrypting this information does not additionally protect the user's privacy. Again, confidentiality-protection may be more in the interest of L, who may want to make sure that only paying customers can look at the data posted or retrieved.
- Another type of application data is the data exchanged for the non-*idemix* payment. As the user does not expect this payment to be anonymous, a simple account or user identifier in the payment may not need confidentiality-protection. Of course, if sensitive information such as a credit card number is exchanged, it would need appropriate confidentiality-protection; this should be taken care of by the payment protocol used.

8.3.2 Sender Anonymity

As discussed in Section 3.1, anonymity at the application level has to be supported also by anonymity of the user at the communication level. We assume the use of one of the methods mentioned for anonymyzing the user in a communication; some of these methods provide their own integrity- or confidentiality-protection.

8.3.3 Linking

In some cases, a sequence of several exchanges (such as showing a credential and receiving a new credential) have to be securely linked in one authenticated communication session in order to ensure the desired application-level security. These cases will be discussed on an individual basis.

8.4 Protocols for Registration and Service Access

We now derive the protocols satisfying the various requirements. By discussing the requirements from organizations and users for specific exchanges (e.g., posting application data), we will derive the exact protocol flows as depicted in Figures 8 and 9, as well as the contents of their parameters in Figure 10.

For each exchange, we will analyze the following requirements:

- Organization requirements:
  - which condition(s) has (have) to be fulfilled: the receipt of a payment (an *idemix*-external condition), or the verification of an *idemix* credential;
  - *idemix* linkability and deanonymization requirements: e.g., an *idemix* transaction has to be securely linked to a nym, or to another *idemix* transaction; or, it has to be deanonymizable;
  - application linkability requirements: e.g., an *idemix* transaction has to be securely linked to an actual application action (e.g., obtaining the actual service or data has to be linked to showing the necessary credential).

  The first two sets of requirements define the *idemix* access rules as defined in [9]; in case of a credential show condition, they define the CredShowInfo and CredShowFeatures for the credential show invocation. The third set of requirements will define whether a secure communication channel has to link an *idemix* transaction to an application action.
- User requirements: whether the transaction may or may not be linkable in a certain way to a user's identity or nyms.

Requirements such as deanonymization are related to the final application between U and L; they create requirements for the exchanges between U and K; these will in turn show the need for a root credential and root nym registration. Thus, we will start the analysis with the final exchange which is the actual application: retrieving data from, or posting data to, LostFound.

8.4.1 Retrieving Data From LostFound

Organization Requirements

- U may retrieve (Get()) application data (`RetrievedData`) from L on condition of having paid for the service; payment for the service can be proved by a credential from K (with an appropriate expiration). There is no need for U to sign an additional message in the show transcript. Thus, a $\texttt{ShowTranscript}_{UL_K(Get)}$ is required with:

$$\text{CredShowInfo}_{UL_K} = (\text{IPKey}_K, \texttt{any}, \text{Expiration}_{UL_K}, \texttt{any}),$$
$$\texttt{ShowTranscript}_{UL_K(Get)}.\text{Msg} = \texttt{null}$$

  The use of `any` means that so far there is no requirement by L on the value of the parameter or field.

- The access need not be identifiable; thus, no linkability with a nym or deanonymization is required; this means there are no requirements on $\text{CredShowFeatures}_{UL_K(Get)}$ other than the specification of L (VPKey_L) as verifier:

$$\text{CredShowFeatures}_{UL_K(Get)} = \{\texttt{any}, \text{VPKey}_L, \texttt{any}, \texttt{any}\}$$

- Showing the credential and retrieving the data should be linked by the same communication channel in order to make sure that the data is delivered to the user who showed the credential.

User Requirements

To fulfill the access condition, U will show a credential with CredInfo_{UK} fulfilling $\text{CredShowInfo}_{UL_K}$. I.e.,

$$\text{CredInfo}_{UK} = \{\text{IPKey}_K, \texttt{any}, \text{Expiration}_{UK}, \texttt{any}\}, \text{ with } \text{Expiration}_{UK} \geq \text{Expiration}_{UL_K}$$

U requires the access to be unconditionally unidentifiable, and thus unlinkable to any nym or identity. Neither $\text{CredShowInfo}_{UL_K}$ nor $\texttt{ShowTranscript}_{UL_K(Get)}.\text{Msg}$ is unique (we do not consider the expiration date in CredShowInfo as an attribute that makes a credential show unique); according to Section 6.7, the credential show is unconditionally unidentifiable if it is neither relative to a nym, nor deanonymizable, regardless of the number of accesses:

$$\text{CredShowFeatures}_{UL_K(Get)} = \{\texttt{false}, \text{VPKey}_\text{L}, \texttt{null}, \texttt{null}\},$$
$$\text{CredInfo}_{UK}.\text{MultiShow} = \text{CredShowInfo}_{UL_K}.\text{MultiShow} = \texttt{true}$$

Thus, U also chooses

$$\texttt{ShowTranscript}_{UL_K(Get)}.\text{OrgNym} = \texttt{null}$$

Also, as L does not require any specific attribute for the kiosk credential:

$$\text{CredShowInfo}_{UL_K}.\text{Attrs} = \texttt{any},$$

U may prefer that

$$\text{CredInfo}_{UK}.\text{Attrs} = \texttt{null}$$

to avoid any identifiability through attributes.

Figure 9 shows the user retrieving data after showing the credential. After $\texttt{ShowTranscript}_{UL_K(Get)}$, U can perform a number of Get() operations within the same secured communication session Comm_{UL}, fulfilling the organization's requirement to link the showing of the credential to the retrieval of the data; L processes them but need not store any transcripts.

8.4.2 Posting Data To LostFound

Organization Requirements

- U may post application data (`PostedMessage`) to L on condition of having paid for the service; payment for the service can be proved by a credential from K of the same type as for retrieving data. Thus, a $\texttt{ShowTranscript}_{UL_K(Post)}$ is required with the same CredShowInfo as for data retrieval:

  $$\text{CredShowInfo}_{UL_K} = \{\text{IPKey}_K, \texttt{any}, \text{Expiration}_{UL_K}, \texttt{any}\}$$

- The transaction needs both global and local deanonymization, depending on criminal or other misuse. As K and L belong to the same business entity, the issuer (K) of the credential can be assumed to be cooperating with the verifier (L) and can guarantee the global deanonymization. E.g., using the technique in Section 7.1, L then needs to implement only local deanonymization in order to also allow global deanonymization in case of criminal misuse. Deanonymization is the responsability of the dedicated deanonymization organization D:

  $$\text{CredShowFeatures}_{UL_K(Post)} = \\ \{\texttt{any}, \text{VPKey}_L, \{\text{DPKey}_D, \texttt{CriminalMisuseCond}|\texttt{OtherMisuseCond}\}, \texttt{any}\}$$

  In designing our application, we have assumed the presence of a Root organization R; we will refer to the nyms and credentials it issues as root nyms and root credentials. However, by choosing to realize global deanonymization using local deanonymization, we will not actually exploit the specific fact that the nyms R registers are root nyms (`RootNym()`). However, we do rely on R to ensure a linking between the nyms it registers and external certificates; we will, thus, exploit the fact that R registers signed nyms. To be precise, we rely on R to only issue a credential (as accepted by K) on a nym which was registered in a signed nym registration procedure using a signature key and certificate for which a user can be held accountable.

- L has another requirement: if ever deanonymization occurs, it does not suffice to prove the linking between U and the credential show; the linking between the user and the posted message must also be provable to a third party in order to guarantee U's accountability for the posted information. This cannot be achieved by linking the credential show and the posting through a secure communication channel, as such a linking cannot be proved to other parties. A provable linking can only be achieved if the posted message is also signed with the credential:

  $$\texttt{ShowTranscript}_{UL_K(Post)}.\text{Msg} = \texttt{PostedMessage}$$

- As the `PostedMessage` application data is already linked to the credential show, no additional application linking is needed.

User Requirements

U requires the access to be unlinkable to any nym or identity unless one of the above specified conditions for deanonymization is fulfilled. $\texttt{ShowTranscript}_{UL_K(Post)}.\text{Msg}$ is unique but does not occur in any other transaction; we do not consider $\text{CredShowInfo}_{UL_K}$ unique (verification of the expiration date should not make a credential show unique); from Sections 6.7 to 6.11 , we then conclude that the user requirement is fulfilled if there are no other ways of deanonymizing, and if the show is not relative to a nym:

$$\text{CredShowFeatures}_{UL_K(Post)}.\text{GlobalDeAnData} = \texttt{null}, \\ \text{CredShowFeatures}_{UL_K(Post)}.\text{RelNym} = \texttt{false}, \text{ and thus also} \\ \texttt{ShowTranscript}_{UL_K(Post)}.\text{OrgNym} = \texttt{null}$$

Figure 9 shows the posting of data. L processes the request and stores the show transcript $\text{Transcript}_{UL_K(Post)}$. In case of misuse, this will ensure that L and K can perform the necessary deanonymization to take measures, as discussed in Section 8.5.

8.4.3 Obtaining the Credential from Kiosk

In this procedure, K issues the credential to access L on condition of having received the appropriate payment. The payment can be any form of electronic payment and need not be anonymous (e.g., it can be a secure credit-card payment).

Organization Requirements

- K issues a credential only on the nym of a user who paid. Thus, either the credential issuing or the nym registration has to be securely linked to the payment. Also, K issues a credential on a nym only if K knows it can retrieve the user's root nym if the global deanonymization condition is fulfilled. We decided earlier that local deanonymization of a root credential show suffices for this.

  Thus, the procedure involves a $\texttt{Nym}_{UK}$ establishment (necessary to issue the credential $\texttt{Cred}_{UK}$ on), a locally deanonymizable root credential show $\texttt{ShowTranscript}_{UK_R}$, a payment, and the credential issuing $\texttt{Cred}_{UK}$. The root credential show can be securely linked to $\texttt{Nym}_{UK}$ by making it relative to $\texttt{Nym}_{UK}$. The payment can be securely linked to the credential issuing by a secure communication channel.

  $\texttt{Cred}_{UK}$'s CredInfo_{UK} was already specified in the previous paragraphs. As for $\texttt{ShowTranscript}_{UK_R}$, it requires a credential issued by R with a certain expiration limit and no specific attributes, local deanonymization (in case of criminal misuse only) and a showing relative to $\texttt{Nym}_{UK}$ and verifiable by K:

$$\text{CredShowInfo}_{UK_R} = \{\text{IPKey}_R, \texttt{any}, \text{Expiration}_{UK_R}, \texttt{any}\},$$
$$\text{CredShowFeatures}_{UK_R} = \{\texttt{true}, \text{VPKey}_\texttt{K}, \{\text{DPKey}_\texttt{D}, \texttt{CriminalMisuseCond}\}, \texttt{any}\},$$
$$\texttt{ShowTranscript}_{UK_R}.\text{OrgNym} = \texttt{Nym}_{UK}.$$

User Requirements

To fulfill $\texttt{ShowTranscript}_{UK_R}$, U shows a credential with CredInfo_{UR} fulfilling $\text{CredShowInfo}_{UK_R}$. Thus,

$$\text{CredInfo}_{UR} = \{\text{IPKey}_R, \texttt{any}, \text{Expiration}_{UR}, \texttt{any}\} \text{ with}$$
$$\text{Expiration}_{UR} \geq \text{Expiration}_{UK_R}$$

$\texttt{ShowTranscript}_{UK_R}$ is directly linked to $\texttt{Nym}_{UK}$. U's requirement is that the link between $\texttt{Nym}_{UK}$ and a root nym or real identity is revealed only when the local deanonymization condition is fulfilled, and that there are no other ways of deanonymizing (regardless of the number of uses of the root credential). As $\text{CredShowInfo}_{UK_R}$ is not unique (except for the expiration, which we do not consider an identifying factor), and $\texttt{ShowTranscript}_{UK_R}.\text{Msg} = \texttt{null}$, it follows that

$$\text{CredShowFeatures}_{UK_R}.\text{GlobalDeAnData} = \texttt{null}$$
$$\text{CredInfo}_{UR}.\text{MultiShow} = \texttt{true}$$

Also, as

$$\text{CredShowInfo}_{UK_R}.\text{Attrs} = \texttt{any},$$

the user may prefer that

$$\text{CredInfo}_{UR}.\text{Attrs} = \texttt{null}$$

to avoid any identifiability through attributes in the root credential.

Figure 8 shows the protocol flows. The payment and the issuing of the credential both use the same secure communication channel $\text{Comm}_{U\underline{K}_3}$, fulfilling K's requirement for linking these two exchanges.

Note that the order of the exchanges could be changed, as long as K has the same guarantees: namely, that the credential is issued to someone who paid, and on a nym that can be linked with a root nym. This guarantee could also be achieved by securely linking the payment with $\texttt{Nym}_{UK}$ establishment in Comm_{UK_1}, instead of linking it with the $\texttt{Cred}_{UK}$ issuing in Comm_{UK_3}.

8.4.4 Registering to Root

The goal of this procedure is for U to obtain the root credential needed to prove registration with R and therefore a linking with an external certificate. U will have to show this credential in order to obtain a subscription credential from Kiosk. As mentioned before, the fact that R issues root nyms ($\texttt{RootNym}()$) as opposed to 'regular' nyms ($\texttt{Nym}()$) is strictly speaking not exploited by our choice of deanonymization procedure; we could thus replace the root nym registration with a signed nym registration procedure.

Root nym registration and root credential issuing are standard procedures: R requires that it knows a provable linking between the root nym with an external certificate before issuing the root credential on it; the contents of $\texttt{Cred}_{UR}$ were discussed in previous paragraphs. The user knows that his root nym and root credential (the latter if shown in a deanonymizable way) are linkable to his external certificate Cert_{CA-U}. One parameter option still to be chosen is the contents of Msg_{UR} during root nym registration ($\texttt{SignedNymProof}_{UR}$): as Root needs no other proof than the linking with Cert_{CA-U}, we can conclude that

$$\texttt{SignedNymProof}_{UR}.\text{Msg} = \texttt{null}.$$

Figure 8 shows the root nym registration and root credential issuing. After the root nym registration, R stores the mapping between OrgNym_{UR} and Cert_{CA-U} (and the proof SIG_{UR} of this mapping). After issuing the root credential, R updates this information with CredInfo_{UR}.

Note that absence of authentication of R to U on Comm_{UR_1} would allow another organization to impersonate R. The impersonator would know a linking between U's real identity and the established nym, and U would assume the nym is shared with R. However, the impersonator could not issue a root credential on the nym, as this credential issuing would require the knowledge of IssuerKeys_R. The user would not be able to perform any operations with the nym. The only threat is thus one of denial of service. Absence of authentication of R to U on Comm_{UR_2} poses no threat as the issuing of the credential includes an authentication of R using its IssuerKeys_R.

8.5 Verifying Organizations' Accountability Requirements

In this Section, we now verify that our design satisfies our accountability goals. I.e., we verify that, after a message posting, L and K have the appropriate information to, together, locally or globally deanonymize the transaction. We illustrate the correctness of the deanonymization results by applying the derivation rules in Section 5 to the actual parameters of this transaction.

Figure 11 shows the deanonymization steps in a possible scenario. We have assumed deanonymization with proof: this will allow any party relying on the deanonymization result to verify its correctness; but also, to hold D accountable for the deanonymization (see also Section 9.1.2).

After any message posting, the assertions $\texttt{RootNym}_{UR}$, $\texttt{SignedNymProof}_{UR}$, $\texttt{Cred}_{UR}$, $\texttt{Nym}_{UK}$, $\texttt{ShowTranscript}_{UK_R}$, $\texttt{Cred}_{UK}$, and $\texttt{ShowTranscript}_{UL_K(Post)}$ hold, with the respective parameters as in Figure 10.

When the content $\texttt{PostedMessage}$ of a data posting transcript shows misuse, L can send the associated $\text{Transcript}_{UL_K(Post)}$ to D with the request to deanonymize it; this deanonymization returns OrgNym_{UK} and the correctness proof $\text{DeAnTranscript}_{D_{UL}}$. This can be derived as follows:

- **Step 1: Deanonymization to retrieve** OrgNym_{UK}

$$\mathbf{L}: \text{Send}() \xrightarrow{\text{Comm}_{\underline{LD}_1} : \text{Transcript}_{UL_K(Post)}} \mathbf{D}: \text{Receive}()$$

$$\mathbf{D}: \mathsf{DODeAnonLocalWProof}()$$

$$\mathbf{L}: \text{Receive}() \xleftarrow{\text{Comm}_{\underline{LD}_1} : \text{OrgNym}_{UK}, \text{DeAnTranscript}_{D_{UL}}} \mathbf{D}: \text{Send}()$$

- **Step 2: Deanonymization to retrieve** OrgNym_{UR}

$$\mathbf{K}: \text{Send}() \xrightarrow{\text{Comm}_{\underline{KD}_1} : \text{Transcript}_{UK_R}} \mathbf{D}: \text{Receive}()$$

$$\mathbf{D}: \mathsf{DODeAnonLocalWProof}()$$

$$\mathbf{K}: \text{Receive}() \xleftarrow{\text{Comm}_{\underline{KD}_1} : \text{OrgNym}_{UR}, \text{DeAnTranscript}_{D_{UK}}} \mathbf{D}: \text{Send}()$$

Figure 11: Example Application: Local and Global Deanonymization

From Figure 10:

`Nym`$(\text{UserSecret}_U, \text{OrgNym}_{UK}, \text{UserNym}_{UK})$,
`Cred`$(\text{OrgNym}_{UK}, \text{UserCred}_{UK}, \text{CredInfo}_{UK})$,
`ShowTranscript`$(\text{Transcript}_{UL_K(Post)}$, UserCred_{UK}, $\text{CredShowInfo}_{UL_K}$, $\text{CredShowFeatures}_{UL_K(Post)}$, `null`, `PostedMessage`),
$\text{CredShowFeatures}_{UL_K(Post)}.\text{LocalDeAnData.DPKey} = \text{DPKey}_D$.

From Section 5.1, it follows that an invocation of

$$\mathsf{DODeAnonLocalWProof}(\{\text{DSKey}_D, \text{DPKey}_D\}, \text{Transcript}_{UL_K(Post)})$$

will return $\text{DeAnTranscript}_{D_{UL}}$ with

$$\boxed{\texttt{LDeAnProof}(\text{DeAnTranscript}_{D_{UL}}, \text{Transcript}_{UL_K(Post)}, \text{OrgNym}_{UK})}$$

Based on OrgNym_{UK} and the data posting transcript with `PostedMessage` provided by L, K can decide whether or not to revoke $\texttt{Cred}_{UK}$. In case of misuse motivating global deanonymization, the second deanonymization step will also be executed in order to provide for global deanonymization of $\texttt{ShowTranscript}_{UL_K(Post)}$. K then retrieves the stored $\{\text{OrgNym}_{UK}, \text{Transcript}_{UK_R}, \text{CredInfo}_{UK}\}$. From Figure 10:

`ShowTranscript`$(\text{Transcript}_{UK_R}$, UserCred_{UR}, $\text{CredShowInfo}_{UK_R}$, $\text{CredShowFeatures}_{UK_R}$, OrgNym_{UK}, `null`)

From Section 4.6, it follows:

$$\boxed{\texttt{SigProof}(\text{Transcript}_{UK_R}, \text{CredShowInfo}_{UK_R}, \text{CredShowFeatures}_{UK_R}, \text{OrgNym}_{UK}, \texttt{null})}$$

Thus, K can prove the relationship between Transcript_{UK_R} and OrgNym_{UK}. K then asks D for local deanonymization of Transcript_{UK_R} to obtain OrgNym_{UR} (and $\text{DeAnTranscript}_{D_{UK}}$):

From Figure 10:

$\texttt{ShowTranscript}(\text{Transcript}_{UK_R}, \text{UserCred}_{UR}, \text{CredShowInfo}_{UK_R}, \text{CredShowFeatures}_{UK_R}, \text{OrgNym}_{UK}, \texttt{null})$, $\text{CredShowFeatures}_{UK_R}.\text{LocalDeAnData.DPKey} = \text{DPKey}_D$

From Section 5.1, it follows that an invocation of

$$\textsf{DODeAnonLocalWProof}(\{\text{DSKey}_D, \text{DPKey}_D\}, \text{Transcript}_{UK_R})$$

returns $\text{DeAnTranscript}_{D_{UK}}$ with

$$\boxed{\texttt{LDeAnProof}(\text{DeAnTranscript}_{D_{UK}}, \text{Transcript}_{UK_R}, \text{OrgNym}_{UR})}$$

With OrgNym_{UR}, K can obtain the associated Cert_{CA-U} from R, who has stored $\{\text{OrgNym}_{UR}$, Cert_{CA-U}, SIG_{UR}, $\text{CredInfo}_{UR}\}$. Note that the linking between OrgNym_{UR} and Cert_{CA-U} is also a provable linking according to (from Figure 10):

$$\boxed{\texttt{SignedNymProof}(\text{SIG}_{UR}, \text{Cert}_{CA-U}, \text{OrgNym}_{UR}, \texttt{null})}$$

Thus, from a post transcript, D can locally deanonymize $\text{Transcript}_{UL_K(Post)}$ to obtain OrgNym_{UK}; K can link OrgNym_{UK} to a Transcript_{UK_R}, D can deanonymize Transcript_{UK_R} to obtain OrgNym_{UR}, and R can link OrgNym_{UR} to Cert_{CA-U}. Each of the linkings is provable, as can be seen from the framed statements.

We now discuss the security requirements on the communication channels used, and other requirements related to authentication in the deanonymization exchanges. Also here, we assume at least one-way (D to L and D to K) authentication in order to protect against denial-of-service attacks, and to prevent sending sensitive information to a non-authorized party. As the deanonymization result is sensitive information, it is important that in both cases D's response is sent only to the requestor of the deanonymization. This can be achieved by ensuring that the response is sent over the same secured (integrity-and confidentiality-protected) communication channel. Most likely, D also wants to authenticate the sender of the request: first, D does not want to be used as a deanonymization oracle; second, it is probable that the requesting organization has to be authenticated in order to securely verify the deanonymization condition. Authentication by the requestor can be achieved by using also client-side authentication in the communication channel using CSKey_L, resp. CSKey_K. Such a two-way authenticated communication channel is depicted in Figure 11 as $\text{Comm}_{\underline{KD}}$. Alternatively, L and K could sign the requests to D using their signature key (SSKey_L, resp. SSKey_K). This makes L, resp. K, accountable for the deanonymization request itself. This may be preferable, e.g., if D, when having deanonymized a transcript, needs to be able to prove that the deanonymization indeed was requested by the specific organization.

Note that $\text{DeAnTranscript}_{D_{UK}}$ ($\text{DeAnTranscript}_{D_{UL}}$) also includes the equivalent of a signature (with DSKey_D) and therefore is also implicitly authenticated by D. This is a stronger authentication than the authentication on the communication channel, as it also makes D accountable for the deanonymization of the transcript itself as discussed in Section 9.1.2.

8.6 Verifying User Unidentifiability Requirements

In this section, we verify that the user's unidentifiability requirements are met.

According to Section 6, we can derive following linkability assertions from the credential, nym and transcript assertions that hold after registration and payment.

$\texttt{RootNym}_{UR}$
$\rightarrow -.$

$\texttt{SignedNymProof}_{UR}$
$\rightarrow$ (1) $\texttt{Linked}(R, \texttt{null}, \text{OrgNym}_{UR}, \text{Cert}_{CA-U})$.

As $\texttt{SignedNymProof}_{UR}.\text{Msg} = \texttt{null}$, no additional linking results from $\texttt{SignedNymProof}_{UR}$.

$\texttt{Cred}_{UR}, \neg\texttt{Unique}(\text{CredInfo}_{UR})$
$\rightarrow -.$

As CredInfo_{UR} is generic (not unique for the credential instance), no linking results from $\texttt{Cred}_{UR}$.

$\texttt{Nym}_{UK}$
$\rightarrow -.$

$\texttt{ShowTranscript}_{UK_R}$,
$\text{CredShowFeatures}_{UK_R}.\text{RelNym} = \texttt{true}$
$\rightarrow$ (2) $\texttt{Linked}(K, \texttt{null}, \text{Transcript}_{UK_R}, \text{OrgNym}_{UK})$.

$\texttt{ShowTranscript}_{UK_R}$,
$\text{CredShowFeatures}_{UK_R}.\text{LocalDeAnData.DPKey} = \text{DPKey}_D$,
$\texttt{Cred}_{UR}$,
$\texttt{Nym}_{UR}$
$\rightarrow$ (3) $\texttt{Linked}(D, \texttt{CriminalMisuseCond}, \text{OrgNym}_{UR}, \text{Transcript}_{UK_R})$.

As neither $\text{CredShowInfo}_{UK_R}$ nor $\texttt{ShowTranscript}_{UK_R}.\text{Msg}$ $(= \texttt{null})$ are unique, no other linkabilities result from $\texttt{ShowTranscript}_{UK_R}$.

With the $\texttt{Cred}_{UK}$ credential obtained, U can now retrieve data or post messages; these transactions may add additional linkability information.

8.6.1 Data Retrieval

Data retrieval is not relative to a nym, has no deanonymization option and no unique parameters and thus does not result in linkability information. Data retrieval is thus unconditionally unidentifiable, which is consistent with requirements.

$\texttt{ShowTranscript}_{UL_K(Get)}$
$\rightarrow -.$

8.6.2 Message Posting

Message posting results in an additional $\texttt{ShowTranscript}_{UL_K(Post)}$ assertion. $\texttt{ShowTranscript}_{UL_K(Post)}.\text{Msg}$ $(= \texttt{PostedMessage})$ is unique, but does not occur in any other transaction. Thus, the only linkability follows from deanonymization:

$\texttt{ShowTranscript}_{UL_K(Post)}$,
$\text{CredShowFeatures}_{UL_K(Post)}.\text{LocalDeAnData.DPKey} = \text{DPKey}_D$,
$\texttt{Cred}(\text{OrgNym}_{UK}, \text{UserCred}_{UK}, \text{CredInfo}_{UK})$,
$\texttt{Nym}(\text{UserSecret}_U, \text{OrgNym}_{UK}, \text{UserNym}_{UK})$

$\rightarrow$ (4) $\texttt{Linked}(D, \{\texttt{CriminalMisuseCond}|\texttt{OtherMisuseCond}\}, \text{OrgNym}_{UK}, \text{Transcript}_{UL_K})$.

The user can conclude that only OrgNym_{UK} can be linked to the post transcript, and only in case of one of the misuses mentioned. From (2), OrgNym_{UK} can directly be linked to Transcript_{UK_R} and, from (3), Transcript_{UK_R} to OrgNym_{UR} in case of criminal misuse. From (1), OrgNym_{UR} can directly be linked to the user's identity in Cert_{CA-U}. This is consistent with requirements.

9 Trust, Accountability, Liability and Certification

This section deals with measures needed to create a system where users can trust the correct application of mechanisms protecting their unidentifiability, and organizations can trust the correct functioning of mechanisms guaranteeing user accountability. We use the Lostfound application developed in Section 8 to illustrate the challenges as well as the solutions.

Users fundamentally mistrust organizations: they expect organizations to collaborate with each other and to combine their knowledge; any unconditional linkability, as well as any conditional linkability of which the condition is fulfilled, can be considered public. E.g., the linking of a user's external certificate to his root nym is public; unidentifiability is threatened only when a user executes an action which is traceable to that root nym.

From the linkabilities described in Section 6, the ones resulting from double-spending and deanonymization are the only conditional ones. As for double-spending, the user can control the fulfillment of the condition; in addition, the linkability is technically unfeasible if the condition is not met. The case is different for deanonymization: the user may have less control over the fulfillment of the condition (e.g., because it is subject to interpretation); in addition, we assume deanonymization to be technically feasible even if the condition is not fulfilled.

Deanonymization thus takes a special place in a discussion about trust: it can have a far-reaching (negative) effect on users' unidentifiability; while at the same time being the only process where trust by users is required.

In Section 9.1, we investigate trust required specifically in the deanonymization process. As many of the issues are relevant to both users and organizations, we discuss trust in deanonymization from both users' and organizations' point of view. We discuss how accountability and verifiability of the deanonymization process can help safeguarding fairness. These measures limit the need for 'blind trust' by at least exposing and proving untrustworthy behaviour.

Section 9.2 then specifically discusses trust required by organizations in the system. Some issues are similar to the ones discussed for the case of deanomymization; we again discuss measures that expose misbehaviour by other organizations.

Exposure of misbehaviour or cheating may discourage such behaviour; it does, however, not provide any guarantees that it is punished. Section 9.3 deals with these guarantees through liability-enhanced certification as introduced in [24]. In Section 9.1.2, we then illustrate the concepts introduced in this section using a LostFound deanonymization scenario.

9.1 Trust by Users and Organizations in Deanonymization

In the LostFound application, organizations' accountability requirements are fulfilled by the application of local and global deanonymization, on condition of misuse being shown; from the users' perspective, their unidentifiability is unconditionally protected during information retrieval; for information posting, it is conditional on the absence of misuse.

In order for the deanonymization process to work as expected by both users and organizations, the following have to be fulfilled:

- Conditions under which accountability (deanonymization) is needed have to be stated in a concrete enough way that D can either evaluate them automatically, or can at least make an evaluation which can be considered fair by both parties (the user and the entity requesting deanonymization);
- The deanonymizer acts correctly, i.e., deanonymizes if and only if the condition is fairly evaluated to be true.

The latter condition requires trust by users, who do not want unmotivated deanonymization to occur, as well as by the organizations who need to rely on deanonymization in order to fulfill accountability requirements. In the following paragraphs, we discuss the fairness of condition evaluation and deanonymization, and ways to increase fairness (or decrease the amount of trust) through verifiability.

9.1.1 Fairness of Condition Evaluation

In fact, the entity evaluating the deanonymization condition need not be the deanonymizer D: just as liability conditions in certificates can be subject to evaluation by a trusted Arbiter, deanonymization conditions may be subject to evaluation by an Arbiter (or multiple Arbiters) other than the deanonymizer. E.g., in $\mathrm{CredShowFeatures}_{UK_R}$,

$$\texttt{CriminalMisuseCond} = \{\texttt{CriminalMisuseDesc}, \mathrm{SPKey}_{LE}\}$$

would indicate that D is allowed to deanonymize if LE (a Law Enforcement entity) has attested, by a signature using its signature key SSKey_{LE}, that criminal misuse (as described in the description `CriminalMisuseDesc`) has occurred.

$$\texttt{OtherMisuseCond} = \{\texttt{OtherMisuseDesc}, \mathrm{SPKey}_{L}, \mathrm{SPKey}_{A}\}$$

could indicate that D is allowed to deanonymize if LostFound as well as an arbiter organization A have attested that other misuse (as described in `OtherMisuseDesc`) has occurred.

This separation of responsibilities has the advantage of separating *idemix* functionality from high-level application (and possibly human) condition evaluation, and potentially allows to automate the condition evaluation by deanonymizers; also, it facilitates the definition of conditions that have to be evaluated by multiple Arbiters.

The separation between condition evaluation and deanonymization may also result in an additional distribution of trust. For this to be the case, D's deanonymization actions should be verifiable, as discussed in the next paragraph.

9.1.2 Verifiability of Deanonymization: Correctness and Accountability

Verifiability of the deanonymizers' actions by both users and organizations helps to reduce the level of trust required. It should be verifiable *that D* performed a certain deanonymization (accountability), and that the deanonymization was *correct* w.r.t. the deanonymized transcript (correctness). Both accountability and correctness are achieved if deanonymization is always accompanied by the proof of correctness. Thus, users as well as organizations can verify (and prove to another entity) who performed a deanonymization, and whether it was correct.

9.1.3 User Trust Requirements Related to Deanonymization

Trust requirements by the user in the system are thus minimized through distribution of trust, separation of duties and verifiability of actions.

When a user U shows a credential issued by issuer I to verifier V, I and V together can under no condition (unless in case of double-spending of a one-show credential) identify the user or its OrgNym_U. This is true even if the credential was shown in a deanonymizable way: the introduction of a deanonymizing organization D as the only entity to be able to identify the user or OrgNym_{UI} is a first separation of duties (and trust), and protects the user against coalitions of issuers and verifiers.

As to the trust in a deanonymization organization D: if a user has a choice between multiple deanonymizers, this reduces trust requirements; as discussed in the previous paragraphs, the (verifiable) evaluation of deanonymization conditions by dedicated (sets of) Arbiters, and the verifiability of the deanonymization itself, introduces an additional separation of trust.

Accountability, verifiability and correctness need of course to be backed by appropriate liability of (in this case) the deanonymization organization for its actions, or by a liability of the entity certifying D to revoke D's keys in the event of D malfunctioning or misbehaving. These liabilities are discussed in Section 9.3. Of course, revocation of keys does not solve (or appropriately punish) a misbehavior or malfunctioning resulting in DSKey_D being stolen or even published. In such a case, D has to be held accountable for existing transcripts being deanonymized without reason.

In conclusion, accountability and correctness, together with verifiable fairness of condition evaluation, protect a user from being unfairly accused of actions he did not perform, and of losing his unidentifiability without a valid reason.

9.1.4 Organizations' Trust Requirements Related to Deanonymization

K and L in our example application, in order to rely on a fair deanonymization, need to rely on D to deanonymize transcripts according to evaluated deanonymization conditions.

Verifiable fairness of condition evaluation, together with accountability and correctness of deanonymization, also provide organizations such as K and L with a guarantee that deanonymization is done correctly and fairly. However, for organizations, an additional source of unfairness could be that D is not available, or not willing to respond to a request. This type of unfairness can be dealt with by introducing a 'Fairness-Arbiter' in the system: such an arbiter can be used as an intermediary in deanonymization (and other) requests and can evaluate whether there is a response, and whether it is of the expected form. E.g., in the case of deanonymization, the Fairness-Arbiter could verify that the response is either a deanonymization result, or a (signed) motivation why deanonymization could not be performed, e.g., the absence of a positive condition evaluation.

Though K and L (and any other entity relying on deanonymization) need to rely on D for deanonymization, D's liabilities are concrete and can be high, as explained in the next section. Thus, the trust in the correct functioning of accountability need not be higher than in a conventional PKI. In a conventional PKI, every transaction is linked to the same (non-anonymous) certificate issued by a CA. However, in order to realize accountability for a transaction, CA still may have to cooperate to map an identity in a certificate to a real human identity; in addition, typical CA liabilities for guaranteeing such a mapping are unclear or nonexistent.

9.2 Trust by Organizations in the System

For relying on correct functioning of the system, organizations have to trust more than only the deanonymization process and deanonymizing organizations.

Firstly, in order for accountability requirements to be fulfilled, they also need to rely on obtaining 'public information' from each other. In the LostFound application (see Section 8.5), R needs to be relied on to make available (through publicizing or on request) the linkings between external certificates and root nyms; K has to be relied on to make available root credential show transcripts of all the nyms on which it issued subscription credentials.

Secondly, depending on business agreements and contracts, organizations may have obligations to each other subject to certain credentials being shown. E.g., an organization accepting e-coins issued by an e-bank as payment for a service expects that the e-bank will endorse a spent e-coin with 'real money'. Depending on the business relationship between L and K in the LostFound example, K may need to pay the subscription amount paid by a user to L.

Fairness of (and thus trust in) all these processes again requires accountability and verifiability of the actions leading to such an obligation. It can be verified that an e-coin issued by a specific e-bank was spent, and it can be verified that the e-bank did (or did not) endorse the e-coin by crediting the receiving organization's bank account.

As mentioned before, verifiability and accountability may discourage but do not prevent misbehaviour; nor do they specify consequences (liabilities or punishments) for the misbehaving party. In the following section, we discuss the application of liability-enhanced certification to users' and organization's certificates.

9.3 Certificates, Liabilities and Contracts

Accountability of organizations or users for any *idemix*-related actions needs to be based on appropriate registration and certification procedures for users and organizations in the system; appropriate certification has to involve a liability of the certifier for actions with the certified key:

- User accountability is based on the enforced use of signed (in this case root) nym registration and the existence of an external certificate Cert_{CA-U}, guaranteeing a linking to a legal entity.
- Accountability of a deanonymizing organization D is based on the fact that DPKey_D is certified by D's signature key; this certification (and liability) has a value because D's signature public key SPKey_D itself is certified by an entity, CA, who can make guarantees about or state liability for D (e.g., to revoke D's keys upon misbehaviour or malfunctioning).
- A credential based on a credential issuing key ISKey_I has a value through appropriate self-certification of IPKey_I by I using its signature key SSKey_I; the signature public key SPKey_I again is certified by CA, which provides a guarantee for actions or certificates signed with SSKey_I.

Not only *idemix* keys (issuing, verification and deanonymization) and signature keys have to be certified; also the keys used for communication (e.g., SSL) have to be certified in order to provide secure authentication, integrity and confidentiality of communication channels.

We now discuss the certificate infrastructure for our example application described in Section 8. Figure 12 shows the contents of various certificates.

9.3.1 Users' External Certificates

The users' external certificate Cert_{CA-U} is certified by a certification authority CA; it contains the user's (signature) public key SPKey_U as well as Liab_{CA-U}, the liabilities of the certifying party. Cert_{CA-U}, as well as the other certificates discussed in the following paragraphs, may contain additional attributes such as expiration time which are not discussed here.

The main function of Cert_{CA-U} is to guarantee an unconditional linking with a human user. Therefore, Liab_{CA-U} carries a high liability for guaranteeing this linking:

$$\text{Liab}_{CA-U} = \{\{\text{LiabTp}=\texttt{Li},\ \text{LiabAmount}=\$100000,\ \text{LiabCondition}=\texttt{none},\ \text{LiabVerifier}=\texttt{none}\}\}$$

indicates an unconditional (LiabCondition=none, LiabVerifier=none) liability of \$100000 for providing, to a party relying on a signature with SSKey_U, the identity (LiabTp=Li) of a human user owning the certificate.

Cert_{CA-U}	$S_{CA}(\mathrm{CertdKeys} = \{\mathrm{SPKey} = \mathrm{SPKey}_U\}, \mathrm{Attrs} = \{\mathrm{CertTp} = \texttt{CERT}, \mathrm{Liab} = \mathrm{Liab}_{CA-U}\})$
X	$\in \{R,\ K,\ L,\ D\}$
Cert_{CA-X}	$S_{CA}(\mathrm{CertdKeys} = \{\mathrm{SPKey} = \mathrm{SPKey}_X\}, \mathrm{Attrs} = \{\mathrm{CertTp} = \texttt{CERT}, \mathrm{Liab} = \mathrm{Liab}_{CA-X}\})$
Cert_{X-X}	$S_X(\mathrm{CertdKeys} = \mathrm{CertdKeys}_{X-X}, \mathrm{Attrs} = \mathrm{Attrs}_{X-X})$
Attrs_{X-X}	$\{\mathrm{CertTp} = \texttt{SELFCERT}, \mathrm{Liab} = \mathrm{Liab}_{X-X}, \mathrm{Addr} = \mathrm{Addr}_X,$ $\mathrm{Rules} = \mathrm{Rules}_X, \mathrm{Contr} = \mathrm{Contr}_X\}$
$\mathrm{CertdKeys}_{R-R}$	$\{\mathrm{IPKey} = \mathrm{IPKey}_R, \mathrm{CPKey} = \mathrm{CPKey}_R\}$
$\mathrm{CertdKeys}_{K-K}$	$\{\mathrm{IPKey} = \mathrm{IPKey}_K, \mathrm{VPKey} = \mathrm{VPKey}_K, \mathrm{CPKey} = \mathrm{CPKey}_K\}$
$\mathrm{CertdKeys}_{L-L}$	$\{\mathrm{VPKey} = \mathrm{VPKey}_L, \mathrm{CPKey} = \mathrm{CPKey}_L\}$
$\mathrm{CertdKeys}_{D-D}$	$\{\mathrm{DPKey} = \mathrm{DPKey}_D, \mathrm{CPKey} = \mathrm{CPKey}_D\}$

Figure 12: Certificate Contents For LostFound Application Example

9.3.2 Organizations' External Certificates

An organization X's external certificate Cert_{CA-X} follows the same format as the user's. Liab_{CA-R}, Liab_{CA-K}, Liab_{CA-L} and Liab_{CA-D} probably contain a liability for certain of the attributes Attrs_{CA-X} of these organizations. In addition, parties relying on these certificates want a liability for the correctness of these organizations' actions. As *CA* probably cannot take financial liability for these, its liability may be limited to ensuring that behind each of these organizations there is a legal entity that can be held responsible for incorrect actions. This is essentially again a liability of type Li: every organization can be held accountable for its actions.

Liab_{CA-X} = {{LiabTp=`Ld`, LiabAmount=\$100000,
LiabCondition='attrs incorrect', LiabVerifier=Arbiter},
{LiabTp=`Li`, LiabAmount=\$100000,
LiabCondition=`none`, LiabVerifier=`none`}}
indicates that *CA* takes a data (LiabTp=`Ld`) liability of \$100000 for correctness of Attrs_{CA-X} (as witnessed by Arbiter), and an unconditional identification (Li) liability of \$100000 for mapping the entity X to a liable human party or other legal entity. X can now express concrete liabilities for actions in its self-certified *idemix* certificates.

Alternatively or additionally to this Li liability, *CA* may express a liability for revoking Cert_{CA-X} in certain cases of misbehaviour.

9.3.3 Organizations' *idemix* Certificates

Depending on the role of the organization, its *idemix* self-signed certificate Cert_{X-X} certifies one or more other public keys IPKey_X, VPKey_X, DPKey_X, CPKey_X. Rules_X in Attrs_{X-X} contains access rules as described in [9]; Contr_X expresses the set of an organization's guarantees or promises towards parties relying on the various certified keys; Liab_{X-X} expresses various liabilities taken by X. These may be liabilities for credentials issued by X using ISKey_X, for verifications using VSKey_X, for deanonymizations with DSKey_X, or for not fulfilling part of the guarantees in Contr_X. Some examples are:

- Contr_R captures the fact that R should be able to provide an external Cert_{CA-U} for a valid root nym OrgNym_{UR}. More precisely, if an OrgNym_{UR} is the deanonymization result of the show of a root credential, then R has to be able to provide a verifiable linking $(\mathrm{SIG}_{UR}, \mathrm{Cert}_{CA-U}, \mathrm{OrgNym}_{UR})$. Also for root credentials that are being revoked, R still has to be able to provide this linking during a certain period of time. If $\texttt{contr\_breach\_cond}_R$ is a condition expressing that the above contract guarantee is not fulfilled, and Arbiter is an entity trusted to verify that the conditions for providing the linking were fulfilled and the guarantee was not honored, then the following expresses R's liability for not providing the linking:

$$\begin{aligned}\text{Liab}_{R-R} = \{\{&\text{LiabTp} = \texttt{Lt}, \text{LiabAmount} = \$100000,\\ &\text{LiabCondition} = \texttt{contr\_breach\_cond}_R, \text{LiabVerifier} = \text{Arbiter}\}, \ldots\}\end{aligned}$$

where the liability type (LiabTp=$\texttt{Lt}$) indicates that this is a transaction-related liability.

- Contr$_D$ could include the service of deanonymizing a transaction if and only if the deanonymization condition is fulfilled. A condition $\texttt{contr\_breach\_cond}_{D_1}$ for contract breach could then be a refusal or incapacity to deanonymize a transaction if the deanonymization condition is satisfied (as witnessed by the Arbiter in the liability condition); another example of a contract breach condition $\texttt{contr\_breach\_cond}_{D_2}$ would be a deanonymization that was carried out without its deanonymization condition being fulfilled (again as witnessed by the Arbiter), or without D being able to show that an authorized party requested the deanonymization.

$$\begin{aligned}\text{Liab}_{D-D} = \{&\{\text{LiabTp} = \texttt{Lt}, \text{LiabAmount} = \$100000,\\ &\quad\text{LiabCondition} = \texttt{contr\_breach\_cond}_{D_1}, \text{LiabVerifier} = \text{Arbiter}\},\\ &\{\text{LiabTp} = \texttt{Lt}, \text{LiabAmount} = \$100000,\\ &\quad\text{LiabCondition} = \texttt{contr\_breach\_cond}_{D_2}, \text{LiabVerifier} = \text{Arbiter}\}, \ldots\}\end{aligned}$$

- Contr$_L$ captures the fact that L should be able to provide a transcript Transcript$_{UL_K(Post)}$ for every item posted on LostFound. If $\texttt{contr\_breach\_cond}_L$ is a condition expressing that the above contract guarantee is not fulfilled, and Arbiter is a trusted arbiter for this conditon:

$$\begin{aligned}\text{Liab}_{L-L} = \{\{&\text{LiabTp} = \texttt{Lt}, \text{LiabAmount} = \$100000,\\ &\text{LiabCondition} = \texttt{contr\_breach\_cond}_L, \text{LiabVerifier} = \text{Arbiter}\}, \ldots\}\end{aligned}$$

- Contr$_K$ captures the fact that K should be able to provide a transcript Transcript$_{UK_R}$ for every nym $\texttt{Nym}_{UK}$ registered with K; this transcript, in itself, bears the proof that it is related to $\texttt{Nym}_{UK}$. If $\texttt{contr\_breach\_cond}_K$ is a condition expressing that the above contract guarantee is not fulfilled, and Arbiter is a trusted arbiter for this conditon:

$$\begin{aligned}\text{Liab}_{K-K} = \{\{&\text{LiabTp} = \texttt{Lt}, \text{LiabAmount} = \$100000,\\ &\text{LiabCondition} = \texttt{contr\_breach\_cond}_K, \text{LiabVerifier} = \text{Arbiter}\}, \ldots\}\end{aligned}$$

9.4 A Scenario Illustrating Trust in Accountability and Anonymity: Law Enforcement and Credential Revocation

With the certificate contents defined above, and the definition of the various transactions and their parameters in Figures 8 to 10, we can now give some concrete illustrations of trust and liability related to certification.

We will again consider the accountability and anonymity requirements stated in Section 8 and verified in Sections 8.5 and 8.6; this time, we will analyse them from the point of view of trust needed, by either the organization relying on accountability, or the user relying on unidentifiability.

9.4.1 Trusting Accountability and Deanonymization

Consider the following scenario: an item posted on LostFound is judged by LE, the law enforcement entity, to represent criminal abuse (e.g., `PostedMessage` is an illegal drugs advertisement). LE requires that the identity of the user who posted the message be revealed in a provable way.

Other organizations involved in this process are L, K, D, R and an Arbiter. Arbiter is essentially a 'Fairness-Arbiter' verifying whether responses to requests are obtained and whether they have the correct format and contents; it also judges the various contract breach conditions listed in Section 9.3.3.

Depending on who drives the various processes to obtain nyms, transcripts etc., we can describe multiple deanonymization scenarios. In one scenario, LE would contact L with the request to deanonymize the transaction; L would then be responsible for contacting K and D, etc. However, this would make L partially responsible for deanonymization, which would have to be expressed in its liabilities; L would have to take a liability of another organization on which it depends.

In order to clearly separate liabilities and responsibilities of individual organizations, we assume that LE is driving the deanonymization process: LE contacts one organization at a time and requests a specific service. The scenario is shown in Figure 13. We also assume that, when asking for a linking or deanonymization, LE also receives the proof of the linking or of the correctness of the deanonymization result. This proof allows Arbiter to verify the correctness of the response.

The communication channels between LE and each of the organizations it contacts are mutually authenticated and confidentiality-protected. Authentication by LE allows the various organizations to verify that an authenticated entity (LE) is the one requesting the information; authentication by the organization receiving a request allows LE to verify that it does not send sensitive deanonymization requests to other than the intended organization; encryption protects the sensitive data (requests as well as responses) from being read by unauthorized parties. The deanonymization requests by LE will, in addition, be digitally signed: this allows the deanonymizing organization D serving the request to prove that the actual request occurred.

In an exchange with L, LE requests and receives the transcript related to the `PostedMessage` advertisement. $\text{Transcript}_{UL_K(Post)}$ in itself carries the proof that it is the transcript related to `PostedMessage`. Thus, both Arbiter and LE can verify and prove that the response is correct.

From D, LE then requests deanonymization of this transcript. In the signed request, LE includes also `CriminalMisuseDesc` and `PostedMessage`: this allows D to prove not only that the deanonymization request was made by LE, but also fulfills the `CriminalMisuseCond` as defined in Section 9.1.1 and will allow D to prove this. The response contains OrgNym_{UK} as well as the proof of correctness $\text{DeAnTranscript}_{D_{UL}}$; again, both LE and Arbiter can verify the correctness of the result.

LE then asks K for the root credential show transcript related to OrgNym_{UK}. The transcript returned, Transcript_{UK_R}, carries the proof of being related to OrgNym_{UK}, which can again be verified and proved by LE as well as Arbiter.

In a second exchange with D, LE then asks deanonymization of Transcript_{UK_R}. LE again includes `Criminal-MisuseDesc` and `PostedMessage`. However, as Transcript_{UK_R} is not related to `PostedMessage`, LE needs to provide D with a proof of this relationship. This is done by sending along OrgNym_{UK} and $\text{Transcript}_{UL_K(Post)}$. D has cached or re-verifies the linking between OrgNym_{UK} and $\text{Transcript}_{UL_K(Post)}$; as Transcript_{UK_R} can be proved to be related to OrgNym_{UK}, D is convinced that Transcript_{UK_R} is indeed generated by the same user who generated the criminal posting. The result of this second deanonymization consists of OrgNym_{UR} and $\text{DeAnTranscript}_{D_{UK}}$; also this result can be verified and proved.

In a final step, LE requests U's external certificate from R. Once more, including the proof SIG_{UR} of the linking in the response allows the correctness of this linking to be verified and proved.

In each of the above steps, LE requests a service for which the responding organization has taken some responsibilities and liabilities. Concretely, according to our liabilities specification in Section 9.3.3, unfair behaviour in the various steps would result in either of the following contract breach conditions being fulfilled:

$$\begin{array}{ll}
LE \rightarrow L: & \texttt{request\_trscr}, \texttt{PostedMessage} \\
LE \leftarrow L: & \text{Transcript}_{UL_K(Post)} \\
\\
LE \rightarrow D: & \text{SIG}_{LE}(\texttt{request\_dean}, \texttt{CriminalMisuseDesc}, \texttt{PostedMessage}, \text{Transcript}_{UL_K(Post)}) \\
LE \leftarrow D: & \text{OrgNym}_{UK},\ \text{DeAnTranscript}_{D_{UL}} \\
\\
LE \rightarrow K & \texttt{request\_trscr}, \text{OrgNym}_{UK} \\
LE \leftarrow K: & \text{Transcript}_{UK_R} \\
\\
LE \rightarrow D & \text{SIG}_{LE}(\texttt{request\_dean}, \texttt{CriminalMisuseDesc}, \texttt{PostedMessage}, \text{Transcript}_{UK_R}, \\
 & \qquad \text{OrgNym}_{UK}, \text{Transcript}_{UL_K(Post)}) \\
LE \leftarrow D: & \text{OrgNym}_{UR},\ \text{DeAnTranscript}_{D_{UK}} \\
\\
LE \rightarrow R: & \texttt{request\_cert}, \text{OrgNym}_{UR} \\
LE \leftarrow R: & \text{Cert}_{CA-U}, \text{SIG}_{UR}
\end{array}$$

Figure 13: A Scenario Illustrating Trust in Accountability and Anonymity

- $\texttt{contr\_breach\_cond}_{\texttt{L}}$, if L does not return a correct transcript of the message posting;
- $\texttt{contr\_breach\_cond}_{\texttt{D}_1}$, if D does not respond with a correct deanonymization result and proof for either deanonymization request;
- $\texttt{contr\_breach\_cond}_{\texttt{K}}$, if K does not reply with a correct root credential show transcript;
- $\texttt{contr\_breach\_cond}_{\texttt{R}}$, if R does not return the correct external certificate.

As the presence as well as the correctness of a response can be verified by Arbiter, the relying party (LE in this case) is protected against unfair or uncorrect behaviour by the responding organization according to the (high) liability amounts in the various organizations' *idemix* certificates.

9.4.2 Trusting Anonymity and Unlinkability

Let us now look at the above scenario from the user's point of view, who wants to trust that his rights to unidentifiability are respected.

Getting $\text{Transcript}_{UL_K(Post)}$ from L, or obtaining Transcript_{UK_R} from K, is not tied to any condition to be fulfilled, and neither Cert_{K-K} nor Cert_{L-L} contain liabilities for giving away this information without reason. Indeed, this information may as well be considered public as a party receiving it cannot use the transcripts without deanonymization by D. This is also the reason why we did not require LE to sign these requests: no liability is attached to giving the response without proving an appropriate reason.

In order for the user to be unfairly deprived of unidentifiability, at least one of the deanonymizations must be unfair. If this is the case, it fulfills $\texttt{contr\_breach\_cond}_{\texttt{D}_2}$.

Clearly, U cannot be unfairly accused of misuse as this would require proofs which do not exist, including deanonymization transcripts which provably fulfill $\texttt{contr\_breach\_cond}_{\texttt{D}_2}$. However, D could collude with LE or another entity to deanonymize even if the deanonymization condition is not fulfilled; as long as they do not use the result to accuse U of the stated misbehaviour, this collusion is difficult to prove. One way to protect against this is making all D's actions auditable, e.g., by having Arbiter or another monitor audit all deanonymizations handled by D.

10 Conclusion

In this report, we have shown how to design secure applications based on the *idemix* anonymous credential system. We first described the *idemix* protocols at a detailed enough interface level to be able to concretely specify applications based on them. Based on these interfaces, we also were able to define relationships, 'assertions', between nyms, credentials and credential show transcripts resulting from *idemix* interactive protocol executions. These assertions not only represent compact and intuitive building blocks for describing complex applications using *idemix* credentials for attribute-based authorization; they also allow to describe accountability and linkability consequences resulting from *idemix* protocol executions. Therefore, they are necessary in order to build applications based on authentication, accountability and unidentifiability requirements of the various parties (users and organizations) in the system.

We illustrated these findings with an example application involving a two-step deanonymization process. We constructed the protocols used by the application based on authentication, accountability and unlinkability requirements of users and organizations, and showed that the resulting application indeed fulfilled these requirements.

Correctness of the application protocols, however, is only one step in showing the correct operation of the overall system; it does not give any indication on the correct behaviour of various parties on which users or organizations in the system rely; nor does it stipulate consequences or liabilities related to such misbehaviour. E.g., a deanonymization organization may behave unfairly towards relying organizations as well as towards users. The last section of this report dealt with these issues of fairness and trust. We showed how various organizations' behaviour can be made verifiable; and how users' and organizations' trust in the correct operation of other organizations could be minimized by defining appropriate liabilities in organizations' certificates.

A A Generic Certificate Format

In this appendix, we describe the certificate format used in Section 9.

The definition of our generic certificate format is inspired by following requirements:

- Certificates should allow to express liabilities taken by the certificate issuer, as introduced in [24]. A liability can be related to
  - correctness of data in the certificate (liability of type `Ld`);
  - transactions made with (the private key associated with) the certificate (liability of type `Lt`);
  - being able to reveal of the certificate holder for a pseudonymous certificate (liability of type `Li`).
- The certificate notation should allow to distinguish between externally certified and self-certified certificates. In the example in Section 9, an external *CA* certifies a user's or organization's master public signature key (of type SPKey) in an external certificate; organizations self-certify special purpose-keys (such as credential issuing, communication security, deanonymization or verification keys) using their master private signature key (of type SSKey).
- The certificate notation should also allow to express a certificate request as a special type of certificate. This notion was introduced in [24]. Though this feature is not exploited in the examples in this report, it is added for completeness as we expect future work to refer to the generic certificate format in this appendix.

In order to provide a unified representation for all certificates, certificate requests and self-signed certificates, we define a generic notation. Note that the certificate notation presented here merely serves to facilitate our discussion and does not claim to be a general all-purpose format (e.g., the key and certificate types discussed are limited to the ones needed in our application). The notation is shown in Figure 14. Naming of certificate fields and values is kept intuitive; e.g., Attrs_{Y-X} is the value of the field Attrs in the certificate Cert_{Y-X}.

An entity Y signing a message can do so with any of a number of keys certified for him. As only signatures with master keys and (credential- or certificate-) issuing keys are relevant to our discussion, we introduce a simple notation which allows us to distinguish between these different signatures: $\mathrm{S}_Y(\mathrm{Msg})$ and Cert_{Y-X} are signed with Y's master signature key; while $\mathrm{S}_{Y_i}(\mathrm{Msg})$ and Cert_{Y_i-X} are signed with Y's special-purpose issuing key (private key ISKey_Y with public counterpart IPKey_Y). Issuing and other special-purpose keys, such as keys for verification, deanonymization (re-identification) and securing communication, were introduced in Section 3.1.

In a certificate Cert_{Y-X} (Cert_{Y_i-X}), an entity Y certifies, by a signature with a master signature key SSKey_Y (issuing key ISKey_Y) a key or set of keys $\mathrm{CertdKeys}_{Y-X}$ ($\mathrm{CertdKeys}_{Y_i-X}$) belonging to entity X together with a set of attributes Attrs_{Y-X} (Attrs_{Y_i-X}).

Depending on the relationship between entities Y and X, Cert_{Y-X} (or Cert_{Y_i-X}) represents different types of certificates:

- $Y = X$. With Cert_{X-X}, an entity (organization) X self-certifies special-purpose public keys using a 'master' signature key. This type of certificate is introduced in Section 9.3.
- Y and X are different identities belonging to the same user, such as C and P (for a customer's real and pseudonym identity) as used in this chapter. Such a certificate is the equivalent of the CERT_REQ (certificate request) type certificate in [24]: it is at the same time a request for a certificate as a self-certification by C of a public key (and attributes) for P. A certificate request carries two liabilities: one is the desired issuer liability in the requested certificate; a second type of liability is that taken by the requestor for the public key on which he is requesting a certificate. The latter liability was not discussed in [24] but is clearly needed to bootstrap user liability and accountability.

Cert_{Y-X}	certificate certifying X's public key(s), signed with Y's master signature key $=\mathrm{S}_Y(\mathrm{CertdKeys} = \mathrm{CertdKeys}_{Y-X}, \mathrm{Attrs} = \mathrm{Attrs}_{Y-X})$
Cert_{Y_i-X}	certificate certifying X's public key(s), signed with Y's issuing key $=\mathrm{S}_{Y_i}(\mathrm{CertdKeys} = \mathrm{CertdKeys}_{Y_i-X}, \mathrm{Attrs} = \mathrm{Attrs}_{Y_i-X})$
$X,\ Y$	identifiers (names) of entities
Y_i	entity Y in its credential or certificate issuing role (signing with its issuing key ISKey_Y)
Z	stands for either Y or Y_i
$\mathrm{SPKey}_Y, \mathrm{SSKey}_Y$	Y's (master) signature public/private key pair
$\mathrm{IPKey}_Y, \mathrm{ISKey}_Y$	Y's public/private key pair for issuing certificates or credentials (Section 3.1)
$\mathrm{VPKey}_Y, \mathrm{VSKey}_Y$	Y's public/private key pair for verifying credentials (Section 3.1)
$\mathrm{DPKey}_Y, \mathrm{DSKey}_Y$	Y's public/private key pair for deanonymizing (Section 3.1)
$\mathrm{CPKey}_Y, \mathrm{CSKey}_Y$	Y's public/private key pair for securing communication (Section 3.1)
$\mathrm{S}_Y(\mathrm{Msg})$	signature on Msg with SSKey_Y (corresponding to signature public key SPKey_Y)
$\mathrm{S}_{Y_i}(\mathrm{Msg})$	signature on Msg with ISKey_Y (corresponding to issuing public key IPKey_Y)
CertTp	`CERT`| `CERTREQ`| `SELFCERT`
`CERT`	certificate type for traditional certificate
`CERTREQ`	certificate type for certificate request
`SELFCERT`	certificate type for self-signed certificate
$\mathrm{CertdKeys}_{Z-X}$	X's (public) keys certified in Cert_{Z-X} $= \{[\mathrm{SPKey} = \mathrm{SPKey}_X], [\mathrm{IPKey} = \mathrm{IPKey}_X], [\mathrm{VPKey} = \mathrm{VPKey}_X], [\mathrm{DPKey} = \mathrm{DPKey}_X], [\mathrm{CPKey} = \mathrm{CPKey}_X]\}$
Attrs_{Z-X}	attributes in Cert_{Z-X} $=\{\mathrm{CertTp} = \mathrm{CertTp}_{Z-X}, [\mathrm{Liab} = \mathrm{Liab}_{Z-X}], [\mathrm{LiabReq} = \mathrm{LiabReq}_{Z-X}], [\mathrm{CertrKeyReq} = \mathrm{CertrKeyReq}_{Z-X}], \ldots\}$
`Ld`	liability type for data liability
`Lt`	liability type for transaction liability
`Li`	liability type for identity liability
LiabAmount	amount of a liability item
LiabCondition	condition under which a liability item becomes effective
LiabVerifier	identifier of entity trusted to verify a LiabCondition
Liab_{Z-X}	liability taken by the certifier; consists of zero or more liability items as defined in [24] $=\{\{\mathrm{LiabTp} = \{\mathtt{Ld}\vert\mathtt{Lt}\vert\mathtt{Li}\}, \mathrm{LiabAmount} = \ldots, \mathrm{LiabCondition} = \ldots, \mathrm{LiabVerifier} = \ldots\}*\}$
$\mathrm{LiabReq}_{Z-X}$	(in Cert_{Z-X} of type `CERTREQ`) liability requested by the certifier of Cert_{Z-X} to be present in the requested certificate; same structure as Liab_{Z-X} $= \mathrm{SPKey}_Y$ (for $Z = Y$), $= \mathrm{IPKey}_Y$ (for $Z = Y_i$)
$\mathrm{CertrKeyReq}_{Z-X}$	(in Cert_{Z-X} of type `CERTREQ`) public issuing key of requested certificate

Figure 14: Generic Certificate Format

- Y and X belong to different entities. This is the case of a traditional certificate where an entity Y certifies a key or set of keys for another entity X.

CertdKeys_{Z-X} (Z standing for Y or Y_i) may thus contain a signature public key SPKey_X associated with entity X, or any special-purpose keys. Note that also public encryption keys could go here; they are however not used in this report.

Attrs_{Z-X} contains all the attributes and liabilities associated with Cert_{Z-X}. The only mandatory element in Attrs_{Z-X} is a field CertTp indicating whether Cert_{Z-X} is a traditional certificate (CertTp=`CERT`), a certificate request (CertTp=`CERTREQ`) or a self-signed certificate (CertTp=`SELFCERT`); other fields are optional and their presence may depend on this CertTp.

The names (identities) of certifying entity and certified entity may also belong in Attrs_{Z-X}. We will however consider public keys to be (pseudonym) identities; we will thus not explicitly mention identities in certificates. Also protocol and role identifiers as used in [24] are not anymore explicitly mentioned; they can be assumed to be captured by the combination of key types and liability conditions.

Other attributes may be defined on a per-application basis. E.g., in the example in Section 9.3, credential issuing organizations include addressing information, access control rules and contract information as attributes in their self-signed certificates. Liab_{Z-X} is the liability taken by the certificate issuer for the certified key(s). In case the certificate is of type `CERTREQ`, the issuer (signer) of a certificate request is the user requesting the certificate, and Liab_{Z-X} is the liability of the user for the public key on which he requests a certificate of type `CERT`. In that same case, LiabReq_{Z-X} may indicate the desired liability in the requested certificate; and CertrKeyReq_{Z-X} indicates the desired issuing public key (and implicit issuer's identity) of the requested certificate.

References

[1] The anonymizer. `http://www.anonymizer.com`.

[2] G. Ateniese, D. Song, and G. Tsudik. Quasi-efficient revocation in group signatures. In *Proc. 2002 International Conference on Financial Cryptography*, volume 2357 of *Lecture Notes in Computer Science*. Springer-Verlag, 2002.

[3] G. Ateniese and G. Tsudik. Some open issues and new directions in group signatures. In *Proc. 1999 International Conference on Financial Cryptography*, volume 1648 of *Lecture Notes in Computer Science*, pages 196–211. Springer-Verlag, 1999.

[4] M. Blaze, J. Feigenbaum, and A. D. Keromytis. Keynote: Trust management for public-key infrastructures (position paper). In *Proc. 1998 Security Protocols International Workshop*, volume 1550 of *Lecture Notes in Computer Science*, pages 59–63. Springer-Verlag, 1998.

[5] S. Brands. Towards digital credentials. ERCIM News No.49, April 2002.

[6] J. Camenisch. Idemix protocol specification. To be published.

[7] J. Camenisch and A. Lysyanskaya. Efficient non-transferable anonymous multi-show credential system with optional anonymity revocation. In *Advances in Cryptology – EUROCRYPT '01*, volume 2045 of *Lecture Notes in Computer Science*, pages 93–118. Springer-Verlag, 2001.

[8] J. Camenisch and A. Lysyanskaya. Dynamic accumulators and application to efficient revocation of anonymous credentials. In *Advances in Cryptology – CRYPTO '02*, volume 2442 of *Lecture Notes in Computer Science*, pages 61–67. Springer-Verlag, 2002.

[9] J. Camenisch and E. Van Herreweghen. Design and implementation of the idemix anonymous credential system. In *Proc. 2002 ACM conference on Computer and Communications Security*, Washington D.C., November 2002. ACM Press.

[10] D. Chaum and J.-H. Evertse. A secure and privacy-protecting protocol for transmitting personal information between organizations. In M. Odlyzko, editor, *Advances in Cryptology – CRYPTO '86*, volume 263 of *Lecture Notes in Computer Science*, pages 118–167. Springer-Verlag, 1987.

[11] D. L. Chaum. Security without identification: transaction systems to make big brother obsolete. *Communications of the ACM*, 28(10):1030–1044, 1985.

[12] L. Chen. Access with pseudonyms. In *Cryptography: Policy and Algorithms*, volume 1029 of *Lecture Notes in Computer Science*, pages 232–243. Springer Verlag, 1995.

[13] I. B. Damgård. Payment systems and credential mechanism with provable security against abuse by individuals. In *Advances in Cryptology – CRYPTO '88*, volume 403 of *Lecture Notes in Computer Science*, pages 328–335, 1990.

[14] C. Ellison, B. Frantz, B. Lampson, R. Rivest, B.Thomas, and T. Ylonen. SPKI certificate theory. RFC 2693, Sept. 1999.

[15] D. M. Goldschlag, M. G. Reed, and P. F. Syverson. Onion routing for anonymous and private internet connections. *Communications of the ACM*, 42(2):84–88, February 1999.

[16] C. Gulcu and G. Tsudik. Mixing e-mail with BABEL. In *Proc. 1996 Symposium on Network and Distributed System Security*, San Diego, CA, Feb. 1996. Internet Society.

[17] ISO/IEC. Information technology - open systems interconnection - the directory: Authentication framework, June 1994.

[18] R. Kailar. Reasoning about accountability in protocols for electronic commerce. In *Proc. 1995 IEEE Symposium on Research in Security and Privacy*, pages 236–250. IEEE Computer Society Press, 1995.

[19] A. Lysyanskaya, R. Rivest, A. Sahai, and S. Wolf. Pseudonym systems. In *Selected Areas in Cryptography*, volume 1758 of *Lecture Notes in Computer Science*. Springer-Verlag, 1999.

[20] A. Pfitzmann, B. Pfitzmann, and M. Waidner. ISDNMixes: untraceable communication with very small bandwidth overhead. In *Proc. Communication in Distributed Systems*, number 267 in Informatik-Fachberichte, pages 451–463. Springer-Verlag, 1991.

[21] M. K. Reiter and A. D. Rubin. Crowds: anonymity for Web transactions. *ACM Transactions on Information and System Security*, 1(1):66–92, 1998.

[22] K. H. Rosen, editor. *Handbook of Applied Cryptography*, volume B of *CRC Press Series on Discrete Mathematics and its Applications.* CRC Press, 1990. ISBN 0-8493-8523-7.

[23] SET Secure Electronic transaction LLC. *SET Secure Electronic Transactions Technical Specifications*, version 1.0 edition, May 1997. Book One: Business Description, Book Two: Programmer's Guide, Book Three: Formal Protocol Definition. Available from `http://www.setco.org/set_specifications.html`.

[24] E. Van Herreweghen. Secure anonymous signature-based transactions. In *Proc. 2000 European Symposium on Research in Computer Security (ESORICS)*, number 1895 in Lecture Notes in Computer Science. Springer-Verlag, 2000.