Efficient Selective Disclosure on Smart Cards Using Idemix

Pim Vullers, Gergely Alpár

▶ **To cite this version:**

Pim Vullers, Gergely Alpár. Efficient Selective Disclosure on Smart Cards Using Idemix. 3rd Policies and Research in Identity Management (IDMAN), Apr 2013, London, United Kingdom. pp.53-67, 10.1007/978-3-642-37282-7_5 . hal-01470503

HAL Id: hal-01470503
https://inria.hal.science/hal-01470503

Submitted on 17 Feb 2017

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution 4.0 International License

Efficient Selective Disclosure on Smart Cards using Idemix*

Pim Vullers[1]** and Gergely Alpár[1,2]***

[1] Institute for Computing and Information Sciences,
Radboud University Nijmegen, The Netherlands.
{p.vullers, g.alpar}@cs.ru.nl
[2] TNO Information and Communication Technology, The Netherlands.

Abstract In this paper we discuss an efficient implementation for selective disclosure of attribute-based credentials on smart cards. In this context we concentrate on the implementation of this core feature of IBM's Identity Mixer (Idemix) technology. Using the MULTOS platform we are the first to provide this feature on a smart card. We compare Idemix with Microsoft's U-Prove technology, as the latter also offers selective disclosure of attributes and has been implemented on a smart card [9].

Keywords: selective disclosure, attribute-based credentials, smart card, attributes, idemix, u-prove, multos

1 Introduction

The world is moving into a digital era. Many people spend time on the internet, not just for fun or gathering information, but also for shopping and banking. Not only do our activities happen in the digital world, existing systems are also moving to digital alternatives. Train tickets are being replaced by electronic public transport cards. Identity documents, such as passports, are equipped with chips to hold digital copies of the identity data printed on the document and sometimes even additional information is stored there such as fingerprints or other biometric data.

Unfortunately, most such systems use a simple approach to identify entities; they just attach a unique number to them. While this is convenient for bookkeeping, it also has a big drawback with respect to privacy. Using these unique identifiers, it is easy to trace the user. For example, not only might it be possible to track user activities on the world wide web, but real world actions could also be traced through the use of public transport cards or digital identity documents.

* The work described in this paper has been supported in part by the European Commission through the ICT programme under contract ICT-2007-216676 ECRYPT II.
** Sponsored by Trans Link Systems/Open Ticketing.
*** Partly supported by the research program Sentinels as project 'Mobile IDM' (10522). Sentinels is being financed by Technology Foundation STW, the Netherlands Organisation for Scientific Research (NWO), and the Dutch Ministry of Economic Affairs.

These unique identifiers are used to identify entities during authentication and/or authorisation, but actually in most use cases identification is not necessary. For instance, when you want to buy liquor, a merchant only needs to verify that you are of a certain age. The same holds when boarding a train; the system only needs to know whether or not you are allowed to do so, and there is no direct need for the system to know exactly who you are.

A more privacy-friendly approach is possible by using attribute-based credentials. Instead of providing lots of identity information to the service provider, the user can now just provide the required attributes, such that the service can be accessed without the user revealing his identity.

In this paper we use the Identity Mixer (Idemix) technology [7,8,11] developed by IBM Research to implement attribute-based credentials. This system allows the user to receive a signed list of attributes from a trusted party which can then be used to convince a service provider. A core feature of this technology, *selective disclosure*, enables a user to control which attributes from this list get revealed to the service provider.

Having public transport cards and identity documents in mind, we focus on smart card implementations. We use cards running the MULTOS platform that offers an API suitable for implementing cryptographic protocols. Our prototype achieves the best performance for Idemix on a smart card thus far, with running times which are acceptable for on-line, and certain off-line scenarios.

While others have implemented Idemix on a smart card [5,12], we are the first to provide the selective disclosure functionality. We compare our implementation against an implementation [9] of Microsoft's U-Prove system [6,10] which offers similar functionality, and currently provides the best smart card performance. The benefit of using Idemix is its multi-show unlinkability property, which allows a single credential to be used multiple times, whereas U-Prove only provides single-show unlinkability and hence requires multiple credentials to provide anonymity instead of just pseudonymity.

2 Attributes and Credentials

Within this article, an attribute will be understood as some property of a person. One can think of "over 18", "male", or "owner of bank account 1234" as examples of attributes. Informally, a person's identity can be seen as the collection of all attributes that hold for this person. In practice, many transactions can be based on a subset of attributes, such as buying a bottle of whiskey by verifying the "over 18" attribute while other attributes are irrelevant.

There are several cryptographic systems for dealing with attribute-based identities. Typically these systems distinguish credentials and attributes. Informally, a credential is a *cryptographic container of attributes*. As a first approximation, one can think of a credential as depicted in Figure 1.

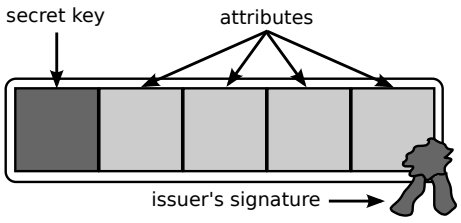


Figure 1. A first look at an attribute-based credential with four attributes.

2.1 Issuance and Selective Disclosure

Credentials are *issued* and *verified*, whereas attributes can be *disclosed* during verification. In the issuance procedure, an issuer and a user together create a new credential. First the user authenticates to the issuer in some reliable but unspecified manner (which may be face-to-face). Once the authentication succeeds, the issuer collects attributes for this user from trusted databases. The user and issuer then carry out a cryptographic protocol in which the attributes are combined into a credential signed by the issuer. The resulting credential contains attributes concerning the user and also his/her secret key.

The fact that the attributes hold for the owner of a credential is guaranteed both by the issuer's signature and by the embedded secret key of the owner. The secret key in the credential is only known by the credential owner and plays an essential role in verification of the credential. It ensures that a credential cannot be transferred from one user to another.

A user may have several credentials, each asserting some collection of attributes. When requesting a service from a service provider, the user is required to authenticate using one (or more) of his/her credentials. In the verification process the user can choose to only provide certain credentials; also, given a specific credential, the user may choose to reveal only a subset of attributes in the credential. By doing this, authentication becomes more privacy friendly. This latter process is called selective disclosure, involving a verification protocol in which only a subset of the credential attributes is revealed to the verifier while the other attributes are only proved to be present in the credential. This allows a user to reveal only the necessary attributes and prove that the credential belongs to him/her. The service provider can verify all information that has been sent, including the issuer's signature.

The roles of a service provider and an issuer can also be combined: after checking one credential, a new one can be issued. For instance, after verifying an "over 18" attribute from an identity credential, a liquor shop might choose to issue a loyalty credential.

In this paper we stick to a simplistic approach to selective disclosure in which an index set $\mathcal{D}$ determines the information to be revealed. Selective disclosure is sometimes defined more generally. We consider only the projection function $F_{\mathcal{D}}$ that is determined by a disclosing subset $\mathcal{D}$ of attribute indexes. For example, if there are four attributes and $\mathcal{D} = \{1, 3\}$, then $F_{\mathcal{D}}(a_1, a_2, a_3, a_4) = (a_1, a_3)$.

However, a more general definition would allow F to be any function of the attributes, for example, $F(a_1, a_2, a_3, a_4) := (a_1 > 18, a_2 + a_3)$.

2.2 Security and Privacy

The cryptographic nature of the credential-as-container concept includes the following four security aspects.

- The issuer's digital signature ensures *authenticity*: the credential originates from the issuer, and this issuer asserts that the attributes hold for the person.
- This signature also guarantees *integrity*: the attributes contained in the credential have not been altered since they were issued.
- A credential is *non-transferable* as it is bound to the secret key of the person involved in the issuance protocol. This secret key should be well protected, for instance via storage in the secure memory of a smart card with a PIN.
- A credential *hides* its content, so it does not reveal the attributes it contains.

Furthermore, a credential protects the privacy of its owner through the following cryptographic properties.

- *Issuer unlinkability* ensures that any information gathered during issuing cannot be used to link a verification of the credential to its issuance.
- *Multi-show unlinkability* guarantees that when a credential is verified multiple times, these sessions cannot be linked.

The privacy of users is protected by these unlinkability properties even if the credential issuer and all verifiers collude. As briefly reviewed below, a variety of ways of achieving the properties above have been proposed.

Blindly signed single-show credentials[3] are credentials in which the issuance involves creating a blind signature which conceals the resulting credential from the issuer. Therefore, the verification instances of this credential cannot be related to the issuing phase. Examples of this approach are Brands credentials [6], which are used in Microsoft's U-Prove system [10], and Baldimtsi and Lysyanskaya credentials [3].

Randomisable credentials employ special cryptographic techniques enabling certain credential structures to be randomised using blinding factors while preserving their verifiability. In such systems, users can randomise their credentials before they are verified. Such credentials have been proposed by Verheul [14], and implemented by Batina et al. [4], although this technique currently only supports binary attributes; for example, instead of a descriptive string of nationality, the answer to the question "Is French?" is stored.

Zero-knowledge proofs allow a user to prove ownership of a credential without revealing the credential itself. Since the verifier does not see the credential, verification instances are unlinkable and they also cannot be related to the issuing procedure. Camenisch and Lysyanskaya [7,8] combine such proofs with randomisation; their technique was further developed and became IBM's Identity Mixer (Idemix) [11].

[3] Multi-show unlinkability for these schemes can be realised by issuing multiple credentials for the same set of attributes which can later be verified independently.

3 Technologies for Selective Disclosure

In the following sections we briefly study the cryptographic techniques how credentials are constructed and used in U-Prove and in Idemix. In both techniques the concept of *"possessing"* a credential is equivalent to *"knowing"* its secret key. Furthermore, to be able to carry out all the computations, the user (or the device in the user's control) has to know the attributes in the credential.

3.1 Selective Disclosure with U-Prove

U-Prove [6,10] relies on the discrete logarithm (DL) cryptographic assumption. Informally, the DL assumption states that given a group and a generator, it is hard to find the exponent that the generator has to be raised to get a randomly given group element. We give a simplified description of U-Prove below using multiplicative notation (for a full specification we refer to [10]):

- A group is required in which the DL problem is hard. For instance, the following multiplicative group:
  - p, q primes (usual DL setup: $q|p-1$) and generator g in $\mathbb{Z}_p^*$ of order q, i.e., $(1, g, g^2, g^3, \ldots, g^{q-1}) \pmod p$ forms a cyclic multiplicative group of q elements. The group can then be described by (p, q, g)
- Key generation is only performed by the issuer, the user obtains distinct secret keys corresponding to each credential during the issuing protocol.
  - Issuer: generate $x, y_1, \ldots, y_l \in_R \mathbb{Z}_q^*$, compute $h \equiv g^x \pmod p$ and $g_1 \equiv g^{y_1} \pmod p, \ldots, g_l \equiv g^{y_l} \pmod p$, store $sk_I = (x, y_1, \ldots, y_l)$, and publish $pk_I = (h, g_1, \ldots, g_l)$, where l denotes the number of attributes.
- Issuing a credential is performed using an interactive protocol in which an issuer blindly signs a commitment. The attributes $(\alpha_1, \ldots, \alpha_l)$ are assumed to be common input to the issuer and the user; the resulting credential and the corresponding secret key α are only known to the user.
  - The resulting credential is $Cred = (h', \sigma)$ and the user's secret key is α. Credential $Cred$ includes a commitment $h' = g^{\alpha} \prod_1^l g_i^{\alpha_i} \pmod p$, which information-theoretically hides the attributes (because of the secret key α) and computationally binds the user to them, and a signature $\sigma = (c', r')$ of the issuer on this commitment, such that $c' = \mathcal{H}\left(h' \| g^{r'} (h \cdot h')^{-c'}\right)$.
- Selective disclosure of attributes is achieved using an interactive protocol between the user and a service provider (verifier). The credential $Cred$, a disclosing index set $\mathcal{D}$ and the corresponding attributes $(\alpha_i)_{i \in \mathcal{D}}$ are revealed by the user. Furthermore the user provides a partial commitment $R = g^{\alpha} \prod_{i \notin D} g_i^{\alpha_i} \pmod p$ to the unrevealed attributes.
  - First, the verifier checks the signature σ in credential $Cred$:

$$c' \stackrel{?}{=} \mathcal{H}\left(h' \| g^{r'} (h \cdot h')^{-c'}\right).$$

- Second, the user proves knowledge of all secret values in the commitment h' in a zero-knowledge protocol, a proof of knowledge

$$pr = PK\{(\alpha, (\alpha_i)_{i \notin D}) : h' \cdot (g_i^{-\alpha_i})_{i \in D} \equiv R \pmod{p}\}.$$

  Without revealing any additional information, it demonstrates that the user knows all the secret values in the partial commitment R: the secret key α and the hidden attributes $(\alpha_i)_{i \notin \mathcal{D}}$. Consequently, the verifier gets convinced that the user possesses the credential $Cred$ that contains the revealed attributes $(\alpha_i)_{i \in \mathcal{D}}$.

3.2 Selective Disclosure with Idemix

Idemix [7,8,11] relies on the Strong RSA assumption. Informally, in a group where this assumption holds, it is computationally infeasible to find *any* root of a given group element. (The RSA assumption is weaker and therefore it provides stronger security as it assumes that a *given* root of a given element is hard to find.)

Unlike in U-Prove, users in Idemix need only one secret key corresponding to *all* credentials and it is called a master key. Optionally, a user may choose to use several master keys but that does not influence the security and privacy properties of the scheme. Furthermore, the Idemix selective disclosure is a hybrid scheme in the sense that it blends all three unlinkability approaches we discussed in Section 2.2: the issuing is a blind signature that does not allow the issuer to learn the master key and the resulting signature; the user can partly randomise a credential in the verification protocol; the user proves to the verifier using zero-knowledge techniques that she possesses a valid credential instead of releasing any information about it.

Idemix's algorithms and parameters can be described as follows (for a detailed description we refer to [11]):

- A group has to be generated in which the Strong RSA problem is hard:
  - Given a special RSA modulus: $n = pq$ where $p = 2p' + 1$, $q = 2q' + 1, p'$, and q' are large prime numbers. The set QR_n of quadratic residues $\pmod{n}$ (i.e. every element r for which there exists some integer x such that $x^2 \equiv r \pmod{n}$) establishes a Strong RSA group.
- Key generation for the issuer and the generation of the system group happen to be the same algorithm here. The reason for that is that the prime components p, q of n determine the group uniquely and they form the issuer's secret key as well.
  - Generate a special RSA modulus $n = pq$. p and q are kept secret as the secret key sk_I of the issuer.
  - Choose, uniformly at random, $R_0, R_1, \ldots, R_l, Z, S \in QR_n$. These values form the set of public parameters together with n.
- Generation of a master key for the user.
  - The user chooses randomly a large integer α (in our case 256 bits) as his master key.

– Issuing a credential. Like a U-Prove credential, an Idemix credential is also a signature on a commitment. The issuing procedure is an interactive protocol between the issuer and the user. Both participants need their secret keys, sk_I and α.
  - The resulting credential includes a commitment $R = R_0^{\alpha} \prod_1^l R_i^{\alpha_i} \pmod n$ to the attributes $\alpha_1, \ldots, \alpha_l$ and the issuer's signature $A = \left(\frac{Z}{S^v \cdot R}\right)^{1/e} \pmod n$ on the commitment. A can only be computed by knowing the divisors of n, that is, by knowing sk_I; therefore, it can only be created by the issuer. The resulting signature is (A, e, v) where e is a prime number chosen by the issuer and v is computed together by the participants but known only to the user. (So, the issuer can store A and e but he doesn't learn α and v.)
– Selective disclosure of attributes is achieved using an interactive protocol between a user and a verifier.
  - Without giving all the details about credential showing, we show how a signature can be randomised. Let r be a random integer from a certain interval. Provided that $A' := A \cdot S^{-r} \pmod n$ and $v' := v + er$,

$$A'^e S^{v'} R \equiv A^e S^{-er} S^v S^{er} R \equiv A^e S^v R \equiv Z \pmod n.$$

    Therefore, a randomised signature (A', e, v') is also a valid signature on R. Note that e should not be revealed during a verification protocol as it provided linkability.
  - Besides the revealed attributes, a user sends A' from the randomised signature and a proof of knowledge pr of all undisclosed attributes and required parameters to the verifier.

$$pr = PK\{(e', v', \alpha, (\alpha_i)_{i \notin \mathcal{D}}) : Z \cdot \left(R_i^{-a_i}\right)_{i \in \mathcal{D}} \equiv \pm A'^{e'} S^{v'} R_0^{\alpha} \prod_{i \notin \mathcal{D}} R_i^{a_i} \bmod n\}$$

  - The verifier checks the validity of the proof pr. (Actually, the prover proves knowledge of a representation of $Z \cdot \left(R_i^{-a_i}\right)_{i \in \mathcal{D}}$ with respect to A', S, R_0, and $(R_i)_{i \in \mathcal{D}}$.)

3.3 Similarities and Differences

As we saw in the previous sections, there are many cryptographic differences between U-Prove and Idemix. Not only do the cryptographic assumptions differ in the two schemes, but also the principle of verification. While a U-Prove credential is revealed every time it is used, an Idemix credential is never revealed to a verifier. Therefore, unlinkability can only be achieved using as many different U-Prove credentials as their verification instances. The same functionality however, can be achieved with only one Idemix credential. Furthermore, while U-prove credentials have different random secret keys calculated during each issuing protocol, Idemix credentials belonging to the same user can operate with one master key. As an application of this feature, several credentials having the same secret key can be proved to belong to the same user.

Besides these differences, there are similarities as well. Both credentials can be abstractly seen as signed commitments. A commitment in this case is basically the product of generators with attributes and a secret key as their exponents. As a result of this resemblance, selective disclosure can be done similarly. Having received some attributes, the verifier can reproduce a partial product from the commitment. Then he can combine it with the presented proof of knowledge about all the other attributes. Only a user having a valid credential can provide such a proof.

4 Idemix on Smart Cards

The main objective of our research is to assess how well privacy-friendly protocols perform when run on a smart card. Hence implementing our prototypes requires an open smart card platform that also provides the necessary cryptographic hardware support. Previous research [13,9] clearly shows that purely software based prototypes do not offer sufficient performance for realistic use, while proper hardware support makes the prototypes practically usable. In this case we used an Infineon SLE78 chip[4] running the MULTOS platform. This platform offers an API which exposes the (hardware supported) modular arithmetic operations required to implement technologies like Idemix and U-Prove.

4.1 Smart Cards

Regardless of the software platform operating the card, all smart cards provide the same external functionality. A smart card is an embedded device that communicates with the environment through Application Protocol Data Units (APDUs) – byte arrays formatted according to the ISO7816 specification [1]. Most notably, the APDUs constrain the communication payload to roughly 256 bytes in each direction for a single APDU exchange. The permanent storage of the card (EEPROM memory) is considered highly secure, accessible only through the APDU commands offered by the application, which in turn are subject to any authentication and secure messaging requirements that the card application may impose.

4.2 The MULTOS Platform

The goal of the MULTOS platform is to provide a secure hardware independent execution platform for smart cards. To this end, they developed a specification for the execution and memory models, explained in more detail below, that all MULTOS implementations must provide. Besides this mandatory part of the specification there are also a number of optional elements, mostly concerning cryptographic functionality that may or may not be available on a specific hardware platform. An overview of which functionality is provided by which card can be found in the MULTOS Implementation Reference [2][5].

[4] This is the successor to the Infineon SLE66 chip used by Mostowski and Vullers [9].
[5] Our card has an M3 generic (ML3-36K-R1) implementation by Multos International.

Execution Model Applications on a MULTOS card are executed in a virtual machine, called the Application Abstract Machine (AAM). The functionality of this virtual machine is defined by the MULTOS specification to assure that applications are portable, that is, independent of the actual chip used[6]. The AAM is a stack machine that interprets instructions from the MULTOS Executable Language (MEL).

Memory Model The AAM virtual machine provides each application with its own memory space. Within an application the code space, residing on non-volatile EEPROM storage, and data space, divided over EEPROM (for persistent storage) and volatile RAM, are handled independently of each other. Code is executed while data is manipulated. The memory of an application is protected by a strong firewall. This means that applications cannot access each others memory. The data of an application is divided over three distinct memory areas, listed below.

Static memory is the non-volatile storage for an application. It is private to the application and cannot be accessed by the terminal or any other application. MULTOS offers mechanisms to avoid corruption of the static memory area such that this data remains consistent.

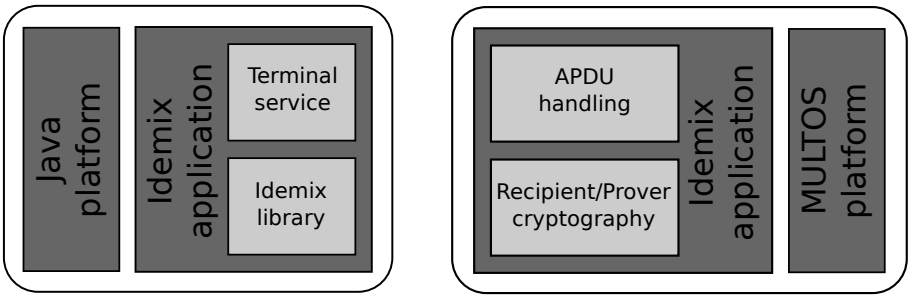
Public memory is the volatile input/output buffer for an application. Incoming command APDUs are held in public memory and outgoing response APDUs are placed here. This buffer is also used to pass information from one application to another when delegation is used. MULTOS guarantees that data in this memory remains private to the running application until it exits or delegates to another application. This means that it can be used as temporary workspace.

Dynamic memory is the volatile storage for an application. It is used to store session data, if any. The size of the session data area is fixed when an application is loaded onto a card and it depends on the amount of variables declared. Furthermore, the dynamic memory contains the stack, which is the application's work area. As mentioned before, the AAM operates as a stack machine, which means that this memory area is used to perform many functions (and provide input for these functions). The maximum size of the stack is fixed by the amount of dynamic memory available. Therefore, applications will need to ensure that their use of dynamic memory does not exceed the limit imposed by the chip [2] on which the application will reside.

4.3 System Architecture

Our system consists of two parts, depicted in Figure 2, a terminal (a) which interacts with a card (b) using APDUs. The terminal application is written in

[6] Application portability can be limited due to specific memory requirements or dependencies on optional parts of the MULTOS specification.



(a) Terminal (b) Card

Figure 2. System architecture for Idemix on a smart card.

Java and uses the Idemix cryptographic library[7] provided by IBM Research. We created an extension to this library, the terminal service, which takes care of all smart card specifics. The service implements the user roles of the Idemix issuance and verification protocols, described by the `Recipient` and `Prover` interfaces respectively. These interfaces are implemented by translating all Java method calls from the library into the corresponding APDU commands and converting the Java data types to raw byte arrays, suitable for APDU communication.

The Idemix application on the card takes care of handling the incoming APDUs and storing the values into the internal data structures. While handling the communication with the terminal is the largest part of the application, the main part is the implementation of the cryptographic operations for the Idemix protocols. This allows the card to perform the user roles without depending on the terminal for any computations or proof generation. The only thing the terminal is responsible for is providing the data in the correct format.

4.4 Smart Card Implementation

Just as Mostowski and Vullers [9] with their U-Prove implementation, we have chosen to use the MULTOS C interface to develop our prototype implementation of Idemix. The programming environment is convenient for smart card programming and allows us some more flexibility for memory management. It will be explained below why this is crucial.

Implementing the Idemix specification did not turn out to be that hard in the beginning, the available API makes it easy to implement the cryptographic protocol. In principle, it is a direct translation from the mathematical description to API calls. The only API restriction we came across was that the `ModularExponentiation` function does not accept exponents larger than the modulus size. For Idemix this only involves exponentiations with base S (see details in Section 3.2). Hence we added a function `SpecialModularExponentiation`

[7] Available for download at `https://prime.inf.tu-dresden.de/idemix/`

which implements the same method as used by Bichsel et al. [5], that is, splitting one exponentiation up into two[8] exponentiations and one multiplication.

Our initial implementation only used static memory to store the variables. This allowed us to work without thinking about which memory segment to use. The drawback of this approach was a bad performance caused by the EEPROM memory which takes a long time, compared to RAM, to write new values.

Once we had a functionally correct implementation, we started to optimise. This was done by moving the buffer, which stores the intermediate results for larger computations, to the public memory. The next step was to move the session variables to the dynamic memory, which required careful organisation due to the limited amount of available storage. However, when the dynamic memory use increased the stack-based execution model started to cause trouble.

Initially, the stack could use the full size of the dynamic memory, such that we had sufficient space to use functions and put the (relatively) large input values on the stack. When using the C-interface, the compiler takes care of managing the stack and putting input values on it when an instruction needs this. However, this makes it difficult to get an idea on how much space the stack actually needs. By trial and error, we discovered that we used quite some amount of memory for the stack, which left us with only limited amount of space for session variables.

To improve this situation, we reduced the number of function calls by inlining some convenience functions. We also switched to using global variables instead of function parameters, such that when we use a function, it does not require much space on the stack. To get the last few, often used, variables into RAM, we decided to split up some computations into smaller parts such that the values to be put on the stack also get smaller. For example, additions can be computed using addition with carry, and multiplications using grade-school multiplication. This adds a number of extra operations, but it is worth the memory gain which results in improved performance.

Finally, we translated most parts of the cryptographic code to MEL assembly which allowed us to optimise the use of stack and reduce the amount of memory operations during calculations. This was required since the provided C-functions moved the values from the stack to the variable locations, while they are needed again in the next operation. By doing this we could reduce the amount of memory required for intermediate values which in the end allowed us to get all necessary variables in RAM.

The resulting code can be found in the `idemix_multos` GitHub repository at `https://github.com/credentials/`. The repositories listed at this page are used within the IRMA project (`https://irmacard.org/`) which aims to develop an infrastructure for using attribute-based credentials on smart cards. A pilot, with government support, will be launched early 2013.

[8] This method actually requires three exponentiations, but we can precompute one exponentiation during initialisation, since we only need this method for the base S.

5 Results and Comparison

We mainly compare our results with the U-Prove implementation by Mostowski and Vullers [9] as this implementation offers similar functionality using the same modulus size (1024 bits) and the same platform. During initial development we even used the same SLE66-based cards, but in the end we managed to get newer SLE78-based cards which we currently use. For the conclusions this makes no difference since already with our SLE66 optimised prototype we got similar results, the SLE78 implementation only improved the running times.

There are two important performance measures: the time it takes to issue a new credential to the card, and the running time of the verification protocol.

5.1 Credential Issuance

For issuance, we measured the running time of the protocol and determined how much time was actually spent on computations and which part was used to transfer and store the values (marked as overhead; see Figure 3). These values offer a clear improvement over the U-Prove issuance times from [9], which take 3623 and 5489 ms for 2 and 5 attributes respectively. Not only are the absolute values better, the additional time it takes to issue more attributes is less than with the U-Prove implementation.

Sterckx et al. [12] implement the Direct Anonymous Attestation (DAA) protocol, which is derived from the Idemix protocols on a Java Card. With a 1024 bits modulus they achieve a running time of 2.4 seconds of which 19% is overhead, which gives a computation time of approximately 1.9 seconds. This is good, but unfortunately the DAA protocol does not support any attributes as it is just targeted at anonymous authentication and hence only uses a secret key.

Bichsel et al. [5] from the Idemix team at IBM Research Zürch also implemented a variant of the DAA protocol on a Java Card. They report a running time of 7.4 seconds for a modulus size of 1280 bits, which is larger than the 1024 bits we used. It is unclear, however, which transaction time they measured. But again, this implementation does not include any attributes.

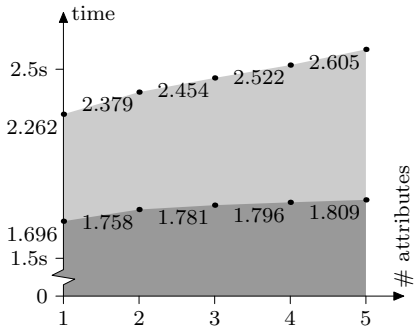


Figure 3. Credential issuance times (■: computations, ■: overhead; 1024 bits).

5.2 Selective Disclosure of Attributes

For selective disclosure, we measured the running times of four configurations (see Figure 4). These configurations have been chosen because two is the smallest number of attributes for which selective disclosure makes sense and five is the largest number of attributes used in our pilot project. By comparing these graphs, it is clear that each attribute that is not disclosed adds roughly 100ms of computation time, whereas the amount of work per undisclosed attribute remains the same.

Comparing these measurements with those from the U-Prove implementation [9], it is clear that U-Prove is more efficient with total running times below 900ms and less. At this point the multi-show unlinkability property of the Idemix technology has an effect on the performance. To achieve this property, the issuer signature is partly randomised and for the remaining part proved using a zero-

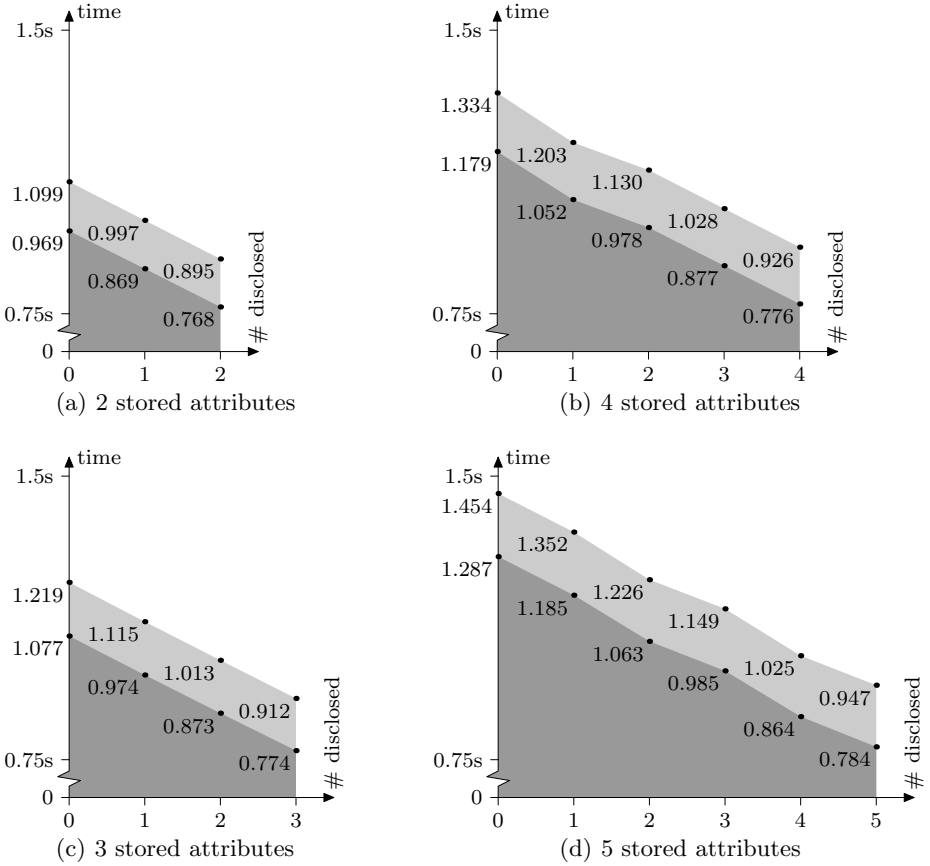


Figure 4. Attribute proving times (: computation, : overhead; 1024 bits modulus).

knowledge proof. This clearly has its drawback compared to U-Prove that only requires computation to hide the undisclosed attributes.

While the multi-show unlinkability property has a negative effect on the running time for Idemix, it requires less storage on the card then U-Prove. This is due to the fact that the card has to store multiple U-Prove tokens to achieve multi-show unlinkability. Given that storage space is rather limited on smart cards[9], this is a huge benefit of the Idemix technology. In the end this integrated multi-show unlinkability property also settled our debate for using Idemix in favour of U-Prove as the technology to be used within the IRMA project.

Comparing our work with the DAA implementations by Sterckx et al. [12] and Bichsel et al. [5] makes no real sense since they do not offer selective disclosure of attributes. It is, however, clear that our implementation provides a performance improvement over these implementations since we can even hide the secret key and two attributes in 1.1 seconds whereas Sterckx et al. already need 4.2 seconds to hide only the secret key.

6 Final Remarks

In this paper we demonstrated that Idemix's selective disclosure can be efficiently implemented on a smart card. Although the running time of the issuing and verification protocols restricts the range of use cases, we can be optimistic already about these results.

- Additional attributes only increases the running time with a small factor.
- A single credential is sufficient to preserve both unlinkability properties.

Our implementation only offers a 1024 bits security level. We have chosen this modulus size since it is lowest acceptable level security wise, whereas it is the highest acceptable level performance wise. Mostowski and Vullers [9] already showed that using a 2048 bits modulus more than doubles the computation time for the primitive operations. This is not even taking the shortage of RAM, due to larger values, into account.

To the best of our knowledge the only other smart card implementation of attribute-based credentials besides what was already mentioned is by Batina et al. [4] who implement Verheul's self-blindable credentials [15]. Their implementation is faster than our Idemix implementation and also offers multi-show unlinkability. However, their credentials only contain a single binary attribute as explained in Section 2.2, and hence do not provide selective disclosure similar to Idemix or U-Prove.

Due to the multi-show unlinkability feature of the Idemix protocol this implementation has been selected to be used in a pilot project. The goal of this project is to gain more experience in actually using these kinds of privacy-preserving technologies and in their usability features on smart cards.

[9] A typical smart card only has 36 to 144 KB of EEPROM available for storing application data.

Acknowledgements We are grateful to Patrik Bichsel, for making the necessary modifications to the Idemix library, to Wouter Lueks, for his help with the performance tests, and to Bart Jacobs, Jaap-Henk Hoepman and the anonymous reviewers for their valuable comments which helped to improve this work.

References

1. ISO 7816-4 Identification cards – Integrated circuit cards – Part 4: Organization, security and commands for interchange. ISO, Geneva, Switzerland (2005)
2. MULTOS implementation report. Tech. Rep. MAO-DOC-TEC-010 v2.4, MAOSCO Limited (2012)
3. Baldimtsi, F., Lysyanskaya, A.: Anonymous credentials light. IACR Cryptology ePrint Archive 2012, 298 (2012)
4. Batina, L., Hoepman, J.H., Jacobs, B., Mostowski, W., Vullers, P.: Developing efficient blinded attribute certificates on smart cards via pairings. In: Gollmann, D., Lanet, J.L. (eds.) CARDIS 2010. LNCS, vol. 6035, pp. 209–222. Springer-Verlag (April 2010)
5. Bichsel, P., Camenisch, J., Groß, T., Shoup, V.: Anonymous credentials on a standard Java Card. In: CCS 2009. pp. 600–610. ACM (November 2009)
6. Brands, S.A.: Rethinking Public Key Infrastructures and Digital Certificates: Building in Privacy. MIT Press, Cambridge, MA, USA (2000)
7. Camenisch, J., Lysyanskaya, A.: An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 93–118. Springer Berlin / Heidelberg (2001)
8. Camenisch, J., Lysyanskaya, A.: A signature scheme with efficient protocols. In: Cimato, S., Persiano, G., Galdi, C. (eds.) Security in Communication Networks. LNCS, vol. 2576, pp. 268–289. Springer Berlin / Heidelberg (2003)
9. Mostowski, W., Vullers, P.: Efficient U-Prove implementation for anonymous credentials on smart cards. In: Kesidis, G., Wang, H. (eds.) SecureComm 2011. LNICST, vol. 96, pp. 243–260. Springer-Verlag (2011)
10. Paquin, C.: U-Prove cryptographic specification v1.1. Tech. rep., Microsoft Corporation (February 2011)
11. Security Team, I.R.Z.: Specification of the Identity Mixer cryptographic library, version 2.3.4. Tech. rep., IBM Research, Zürich (Feb 2012)
12. Sterckx, M., Gierlichs, B., Preneel, B., Verbauwhede, I.: Efficient implementation of anonymous credentials on Java Card smart cards. In: WIFS 2009. pp. 106–110. IEEE (September 2009)
13. Tews, H., Jacobs, B.: Performance issues of selective disclosure and blinded issuing protocols on Java Card. In: Markowitch, O., Bilas, A., Hoepman, J.H., Mitchell, C., Quisquater, J.J. (eds.) WISTP 2009. LNCS, vol. 5746, pp. 95–111. Springer-Verlag (September 2009)
14. Verheul, E.: Self-blindable credential certificates from the weil pairing. In: Boyd, C. (ed.) ASIACRYPT 2001. Lecture Notes in Computer Science, vol. 2248, pp. 533–551. Springer Berlin / Heidelberg (2001)
15. Verheul, E.R.: Self-blindable credential certificates from the Weil pairing. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 533–550. Springer-Verlag (December 2001)