Data-minimizing Authentication goes Mobile

Patrik Bichsel[1], Jan Camenisch[1], Bart De Decker[2],
Jorn Lapon[3], Vincent Naessens[3], and Dieter Sommer[1]

[1] IBM Research – Zurich, Switzerland, *Email:* {pbi,jca,dso}@zurich.ibm.com
[2] IBBT - Distrinet – KU Leuven, Belgium, *Email:* bart.dedecker@cs.kuleuven.be
[3] MSEC – KaHo Sint Lieven, Belgium, *Email:*
{jorn.lapon,vincent.naessens}@kahosl.be

Abstract. Authentication is a prerequisite for proper access control to many e-services. Often, it is carried out by identifying the user, while generally, verification of certified attributes would suffice. Even worse, this kind of authentication makes all the user's transactions linkable and discloses an excessive amount of personal information, and thus erodes the user's privacy. This is in clear contradiction to the data minimization principle put forth in the European data protection legislation.
In this paper, we present *data-minimizing mobile authentication*, which is a kind of attribute-based authentication through the use of anonymous credentials, thereby revealing substantially less personal information about the user. We describe two typical scenarios, design an architecture, and discuss a prototype implemented on a smart phone which minimizes the disclosure of personal data in a user-to-terminal authentication setting. The prototype uses the Identity Mixer anonymous credential system (Idemix) and realizes short-range communication between the smart phone and the terminal using visual channels over which QR codes are exchanged. Furthermore, the security has been improved and unauthorized sharing of credentials prevented by storing the credentials' secret key in a secure element hosted by the mobile phone. Our measurements show that the use of smart phones for data-minimizing authentication can be an actual "game changer" for a broad deployment of anonymous credential systems.

Key words: privacy, anonymous credential systems, mobile computing, Android, secure element, visual communication channel

1 Introduction

In today's service-oriented world, users regularly have to authenticate to a clerk, a terminal, a door, or an on-line service to gain access to a desired resource. Most often, this happens using a physical credential (e.g., a badge), or a user name and password, or an X.509 certificate and the corresponding private key. In all these cases, authentication leads to the *identification* of the user, which is often not required by the underlying business processes (e.g., to access a newspaper site, it suffices that the site is convinced that the user has a valid subscription and does not need to know the identity of the user). These traditional authentication protocols also have the adverse effect that multiple authentications with the same (or even different) service providers are linkable and allows for compiling extensive user profiles.

Many of the scenarios that use identity-based access control could be adapted to use *attribute-based access control* instead, that is, the authorization to access a service is based on proven properties of the user's attributes that are certified by a trusted identity provider.

An example is age verification of a customer who wants to buy an alcoholic beverage in a bar. It should suffice for her to prove to the bartender that she was born more than 16 years ago, instead of having to show her identity card, thereby disclosing not only her exact date of birth, but also other identifying information such as her name or address.

Anonymous credential systems [12, 8, 7, 4] can be used to realize such attribute-based authentication. They allow a user to selectively release information about the attributes embedded in a credential or even combine multiple credentials for making more complex identity statements. For instance, as in the example above, the birthdate attribute can be used to prove to be older than 16, without revealing the birthdate itself. In the sequel of this article, we will focus on anonymous credential systems in which multiple interactions performed with such credentials are unlinkable [12, 8, 2], unless the released attribute information makes them linkable (e.g., a unique serial number or a unique combination of attribute values). Credential systems are, from a privacy point of view, the most suitable technology for realizing attribute-based authentication. A drawback of credential systems is that they require substantially more computational effort than traditional authentication technologies. However, today's most advanced mobile phones, the so called smart phones, are computationally sufficiently powerful to execute anonymous credential protocols in a reasonable time. They also have sufficient memory to store all the user's credentials, and can optionally be extended with a secure element (e.g., a smart microSD card) to protect the corresponding user's secrets. Smart phones are particularly well-suited as host platform for credential protocols because they are usually kept around, they allow for realizing intuitive graphical user interfaces, and they can connect to other devices via short-range communication channels. Currently, ongoing developments in the area of trusted execution environments go into the direction of strengthening future smart phones to make them even better suited for hosting the credentials of a user.

In this paper we illustrate how mobile devices can facilitate attribute-based authentication of a user sitting in front of a terminal. Users will authenticate with their mobile phone to an authorization authority with the restriction that only the user sitting in front of the terminal is able to perform the authentication. We denote these scenarios as *user-to-terminal authentication*. Some real-world examples are: the age verification of customers in a bar when they order alcoholic drinks, access control to the premises of one's employer, or to an event for which one has to acquire a ticket. In all cases, it is important that no third party can perform the authentication instead of the user at the terminal.

Contributions and paper structure. In this paper, we describe two scenarios for attribute-based authentication with a mobile device. We present **protocols** that make use of short-range channels between the user's smart phone and a terminal to establish an authenticated channel between the mobile and an authorization server (Sec. 3). The mobile will perform attribute-based authentication by using the Identity Mixer credential system. The use of short-range communication channels ensures that only the device in front of the terminal can execute the protocol. We present a **system architecture** for realizing those authentication protocols using visual short-range communication channels based on QR-codes (Sec. 4). The architecture also supports a **secure element** (smart microSD card) for handling secret key material throughout their life cycle, that is, storing them and performing all computations involving them. A prototype system has been built on an Android smart phone. We present **measurements** of the key metrics that determine protocol execution times (Sec. 5); the results are very encouraging so that a practical application of anonymous authentication

technologies on standard smart phones can be considered feasible. We conclude in Section 6 and provide an outlook on future research that is required in this area.

Related Work. Chari et al. [11] presented a taxonomy on different m-commerce scenarios with mobile devices. Our user-to-terminal scenarios fit in their *Kiosk-Centric* and *Full Connectivity* model, respectively. Similarly, our solution also fits in the model of Claessens [13], in which the mobile is combined with a workstation in order to have a higher degree of security and mobility.

Many schemes have been presented that use physically constrained channels [18, 22, 19, 16, 23, 25] in order to obtain a secure communication channel between two nearby devices (e.g., audio, motion, QR codes or radio profiling). Although they are often related to device pairing, they also apply to our user-to-terminal authentication. However, privacy is often of a lesser concern (e.g., for device pairing). In our solution, we combine the short-range communication channel with a more privacy-friendly authentication mechanism, namely anonymous credentials, partially implemented on a secure element.

Bichsel et al. [3] presented a full Java card implementation of anonymous credentials similar to the ones used in the Identity Mixer library. Showing a credential takes about 7.4 s for a 1280-bit modulus. Other implementations require partial trust on the host. Sterckx et al. [24] made an implementation of the DAA protocol which is closely related to Identity Mixer credentials, taking about 4.2 s for a credential show using a 1024-bit modulus. Danes [14] presented another approach, similar to ours, in which only the master secret is kept on the card. We present a solution based on a proof of knowledge of a discrete logarithm in hidden order groups.

There are also prototypes implementing U-Prove [4] anonymous credentials, which take about 0.5 s for showing a credential [20]. Note that in order to remain unlinkable, the U-Prove system requires the issuance of a new credential for each transaction, which may quickly exhaust the EEPROM of the card [3] and also relativizes the seemingly good performance.

2 Requirements

In the introduction, we argued why the user's smart phone is an ideal device for storing the user's credentials. First, we list its major advantages:

- the mobile device has ample storage capacity;
- it has sufficient processing power;
- it is equipped with a camera;
- it has several means of communicating;
- it can often be extended with a secure element (smart μSD card);
- it has a high resolution screen that can realize user-friendly interfaces;
- it is kept around all the time, hence, it is almost always available.

In this paper, we focus on anonymous Idemix credentials, but other kinds of credentials can also be supported although they do not offer the same level of privacy for the user and the same level of assurance for the relying party.

The requirements can be summarized as follows:

1. authentication should only happen with the user's consent;
2. authentication should only be possible by the user in front of the terminal;
3. sharing of credentials (e.g., with friends) should be impossible;

4. when the smart phone is lost or stolen, the credentials should no longer be usable.

The *first requirement* demands that for every authentication, at least some **user interaction** is necessary (e.g., at least a click on an OK-button).

The *second requirement* is more difficult to realize. The authentication process needs to **involve a short-range communication channel** between the terminal and the mobile device. An ideal channel would be an NFC-channel. However, most smart phones do not yet support this kind of communication and terminals even less. Bluetooth is not really short-range, and WiFi certainly not. Therefore, our scenarios will use a **visual channel**: QR-codes will be displayed on the screen and scanned by the camera of the device at the other end of the channel. Since the camera must be positioned a few centimeters in front of the displayed QR-code, this way of exchanging information certainly realizes a short-range communication channel.

The *third requirement* can be realized by introducing a **secure element**. In our scenarios, we will assume that the smart phone has a free slot for a μSD card. A (secure) smart μSD card is used to store the user's secrets of the credentials. Every credential is associated with a secret; when using a credential, the user needs to prove knowledge of the corresponding secret. Since the user has no longer direct access to these secrets, the original Identity Mixer library that we use as anonymous credential system, had to be modified as follows:

- the user secret is generated on the smart μSD card and never leaves the card;
- the credential issue protocol and the credential show protocol have been adapted so that all computations (exponentiations) that involve the user secret are delegated to the smart μSD card.

Since one needs the user secret for showing a credential, and user secrets are kept in a secure element, it becomes impossible to share credentials.

For the *fourth requirement*, the use of the smart μSD card is further restricted: the card will only perform an operation if the correct **PIN-code** is given. Hence, when the mobile device is lost or stolen, the new owner will not be able to use the stored credentials since she cannot "activate" the card via the correct PIN-code.

3 Scenarios

In this section, we present two typical uses of the smart phone as a tool to prove the user's attributes:

- a customer wants to buy an alcoholic beverage from a vending machine and needs to prove that she is older than 16 (cf. Section 3.2);
- a user wants to access protected articles of a newspaper website and needs to prove that she is a genuine subscriber to that paper (cf. Section 3.3).

Before we discuss the protocols in detail, we first describe the notation used, the basic building blocks and the assumptions for these settings.

3.1 Notation, Building Blocks and Assumptions

In the scenarios, several different communication channels are used, going from secure connections, over visual connections to GUI-based connections. Table 1 lists the symbols used in the protocol descriptions and their meaning. The arrows show the direction in which information is exchanged; a double arrow means that in this interaction, information is exchanged in both directions (possibly in several rounds). Sets are represented by $\{\ldots\}$.

Table 1. Meaning of symbols used in the protocol descriptions

Symbol	Meaning
$\rightarrow$, $\leftarrow$, $\leftrightarrow$	unprotected communication channel (e.g., a TCP-connection over a wired or wireless network)
$\Rightarrow$, $\Leftarrow$, $\Leftrightarrow$	secure communication channel (e.g., a wired connection that is inaccessible to external parties or a TLS/SSL-protected connection)
$\overset{QR}{\rightarrow}$, $\overset{QR}{\leftarrow}$	visual communication channel, where messages are exchanged via QR codes
$\overset{RF}{\rightarrow}$, $\overset{RF}{\leftarrow}$	unprotected radio communication channel (e.g., GPRS, WiFi, ...)
$\{\ldots\}$	set of items
π	policy
ϕ	statement or claim (e.g., the credential holder is older than 16 and has a valid subscription to the newspaper)
$\phi \vdash \pi$	statement ϕ fulfills the policy π
Ξ	payload added to a protocol message

We also use a few cryptographic building blocks:

Encryption schemes. Public key encryption is used for protecting both the confidentiality and integrity of messages. We will use the following functions to indicate public key encryption/decryption:

$$\textit{ciphertext} \leftarrow \mathsf{AsymEnc}(\textit{PK}, \textit{plaintext})$$
$$\textit{plaintext} \leftarrow \mathsf{AsymDec}(\textit{SK}, \textit{ciphertext})$$

where *PK* and *SK* represent the public and private keys, of which the public key, *PK*, can be certified in a certificate, Cert.

Anonymous credentials. In the protocols, signed proofs of knowledge are generated by the function:

$$\textit{SPK} \leftarrow \mathsf{ConstructSPK}(\pi, \mathscr{C}; \textit{PIN})\{\textit{msg}\}.$$

It has four parameters: a policy, π, which specifies what has to be proven, a set of credentials, $\mathscr{C}$, the PIN-code, *PIN*, for activating the secure element[4], and the message, *msg*, to be

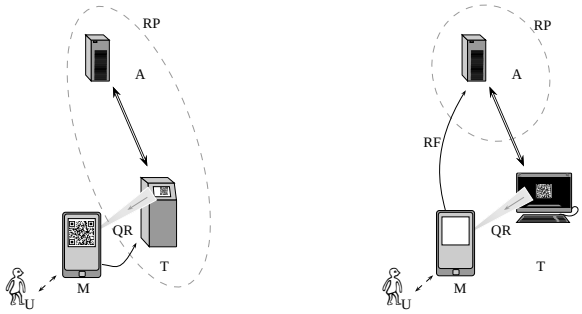
[4] The secure element, a smart μSD card, holds all the credentials' secrets and executes the necessary operations with these secrets to generate the required signatures. The PIN code is necessary to activate the card, so that unauthorized usage is prevented. If no secure element is used, the PIN can be omitted. This, however, requires substantially more trust in the mobile device.

signed. The signature can be verified with

Verify(*SPK*, *info*)

where *SPK* is the signed proof of knowledge and *info* extra information necessary for the verification (e.g., a reference to the session or selected policy).

Roles. Fig. 1 depicts the roles active during user-to-terminal authentication. A user U communicates through his mobile device M with a terminal T. The terminal communicates with the authorization authority A. In Scenario 1, the relying party (i.e., the vending machine) contains both the terminal and the authorization authority. In contrast, in Scenario 2 the terminal is not trusted by the relying party (i.e., the website). Hence, the authorization authority coincides with the relying party.



(a) Scenario 1 (b) Scenario 2

Fig. 1. Roles and Setup for *Scenario 1: a vending machine* and *Scenario 2: a newspaper website*.

Assumptions. If the connection between T and A is unprotected or T is not controlled by the RP, an SSL/TLS-channel will be set up between T and A. We will omit this in the protocol descriptions.

3.2 Scenario 1: a Vending Machine

As presented in Fig. (a), in the first scenario a vending machine for soft drinks and alcoholic beverages will only sell alcohol to customers older than a certain age (e.g., 16 years). The main goal is to prevent that youngsters buy alcoholic beverages *without the consent of an adult*, so that the machine can operate unattended (even in playgrounds or in schools), while complying to regulations.

We assume that the vending machine is equipped with a (small) screen of sufficient resolution (to be able to show QR codes) and a camera able to scan QR-codes. The vending machine contains both the terminal T and the authorization server A. We do not assume that A is online; hence, revocation of credentials will not have an instantaneous effect. However, each time the machine is replenished, the revocation information can be uploaded into the machine.

Table 2 describes the protocol in detail. When the customer selects an alcoholic beverage, RP needs to verify the age of the buyer. Therefore, it sends a nonce (N_{RP}), its identity ($\mathrm{ID}_{\mathrm{RP}}$)

and its policy (π_{RP}) to the terminal T (1). T encodes these items into a QR-code and displays it on its screen (2). The customer scans the QR-code with her smart phone M (3-4). M then selects a set of credentials $\mathscr{C}$ that can be used to fulfill the policy π_{RP} thereby taking the user preferences[5] ψ_{U} into account (5). The smart phone M displays the policy[6] and credential selection to U in a user-friendly manner on its screen (6). The user U selects the claim(s) to prove (ϕ_{U}) and the credentials ($\mathscr{C}^*$) to use and enters her PIN-code to unlock the secure element (SE) (7). Then, the smart phone M and the secure element SE will execute together the signing protocol, thereby proving the claim(s) (ϕ_{U}) with the selected credentials ($\mathscr{C}^*$) and signing the nonce (N_{A}) and the identity ($\mathrm{ID}_{\mathrm{RP}}$) of the vending machine (8). Note that SE will only collaborate if the correct PIN-code has been entered by the user. The signature, claims and possible extra payload[7] (Ξ) is then encoded as a QR-code, displayed on the smart phone's screen and scanned by the vending machine's camera (9-11). The data is then forwarded to the authorization server A (9), which verifies whether the claims fulfill the policy and checks the validity of the signature (and possibly the payment) (12).

Table 2. Scenario 1: the vending machine

(1)			$\mathsf{T} \Leftarrow$	A	: N_{A}, $\mathrm{ID}_{\mathrm{RP}}$, π_{RP}
(2)			T		: $m_1 = \mathsf{EncodeQR}(N_{\mathrm{A}}, \mathrm{ID}_{\mathrm{RP}}, \pi_{\mathrm{RP}})$
(3)		M $\overset{\mathrm{QR}}{\Leftarrow}$	T		: m_1
(4)		M			: $\{N_{\mathrm{A}}, \mathrm{ID}_{\mathrm{RP}}, \pi_{\mathrm{RP}}\} = \mathsf{ScanQR}(m_1)$
(5)		M			: $\mathscr{C} = \mathsf{selectCreds}(\pi_{\mathrm{RP}}, \psi_{\mathrm{U}})$
(6)	U $\Leftarrow$	M			: Present $\pi_{\mathrm{RP}}, \mathscr{C}$
(7)	U $\Rightarrow$	M			: $\mathscr{C}^* \subset \mathscr{C}, \phi_{\mathrm{U}} \vdash \pi_{\mathrm{RP}}, \mathrm{PIN}_{\mathsf{SE}}$
(8)		M $\leftrightarrow$	SE		: $SPK_{\mathsf{U}} = \mathsf{ConstructSPK}(\phi_{\mathrm{U}}, \mathscr{C}^*; \mathrm{PIN}_{\mathsf{SE}})\{N_{\mathrm{A}}, \mathrm{ID}_{\mathrm{RP}}\}$
(9)		M			: $m_2 \leftarrow \mathsf{EncodeQR}(\phi_{\mathrm{U}}, SPK_{\mathsf{U}}, \Xi)$
(10)		M $\overset{\mathrm{QR}}{\Rightarrow}$	T		: m_2
(11)			T		: $\{\phi_{\mathrm{U}}, SPK_{\mathsf{U}}, \Xi\} \leftarrow \mathsf{ScanQR}(m_2)$
(12)			T $\Rightarrow$	A	: $\phi_{\mathrm{U}}, SPK_{\mathsf{U}}, \Xi$
(13)				A	: **if** (! ($\phi_{\mathrm{U}} \vdash \pi_{\mathrm{RP}}$) || ! Verify($SPK_{\mathsf{U}}$, $\{\phi_{\mathrm{U}}, N_{\mathrm{A}}\}$)) **abort**

Note that this scenario cannot prevent that youngsters buy alcoholic beverages by using the smart phone of an adult. However, this assumes a collusion between the youngster and the owner of the phone since the youngster will need the PIN-code to activate the secure element[8]. This is similar to the situation where the adult buys the beverage and gives it to the youngster. Also, a relay-attack can be arranged: the youngster's smart phone could relay the machine's nonce, ID and policy to the smart phone of an adult (e.g., via a Bluetooth connection), which then generates the required signature and returns it to the child's phone.

[5] The user preferences specify which credentials can be used in which transactions. They can also restrict the kind of claims that can be proven with each credential.

[6] The policy may offer different options: e.g., the user can prove that she is older than 16 by using an eID credential and proving that her birth happened more than 16 years ago or by using a driver license credential and simply proving that she possesses a valid one.

[7] E.g. if the smart phonephone is capable of doing payments, payment information can be added here.

[8] Without a secure element, the youngster could surreptitiously "borrow" the phone of an adult, and buy the alcoholic beverage unnoticed.

Again, this assumes a collusion: additional software needs to be installed on the adult's phone and signatures will only be generated after entering the PIN-code (which suggests the adult's consent)[9]. Also, the use of a Bluetooth connection assumes the adult's proximity.

3.3 Scenario 2: a Newspaper Website

In this scenario (see Fig. (b)), the user, sitting behind his laptop or desktop, visits the website of a newspaper and wants to access a protected article. The web server wants to verify whether the visitor is currently subscribed to that newspaper. We assume that subscribers possess an anonymous credential which lists the subscription period. The main goal is to prevent that subscriptions are shared among friends so that different users can access the articles *simultaneously*.

The differences with the previous scenario are:

- the terminal T is under control of the user or an external entity;
- the web server coincides with the authorization server;
- T may not have a webcam; hence, T may not be able to scan a QR-code;
- the smart phone M has Internet access via a radio channel (e.g., GPRS, WiFi, ...)

Table 3 describes the protocol in detail. We assume that a secure communication channel has been set up between the terminal T (e.g., an applet within the browser) and the authorization server A.

In step (1), A sends a nonce, its policy and its certificate Cert_{RP} to T. The certificate contains besides ID_{RP} also the public key of RP. Steps (2)-(8) are similar to those in scenario 1. Next, the smart phone M encrypts the user's claims (ϕ_{U}), the attribute-based signature (SPK_U) and a possible payload with the public key of the relying party (9). The ciphertext is then sent directly (via a radio channel) to the authorization server A (10), which will decrypt the ciphertext (11) and verify the claim(s) and the signature (12).

Table 3. Scenario 2: Access to a newspaper website

(1)			T $\Leftarrow$	A	: N_{A}, π, Cert_{RP}
(2)			T		: $m_1 = \mathsf{EncodeQR}(N_{\text{A}}, \text{Cert}_{\text{RP}}, \pi_{\text{RP}})$
(3)		M $\overset{\text{QR}}{\Leftarrow}$	T		: m_1
(4)		M			: $\{N_{\text{A}}, \text{Cert}_{\text{RP}}, \pi_{\text{RP}}\} = \mathsf{ScanQR}(m_1)$
(5)		M			: $\mathscr{C} = \mathsf{selectCreds}(\pi_{\text{RP}}, \psi_{\text{U}})$
(6)	U $\Leftarrow$	M			: Present $\pi, \mathscr{C}$
(7)	U $\Rightarrow$	M			: $\mathscr{C}^* \subset \mathscr{C}, \phi_{\text{U}} \vdash \pi_{\text{RP}}, \text{PIN}_{\text{SE}}$
(8)		M $\leftrightarrow$	SE		: $SPK_{\text{U}} = \mathsf{ConstructSPK}(\phi_{\text{U}}, \mathscr{C}^*; \text{PIN}_{\text{SE}})\{N_{\text{A}}, \text{Cert}_{\text{RP}}.\text{SUBJECT}\}$
(9)		M			: $m_2 = \mathsf{AsymEnc}(\text{Cert}_{\text{RP}}.\text{PK}, \{\phi_{\text{U}}, SPK_{\text{U}}, \Xi\})$
(10)		M $\overset{\text{RF}}{\Rightarrow}$		A	: m_2
(11)				A	: $\{\phi_{\text{U}}, SPK_{\text{U}}, \Xi\} = \mathsf{AsymDec}(SK_{\text{RP}}, m_2)$
(12)				A	: **if** (! ($\phi_{\text{U}} \vdash \pi_{\text{RP}}$) || ! Verify($SPK_{\text{U}}$, $\{\phi_{\text{U}}, N_{\text{A}}\}$)) **abort**

[9] When the adult's smart phone is also infected with a key-logger, the PIN-code could have been captured in a previous interaction with the secure element. Although not impossible, we deem that the youngster being able to buy the alcoholic beverage this way highly unlikely, since the attack is only possible when the adult's smart phone is nearby.

The nonce N_A serves two purposes: it prevents replay attacks and it also links the authentication interaction (over the RF-channel) to the connection between the terminal and the web server. This protocol ensures for the web server that someone with a valid subscription is sitting in front of the screen of the terminal. If the smart phone is equipped with a secure element (*SE*), a subscription credential can no longer be shared among friends.

3.4 Discussion

The previous two scenarios fulfill all the requirements listed in Section 2.

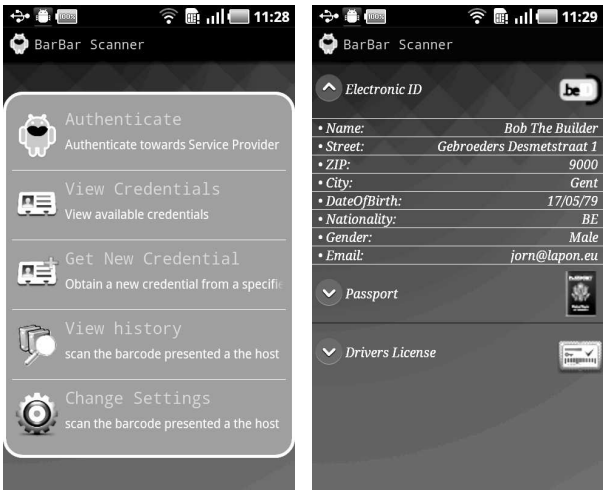
Both scenarios require the *user's consent*, since the user has to select which claim(s) to prove with which credentials. Also, a PIN-code must be entered by the user to activate the secure element (cf. line 7); hence, a lost or stolen smart phone will not allow the new owner to make use of the stored credentials. Malware can circumvent the required user interactions by automating them. A key logger can also capture the PIN-code, although this is less obvious when digits are entered via the smart phone's touch screen; the user interface can even shuffle the digits or substitute them with pictures to make logging extremely difficult.

Both scenarios also use a short-range communication channel (cf. line 3 and 8 in scenario 1, and line 3 in scenario 2). Therefore, a *visual channel* is used, over which QR-codes are exchanged. A QR-code can only be scanned when the camera is positioned a few centimeters in front of the screen displaying the code. This ensures that only the person positioned in front of the terminal can read the information. In the first scenario, a visual channel is used to return the signature to the terminal; in the second scenario, the signature is sent over an RF-channel, because the terminal might not be equipped with a webcam. The signature is based on information received via the (first) short-range channel. In the second scenario, the signature is encrypted with the relying party's public key. Also, the signature is based on $\mathrm{ID}_{\mathrm{RP}}$; hence, a phishing attack, where RP tries to reuse the user's authentication signature to get access to another server's (S) protected resources will fail, unless RP is able to get a trusted certificate with ID_S and PK_{RP}.

In both scenarios, we cannot exclude, with protocol mechanisms, a relay-attack, where the smart phone forwards the received information via the scanned QR-code to another smart phone. However, this assumes the consent of the owner of the other smart phone. Moreover, when Bluetooth is used for relaying the information, it also presumes the proximity of the other user. Although the relay-attack cannot be excluded, the authentication architecture described in this paper is certainly much stronger than the current practice, both in terms of the privacy of the user (where identification is replaced by attributed-based authentication), and in terms of assurance for the relying party (where the disclosed attributes or properties thereof, have been certified by a trusted identity provider). An additional advantage for the relying party is that it has to store less personal information of its clients; hence, less effort and budget has to be spent to protect this sensitive information as is required by legislation.

4 Architecture and Prototype

We have designed an architecture and implemented a prototype for validating our scenarios and showing the practical feasibility of our ideas. The architecture comprises the handheld device M, the terminal T, the authorization server A, and the identity provider IP as introduced in Section 3. We next elaborate on the general architectural concepts and give some details on the implementation. Figure 2 illustrates a part of the user interface of the prototype, namely the main menu and a listing of the available credentials stored on the mobile.



(a) (b)

Fig. 2. Screenshots of (a) the main menu and (b) a list of credentials.

Communication. The short-range communication channels are realized by visual channels over which QR-codes are exchanged. Most smart phones are equipped with a camera. Both the mobile device and the terminal have a screen on which QR-codes can be displayed. If the terminal possesses a webcam, a bi-directional visual channel can be realized; otherwise, only a visual channel from the terminal towards the mobile device is possible.

We used ZXing, a multi-format 1D/2D barcode image processing library in the implementation.

The architecture uses the REST [21] framework provided by RESTlet [17], a lightweight and comprehensive open source REST framework for the Java platform. It supports all major Internet transport protocols, data formats, and service description standards. The framework is extended with a new *connector type* to support the QR-based visual channel.

Access control. To support attribute-based authentication, a security framework was implemented for client and server entities. It includes three manager components: The *credential manager* implements the cryptographic protocols for issuing and showing credentials. It invokes technology-specific credential handlers. For this paper, we implemented a credential handler for the Identity Mixer Anonymous Credential System. The *storage manager* handles the storage and retrieval of credentials on the host device, and the *policy manager* handles service access policies. We use CARL [9] as formal language for specifying those policies. The CARL language offers adequate expressiveness to address advanced authentication requirements and allows for privacy-preserving, i.e., data minimizing, statements, while at the same time allowing for user accountability. The following example policy specifies that the service may be consumed by a requester owning a European electronic ID certifying that the requester's age is over 18 and with the eID not being expired:

01 own p :: eID issued-by EU-Gov
02 where $p.dateOfBirth \leq dateMinusYears(today(), 18)$

$$\wedge p.expDate > today()$$

In the RESTlet [17] framework each resource of a party is protected by a *guard*, also called *ChallengeAuthenticator*, which enforces the authentication requirements an access requester needs to fulfill in order to get access to this resource. Access control at the entities IP and A is technically implemented through such guards. For attribute-based authentication, a new type of guard has been implemented which invokes the proper manager components of our security framework. Thereby, the guard encapsulates the whole authentication and access control process on the services site.

Secure element. Our architecture supports the use of a secure element for storing the user's secret key and performing all related computations. As secure element, we used the secure Mobile Security Card SE 1.0 by G&D, a microSD card comprising a tamper resistant Java Card chip. We have implemented the portions of the idemix protocols comprising operations on the user's secret key as an applet to be executed on the secure element. The protocol implementation of idemix on M has been adapted to invoke the computations on the secure element for those portions. The implementation uses the *OV-Chip 2.0 Bignat library* for computing with arbitrary precision integers on a Java Card.

The secure element requires a PIN code to be unlocked whenever (parts of) a proof protocol is to be computed.

Entities. The following table shows the hardware used for the different entities:

Entity	**Realization**
M	Samsung Galaxy i9000: 1 GHz ARM Cortex-A8, 512MB RAM, 480x800 WVGA Super AMOLED screen, 2592 x 1944 Camera
	The smart phone is running Android 2.2
T, A, and IP	DELL E4300: Intel Core2 Duo P9600 @2.54GHz, 4GB RAM, Windows 7(64), 1280x720 Webcam
	The three entities are run as separate services on a single Windows PC[10]

On the mobile device M runs an Android application on top of our framework which is explained earlier. Communication is handled by the client-side implementation of the RESTlet framework.

The terminal T is realized as a GWT (v. 2.1.1) browser application, using the RESTlet-GWT module and the PC's display and webcam as hardware for realizing the visual channels.

The authorization server A and identity provider IP are realized as Tomcat (v. 6.0) applications, featuring the RESTlet communication framework, the Identity Mixer and other authentication protocols.

Registration of U *with* IP. There are several ways for the user to obtain an anonymous credential from the identity provider IP. For each credential type, IP has a guard that specifies the policy for obtaining such a credential (i.e., the requirements that have to be met). For instance, the policy may specify that the user first has to authenticate with his Belgian eID card. The attribute values that are obtained during this authentication can then be used by IP as attributes of the credential to be issued (i.e., IP re-certifies these values in a new anonymous credential).

The guard- and policy-based design of the identity provider is very flexible and can be easily adapted to whatever authentication is required.

[10] In real-world deployments, these services can easily be distributed over multiple machines.

Authentication of U *to* A. The authentication message flow follows the protocol specification of Section 3. The policy π_{RP}, specified in the CARL-language, is converted into an easy-to-understand format, which is then displayed on M's screen. U is required to choose how to fulfill the policy and to give her consent to the information release. Next, the user is challenged to enter her PIN code to activate the secure element.

5 Measurement Results

In this section we present and discuss the measurements we have obtained with respect to our prototype. We considered three different metrics that have a major effect on the overall system performance:

1. The runtime of a cryptographic idemix proof, that is, the runtimes of the prover and verifier sides of the proof;
2. The additional overhead of an idemix proof incurred by the use of the secure element;
3. The encoding size of the idemix proof, which is relevant in the context of the bandwidth-limited visual channels we employ.

5.1 Identity Mixer Proof Runtimes

We present measurements for a spectrum of different variants of Identity Mixer proofs. All experiments have been repeated with three different sizes of the SRSA modulus: 1024 bits, 1536 bits and 2048 bits. For the experiments, we used different types of credentials and different kinds of proofs.

(a) credentials with no embedded attributes (the user can then only prove to possess a valid credential)
(b) credentials with three embedded integer attributes:
  (b.1) all attributes remain hidden;
  (b.2) all attributes are disclosed;
  (b.3) the proof contains one inequality proof over an attribute; the values of the attributes remain hidden (e.g., $eID.dateOfBirth \leq dateMinusYears(today(), 16)$);
  (b.4) the proof contains a proof over an enumeration-type attribute; the values of the attributes remain hidden (e.g., $driverLicense.type \in [A, B, EB]$).

We use the following encoding triple as a shorthand notation for the structure (i.e., attributes and used features) of a proof: $[a_t, a_r, F]$ with a_t the total number of integer attributes embedded in the credential, a_r the number of revealed attributes, and F a feature to be proven or $\emptyset$ in case of no feature.

Table 4 summarizes the average of the values we have measured at the prover and verifier side (without secure element) of the protocol and the overall runtime for the proof variants (a)-(b.4) with the three different modulus sizes. The communication overhead has not been taken into account.

Note that revealing all attribute values (b.2) is almost as efficient as proving possession of a credential without attributes (a), which can be explained by how the cryptographic proof is computed (cf. [15], Sec. 6.2.3): disclosed attributes are proven with modular exponentiations with exponents that are small compared to the actual attribute sizes and, hence, have no major influence on the overall protocol runtime[11].

[11] Other experiments, not-shown here, confirmed that the overhead, as expected, is linear in the number of attributes that remain hidden.

Proof (b.3) illustrates the computational overhead of an inequality proof $(3,0,\text{ineq})$, such as proving that one's birth happened more than 16 years ago. Similarly, in (b.4) $(3,0,\text{enum})$, proving that an attribute has one of several possible values adds additional overhead (cf. the protocol specification [15]).

A modulus size of 2048 bit, which is recommended for high-security applications, shows a total runtime overhead of at least 0.9 seconds (which is far less than the time necessary for scanning the QR-codes). The experiments revealed that the computation time on the mobile phone is comparable to that on the server, although the prover side of the protocol needs to perform more computations [15]. This is unexpected as the CPU of the phone is substantially slower than the PC's CPU. We discovered, however, that the BigInteger class in the Android environment invokes native code, while on the PC, the class is entirely implemented in JAVA.

Table 4. Timing results (average over 100 runs), in milliseconds: prove, verify, and total.

(ms)	1024			1536			2048		
(a_t, a_r, F)	prove	verify	total	prove	verify	total	prove	verify	total
(a) $0,0,\emptyset$	103	78	181	240	187	427	495	375	870
(b) $3,0,\emptyset$	139	125	264	323	265	588	634	515	1149
(c) $3,3,\emptyset$	102	78	180	243	187	430	495	375	870
(d) $3,0,\text{ineq}$	481	436	917	1182	1077	2259	2358	2184	4542
(e) $3,0,\text{enum}$	247	213	460	617	510	1127	1259	1014	2273

5.2 Overhead Caused by the Secure Element

We have measured the overhead incurred by the use of the smart μSD card as secure element for storing the user's secrets, including the communication between the mobile device and the secure element.

The figures in Table 5 show a substantial additional overhead compared to the timing results in Table 4. The overhead for each modulus length is fixed and independent of the proof specification. Of course, it does not influence the verifier's side. As an example, in a basic proof (type (a)) with a 1024 bit modulus, an additional overhead of 1.26s is added to the 0.18s necessary for the proof generation without secure element, which results in a total of 1.44s. Note that a basic proof without attributes is comparable to the DAA scheme [5]. In related work, implementations of the DAA scheme were made to run entirely on a secure element [24, 3]. For the same key length, a proof takes 4.2s.

Table 5 also shows that a significant share of the additional overhead comes from the communication between the smart phone and the secure element. This can partially be explained by the current implementation requiring four rounds of communication. This can be reduced to two rounds, by piggy-backing the messages for PIN verification and protocol selection (issue or prove) to the messages required by the idemix library.

Note that for the 1024 and 1536 bit modulus the communication delay is the same, while for the 2048 bit modulus, the communication takes longer. This is due to the fact that communication happens in message blocks of 254 bytes. For 2048 bit modulus, two message blocks are necessary.

Table 5. Overhead, in milliseconds, incurred by the secure element

(ms)	1024	1536	2048
build proof	1262	1606	2082
communication	310	310	375
computation	952	1296	1707

Table 6. Size of the message m_2

(bytes)	1024	1536	2048
(a) $0,0,\emptyset$	793	878	1005
proof	589	675	802
header info	147	148	148
response info	57	56	56
(b) $3,0,\emptyset$	1053	1138	1267
proof	811	897	1024
header info	185	317	187
response info	56	57	57
(d) $3,0,$ *ineq*	3243	4031	4855
proof	2867	3657	1504
header info	215	317	216
response info	57	57	57

5.3 Size of QR Codes

Proofs generated by the idemix library are formatted in XML. As the visual channels used for exchanging the QR-codes are severely limited in bandwidth, and since the idemix proof is the largest part of the content to be transferred over this channel, a customized space-efficient binary format has been used for representing idemix proofs, instead of XML. Table 6, presents the size of the message m_2, of which SPK_U is the major part. In the table, the message size is decomposed into: the *proof* size, being the number of bytes of the idemix proof; the *header info* size, being the size of additional information required to encode the idemix proof in the custom format (such as attribute names and lengths of proof values); and the *response info* size, being the size of additional information required by the relying party (such as a reference to the chosen policy).

Table 6 also shows that different proof specifications result in quite different proof sizes. For the more complex proofs, e.g., proof (b.3), the size of the proof becomes too big to be encoded in a single display-readable QR-code[12]. Two solutions exist: either, scenario 2 is used in which the proof is sent via the radio channel to A; or, the message is split into multiple chunks which results in a series of QR-codes that are displayed one after another.

Note that the generation of the QR-codes on the mobile device currently takes a substantial fraction of the overall protocol runtime. For showing a credential without attributes, the QR-code is generated in about 0.8s. For the case of an interval proof with 2048 bit modulus, two (larger) QR-codes are generated in about 2.5s.

6 Conclusion & Future Work

Building on mobile devices, we provide a feasible solution to support attribute-based credentials as a privacy-preserving authentication solution. We presented protocols that employ short-range channels to establish an authenticated channel between a mobile and a relying party. Therefore, our system architecture realizes the attribute-based authentication protocols using visual short-range communication channels based on QR-codes. Nevertheless, other short-range channels can be supported as well.

[12] The QR standard specifies that only about three kilobytes of binary data may be included in one QR code.

For increased security and assurance, our system architecture and implementation comprises an optional secure element based on a secure μSD token. This achieves not only sharing and theft protection for the user's secret key material, but also a stronger binding between a user and her mobile device through authentication between those.
As a validation, a prototype has been built on an Android smart phone that implements two scenarios: authentication to a vending-machine, and to a remote website. Nevertheless, our system is applicable to a wide range of practically-relevant authentication scenarios. We presented measurements that demonstrate the feasibility of our solution and obtained encouraging results regarding the practicality of anonymous authentication technologies on standard smart phones.
Today's smart phones suffer from vulnerabilities that may make the software-based computations or the I/O between the user and her device untrusted, e.g., captured or influenced by a virus. Therefore, Trusted Execution Environments allow certain processes to be executed with a higher level of assurance, thereby, e.g., ensuring that no virus can change computations or intercept the I/O of such process to the user. Developments on this are ongoing and can be employed as orthogonal mechanism in our system architecture once they will be deployed on mainstream platforms.

Future extensions on the protocol level may comprise the introduction of the user accountability property [1, 10] through the use of verifiable encryption [6], or the support of credential revocation mechanisms, e.g., based on dynamic accumulators. Currently, access to the card is protected by a four digit PIN code, but may be replaced by gesture locks or biometric access control for increased usability and security. Though, the CARL language offers adequate expressiveness to address advanced authentication requirements, it lacks a number of useful properties. For instance, the language could be further extended to support the generation of (domain-specific) pseudonyms and once revocation is supported, there should be a proper way to provide revocation-specific information. All these features are not conceptually changing the constructions or architecture, which are the main focus of this paper, but rather require some additions, like for key management.

References

1. Michael Backes, Jan Camenisch, and Dieter Sommer. Anonymous yet accountable access control. In *Proceedings of ACM WPES 2005*, November 2005.
2. Endre Bangerter, Jan Camenisch, and Anna Lysyanskaya. A Cryptographic Framework for the Controlled Release of Certified Data. In *Security Protocols Workshop*, pages 20–42, 2004.
3. Patrik Bichsel, Jan Camenisch, Thomas Groß, and Victor Shoup. Anonymous credentials on a standard Java Card. In Ehab Al-Shaer, Somesh Jha, and Angelos D. Keromytis, editors, *Proc. 16th ACM CCS*, pages 600–610. ACM Press, November 2009.
4. Stefan Brands. *Rethinking Public Key Infrastructures and Digital Certificates: Building in Privacy*. MIT Press, Cambridge, MA, USA, 2000.
5. Ernie Brickell, Jan Camenisch, and Liqun Chen. Direct anonymous attestation. In *Proceedings of the 11th ACM conference on Computer and communications security*, CCS '04, pages 132–145, New York, NY, USA, 2004. ACM.
6. Jan Camenisch and Ivan Damgrd. Verifiable encryption, group encryption, and their applications to separable group signatures and signature sharing schemes. In Tatsuaki Okamoto, editor, *Advances in Cryptology ASIACRYPT 2000*, volume 1976 of *Lecture Notes in Computer Science*, pages 331–345. Springer Berlin / Heidelberg, 2000. 10.1007/3-540-44448-3-25.
7. Jan Camenisch and Els Van Herreweghen. Design and implementation of the *idemix* anonymous credential system. In *ACM Conference on Computer and Communications Security*, pages 21–30, 2002.

8. Jan Camenisch and Anna Lysyanskaya. An Efficient System for Non-transferable Anonymous Credentials with Optional Anonymity Revocation. In *EUROCRYPT*, pages 93–118, 2001.
9. Jan Camenisch, Sebastian Mödersheim, Gregory Neven, Franz-Stefan Preiss, and Dieter Sommer. A card requirements language enabling privacy-preserving access control. In *Proceedings of SACMAT 2010*, pages 119–128, 2010.
10. Jan Camenisch, Dieter Sommer, and Roger Zimmermann. A general certification framework with applications to privacy-enhancing certificate infrastructures. In *SEC*, pages 25–37, 2006.
11. Suresh Chari, Parviz Kermani, Sean Smith, and Ros Tassiulas. Security issues in m-commerce: A usage-based taxonomy. e-commerce agents. In *LNAI*, pages 264–282. Springer, 2001.
12. David Chaum. Security Without Identification: Transaction Systems to Make Big Brother Obsolete. *Commun. ACM*, 28(10):1030–1044, 1985.
13. Joris Claessens. *Analysis and design of an advanced infrastructure for secure and anonymous electronic payment systems on the Internet*. PhD thesis, Katholieke Universiteit Leuven, 2002. Bart Preneel and Joos Vandewalle (promotors).
14. Luuk Danes. *Smart Card Integration in the pseudonym system idemix*. Master's thesis, University of Groningen, 2007.
15. IBM. Specification of the Identity Mixer cryptographic library, v. 2.3.3. Ibm research report, IBM Research, 2011.
16. Kuan-Chieh Liao, Wei-Hsun Lee, Min-Hsuan Sung, and Ting-Ching Lin. A one-time password scheme with QR-code based on mobile phone. In *Fifth International Joint Conference on INC, IMS and IDC, 2009. NCM '09.*, pages 2069 –2071, August 2009.
17. J. Louvel and T. Boileau. *Restlet: Official Developer's Guide to Restful Web Applications in Java*. Apress, 1 edition, 2010.
18. Rene Mayrhofer and Hans Gellersen. Shake well before use: Authentication based on accelerometer data. In Anthony LaMarca, Marc Langheinrich, and Khai Truong, editors, *Pervasive Computing*, volume 4480 of *Lecture Notes in Computer Science*, pages 144–161. Springer Berlin / Heidelberg, 2007.
19. Jonathan McCune, Adrian Perrig, and Michael Reiter. Seeing-Is-Believing: using camera phones for human-verifiable authentication. *International Journal of Security and Networks*, 4(1):43–56, 2009.
20. Wojciech Mostowski and Pim Vullers. Efficient U-Prove implementation for anonymous credentials on smart cards. In George Kesidis and Haining Wang, editors, *7th International ICST Conference on Security and Privacy in Communication Networks, SecureComm 2011, London, UK, September 7-9, 2011. Proceedings*, volume 96 of *Lecture Notes of the Institute for ComputerSciences, Social-Informatics and Tele-communications Engineering (LNICST)*, pages 243–260. Springer-Verlag, 2011. (to appear).
21. L. Richardson and S. Ruby. *RESTful web services*. O'Reilly Series. O'Reilly, 2007.
22. Claudio Soriente, Gene Tsudik, and Ersin Uzun. HAPADEP: Human-assisted pure audio device pairing. In Tzong-Chen Wu, Chin-Laung Lei, Vincent Rijmen, and Der-Tsai Lee, editors, *Information Security*, volume 5222 of *Lecture Notes in Computer Science*, pages 385–400. Springer Berlin / Heidelberg, 2008.
23. Guenther Starnberger, Lorenz Froihofer, and Karl Goeschka. QR-TAN: Secure mobile transaction authentication. In *International Conference on Availability, Reliability and Security, 2009. ARES '09*, pages 578 –583, March 2009.
24. Michaël Sterckx, Benedikt Gierlichs, Bart Preneel, and Ingrid Verbauwhede. Efficient implementation of anonymous credentials on java card smart cards. In *First IEEE International Workshop on Information Forensics and Security, 2009. WIFS 2009.*, pages 106–110. IEEE Computer Society Press, December 2009.
25. Alex Varshavsky, Adin Scannell, Anthony LaMarca, and Eyal De Lara. Amigo: proximity-based authentication of mobile devices. In *Proceedings of the 9th international conference on Ubiquitous computing*, UbiComp '07, pages 253–270, Berlin, Heidelberg, 2007. Springer-Verlag.