User-Centric Identity Management using Trusted Modules

Jan Vossaert¹, Jorn Lapon¹, Bart De Decker², and Vincent Naessens¹

¹ Katholieke Hogeschool Sint-Lieven, Department of Industrial Engineering
Gebroeders Desmetstraat 1, 9000 Ghent, Belgium

`firstname.lastname@kahosl.be`

² Katholieke Universiteit Leuven, Department of Computer Science,
Celestijnenlaan 200A, 3001 Heverlee, Belgium

`firstname.lastname@cs.kuleuven.be`

Abstract. Many service providers want to control access to their services and offer personalized services. This implies that the service provider requests and stores personal attributes. However, many service providers are not sure about the correctness of attributes that are disclosed by the user during registration. Federated identity management systems aim at increasing the user-friendliness of authentication procedures, while at the same time ensuring strong authentication to service providers. This paper presents a new flexible approach for user-centric identity management, using trusted modules. Our approach combines several privacy features available in current federated identity management systems and offers extra functionality. For instance, attribute aggregation is supported and the problem of user impersonization by identity providers is tackled.

Key words: user-centric identity management, privacy, security

1 Introduction

Many service providers want to control access to their services and critical resources. A trivial solution compels users to go through a registration phase. After a successful registration, the user receives a credential that is required to consume certain services or to access resources. The credential can either be a login and password, or a strong authentication token (such as a private key and corresponding certificate). However, this approach has many drawbacks. First, the number of tokens increases linear to the number of service providers. Either the user has to remember many passwords or store a lot of certificates (and their corresponding key). Passwords and certificates can be stored on the user's workstation. However, this decreases the accessibility of services (i.e. services cannot be accessed from a remote location). Second, an increasing number of service providers want to offer personalized services. This implies that the service provider requests and stores personal attributes. However, many service providers are not sure about the correctness of attributes that are disclosed by the user during registration.

Some approaches and initiatives aim at solving these drawbacks. First, *federated identity management systems* [1, 2] aim at increasing the user-friendliness of authentication procedures, while at the same time ensuring strong authentication to service providers. In fact, identity providers keep a set of personal attributes of registered individuals. They release a subset of those attributes when the user authenticates to a service provider. However, in many architectures, the user only has limited (or even no) control about the attributes that are exchanged between the identity provider and the service provider. Moreover, the identity provider is informed about which services are consumed by that user. Second, many countries are rolling out *electronic identity technology*. A majority use a smart (or SIM) card that contains a few certificates (and private keys) that can be used to authenticate the user to multiple services. Service providers develop authentication and authorization modules that are eID compliant. Although each user now has the credentials required to authenticate to multiple services, current solutions have many drawbacks. First, eID cards typically store only static attributes (i.e. personal properties that do not change during a user's lifetime such as name, date of birth ...). Moreover, users often have little impact on the attributes that are released during authentication. In some architectures, they are always identifiable and need to release attributes that are not required for the particular service (e.g. all attributes including a unique identifier embedded in an authentication certificate). An overview of the privacy features of European eID initiatives is given in [3].

This paper proposes *a new approach for user-centric identity management* that tackles several privacy and security problems of current federated identity management systems (FIMs) and also adds extra functionality. First, our approach combines several privacy features available in current FIMs. For instance, identity providers can no longer profile users and the disclosure of personal attributes is controlled by multiple parties (including the user). Second, identity providers can no longer impersonate users. Third, attribute aggregation [4], in which attributes can be retrieved from multiple identity providers and released to a service provider, is supported. Further, the system can also be used in offline settings under limited conditions.

The rest of this paper is structured as follows. The general approach is clearly defined in section 2. Section 3 points to related work. The roles and requirements are described in section 4. Section 5 discusses the protocols in more detail. Next, the approach is evaluated in section 6. This paper ends with general conclusions.

2 General Approach

This paper proposes a privacy friendly user-centric federated identity management approach based on a trusted secure element. It is a federated identity management system (FIM), as multiple identity providers can endorse the user's personal information to multiple service providers. The secure element is the mediator between identity providers and service providers. More precisely, an identity provider can store some of the user's personal attributes (or properties thereof)

in her secure element. Information that is endorsed by identity providers can then be disclosed to service providers. The latter use the information to provide fine-grained access control and offer personalized services. For instance, when acquiring a ticket for a soccer contest, a student can request certain personal attributes from identity providers and release them to the ticketing service get reductions. A university can vouch that the user is a registered student, while the government can guarantee that the user is not a hooligan.

The secure element controls access to identity information. Before the user is authenticated, the service provider first has to authenticate to the secure element and prove that it is authorized to access certain personal attributes. The secure element verifies the acceptability of the service provider's information request. This verification ensures that only information from identity providers is queried, for which the identity providers (or their representative) gave their consent. The authorization info is included in the certificate (or credential) of the service provider. The certificate can be issued by the identity provider or a trusted third party. Additionally, the user may further restrict access to personal information through a policy or an explicit consent. If the query is acceptable, the secure element forwards this request to the identity provider(s) that can provide the information. In this scheme, the service provider remains hidden to identity providers. However, the latter are ensured, by the trusted element, that only trustworthy service providers can request their information.

Our approach combines several privacy features of current FIMs. First, *an identity provider cannot profile the user's actions*, as there is no direct link between identity providers and service providers (i.e. the secure element mediates personal information requests). Moreover, collusion of identity and service providers is prevented by means of service specific pseudonyms. Second, *the disclosure of personal information is controlled by multiple parties*. Service providers cannot retrieve personal data from identity providers without their prior consent. The identity provider only grants (limited) access to trustworthy service providers by means of authorization via certificates. For instance, an identity provider in the eHealth domain can restrict information retrieval to service providers in the same domain. Each eHealth provider can only access a subset of attributes endorsed by the identity provider (namely the attributes that are strictly required for that particular service). However, no commercial service provider can access information stored by that identity provider. Moreover, it is conceivable that the access rights are not issued by the identity provider itself but by a trusted *audit authority* that grants access to service providers on behalf of a federation of identity providers. Users can further restrict the disclosure of personal information or explicit user consent is required prior to data release. Each user can configure its own privacy policy.

Attributes can be cached temporarily in the secure element. This makes the scheme more efficient and usable in an offline environment. A flexible and scalable revocation procedure is foreseen; even offline services can check whether a trusted module is revoked. The mechanism is based on a validation service that regularly updates status information in the secure element. Our approach

is also flexible and scalable in the sense that new service providers and identity providers can easily be added. Moreover, a service provider can also play the role of an identity provider for other service providers. A trusted audit authority can also grant access to the service provider on behalf of an entire group. When a service provider wants to retrieve personal information, it can be supplied by any identity provider in that group. For instance, an individual can prove to be a student, without revealing the university, as it can be any of the universities in the group.

The beneficial privacy properties that this scheme provides, originate in the use of a secure trusted element. Such a secure element can be for instance a smart card, a SIM card or even a trusted module in an embedded device, and is issued by a trusted third party. It acts as a gateway between identity providers and service providers. Figure 1 provides an overview of the proposed architecture.

Fig. 1. Overview of the architecture.

3 Related work

A taxonomy of different identity management strategies is given in [5] together with a detailed discussion of their respective usability and privacy features. The systems that are relevant with respect to our contribution are summarized below.

Jøsang and Pope [6] present a user-centric identity management model using a *personal authentication device (PAD)*. Each service provider can store an authentication token on the PAD of the user. Our work generalizes the PAD concept to a personal identification device with extended functionality (e.g. support for multiple identity providers, deanonymization,...) and a concrete implementation is presented.

OpenID [7] is a prototypical example of a decentralized SSO identity model. In OpenID, each user has a global identifier (i.e. an URL) that is released to the service provider. The latter can use the global identifier to locate the corresponding OpenID provider and handle the authentication procedure. However, the system is vulnerable to phishing attacks and TLS is not required during the authentication process. Moreover, the identity provider learns which services are consumed by a user and attributes are not endorsed by the identity provider.

In the federated identity management model [1, 2], of which Shibboleth [8] and CardSpace [9] are common examples, a user is known to at least one organization (i.e. the identity provider) in the federation (i.e. a group of organizations with mutual trust agreements). If a user contacts a service provider, authentication is delegated to the identity provider of the user. The identity provider releases the personal data that are requested by the service provider. Shibboleth and CardSpace realize a high level of security, scalability and interoperability (e.g. based on standards, easy integration in Web services, support for a multitude of authentication systems) but also have major drawbacks. For instance, a high level of trust is required in the identity providers. They are trusted to release only the required attributes and they can impersonate the user since typically no authentication between user and service provider is required. They may also know which services a user is accessing. Moreover, combining attributes from different identity providers is not possible in Shibboleth. In CardSpace, this requires the user complete several authentication procedures. Further, the possibilities for offline use are limited and many identity providers still support a weak login procedure (based on passwords) because often no infrastructure for strong authentication is available. Therefore, a significant amount of trust is still required in the workstation (e.g. key loggers can compromise user secrets) impeding the security when used on public computers (e.g. in libraries, ...).

Several European countries are issuing governmental eID cards to tackle these shortcomings. A trusted module approach is used by the German eID. A common key pair (SK_{Co} , PK_{Co}) is shared by a large set of smart cards and is used to set up a secure channel with service providers. The service provider can only query a limited subset of the personal information that is stored on the card. This subset is approved by the government and stored in the service provider's certificate $Cert_{SP}$. Moreover, service provider specific nymns are generated by the card, based on a personal master secret in the card and data in $Cert_{SP}$. The Austrian eID [10] is a technology neutral concept that can be deployed on different valid physical tokens (i.e. devices that comply with Austrian legislation for serving as eID card). Moreover, multiple organizations can issue identity cards (e.g. banks issue ATM cards and one of the telecom operators issues SIM cards that are considered valid eID cards).

Governmental identity cards mitigate multiple drawbacks of the presented federated identity management systems. However, many designs are not flexible as service providers can only request attributes that are stored in the card itself. Attributes are typically stored in the card during the whole card's lifetime. This implies that only immutable attributes can be stored in the card. Our approach

aims at eliminating the drawbacks of existing federated identity management systems and current eID initiatives.

Suriadi et al. [11] propose a user-centric federated single sign-on system based on the private credential system proposed in Bangerter et al [12]. However, the computationally expensive nature of the credential scheme limits the feasibility in mobile solutions. Similar concerns apply to the identity management system [13] proposed in the PRIME project [14]. Moreover, this system is less flexible since attributes are embedded in immutable credentials. Multiple credentials need to be used if not all the requested attributes are contained in a single credential.

4 Assumptions, roles and requirements

Each user owns a trusted module that contains logic for processing requests from the different parties in the scheme. The card issuer is trusted to ensure the proper working of the trusted module. Essentially, adversaries should not be able to influence the inner working or deviate from the imposed protocols. Nor can they directly access the memory of the trusted module. Hence, a service provider can trust the data that was returned by the trusted module. A related discussion about trust in trusted computing platforms is given in [15, 16]. Note that the German eID requires similar trust assumptions.

Common trusted modules such as SIM and smart cards do not have a user interface. A host is required for user interactions such as entering a PIN. This implies that the user has to rely on the host or a sophisticated card reader (e.g. with PIN-pad and screen).

4.1 Roles

Each *user* (U) has a *trusted module* (TM), in this text also referenced to as *card* since a prototype implementation was made on a smart card. *Middleware* (M) is installed at the client side (i.e. host or card reader). It allows users and remote parties to interact with the card. The *service provider* (SP) offers personalized services to authorized users. The *identity provider* (IP) returns endorsed personal attributes to trusted modules which can be released to service providers. The *card issuer* (CI) issues trusted modules to users. The *(re)validation authority* (RA) can (re)validate and block trusted modules. The *audit authority* (AU) grants rights to service providers to retrieve personal data from specific identity providers. The *certification authority* (CA) issues certificates to the different actors.

4.2 Functional requirements

- F_1 Service providers can retrieve personal attributes either stored in the card and/or managed by an identity provider.
- F_2 Cards can be personalized (e.g. through privacy policies and preferences).
- F_3 Adding new services and identity providers is straightforward.
- F_4 The card can be used *online* and *offline*.

4.3 Security and privacy requirements

- S_1 Mutual and strong authentication between the card and other actors.
- S_2 Controlled access to personal attributes that are accessible through the card (i.e. based on rights/privileges).
- S_3 Secure communication. An adversary should not be able to gain valuable information by monitoring the communication.
- P_1 A trusted party, namely the audit authority, restricts the information that can be retrieved by service providers.
- P_2 Users can further restrict the personal data that may be released.
- P_3 The solution supports one-time anonymity, pseudonymity as well as identifiability of users towards service providers.
- P_4 The card issuer only provides the trusted environment. This should not give him advantage in gaining information about the owner.
- P_5 Support for conditional anonymity. This allows for identifying suspects in case of abuse.
- P_6 The disclosed information should be trustworthy without the need for proof (e.g. signatures), reducing its economic value towards third parties.

4.4 Trust assumptions

This scheme changes some of the trust assumptions that are inherent to traditional schemes without a trusted module. Besides the trust in the identity provider, to provide correct information about the user, some important assumptions are listed below.

The *card issuer* is trusted by all stakeholders to manufacture and deploy trusted modules that complies with the specifications. Additionally, the *re(validation) authority* is trusted to block stolen and lost cards, and to revalidate valid cards. The *audit authority* is trusted by IP and partially by *U* to give fair access rights to service providers. The *host* and *middleware* are trusted to securely accommodate a user interface for the trusted module, and an anonymous channel [17] to the remote parties.

4.5 Notation

The following notation will be used when describing the protocols. Note that the integrity of encrypted messages is assumed.

- $\rightarrow$ or $\leftarrow$ denote the direction of the data flow.
- $\xrightarrow{\sim}$ or $\xleftarrow{\sim}$ assume communication over an anonymous channel.
- $\boxed{U}$ indicates physical interactions with the card holder.
- Variables* are shown in italic.
- card attributes** are shown in teletype font.
- "String" represents a string constant.
- $\{a, b, \dots\}_K$ denotes the symmetric encryption of variables $(a, b, \dots)$, under the session key K . This key has been established between the two communicating parties.

5 Protocols

The keys and certificates that are maintained in the scheme are listed below:

- Each service provider (SP) generates a key pair, the public key of which is certified by the audit authority (AU) in a certificate $Cert_{SP}$. The certificate also lists the access rights of SP to the user’s personal information (which is approved by AU , IP or a set of IP s). The access rights restrict the set of queries (i.e. the set of attributes or properties thereof) that can be requested from the card.
- Each identity provider (IP) generates a key pair, the public key of which is certified by certificate authority CA . If required, the certificate may also include access rights. These access rights allow the identity provider to obtain personal information from the user during enrollment.
- The public keys of root CAs in the system are placed on the card during initialization. This allows the card to verify the certificates.

5.1 Card Issuance

A card issuer CI issues cards to users. The cards contain a keypair (SK_{Co} , PK_{Co}) – certified by CA – that is identical for a large set of cards. Since the private key is kept inside the trusted environment, making a valid signature with it proves that the card is genuine. Hereby, no uniquely identifying information about the card nor the card owner is released.

Before an individual can use the card, it needs to be activated. At this phase, a card specific pseudonym $\mathcal{S}_C$ is generated on the card, which is later used to generate service(-provider) specific pseudonyms. Additionally, the user has to confirm a personal PIN.

At the end, the CI can already store a set of immutable attributes (of the card or the card holder) in the card. Alternatively, if the card is used as a *governmental identity card*, CI may cooperate with a governmental identity provider IP_{Gov} to bind the card to a specific user. The card discloses the provider specific nym_{Gov} and **chip_number** to IP_{Gov} . The former allows IP_{Gov} to bind the citizen to the card. The card can now request attributes from IP_{Gov} (cfr. section 5.5). The latter enables revocation of stolen and lost cards by IP_{Gov} .

5.2 (re)Validation of the card

The card (re)validation protocol confirms that a card is still valid at a given time. The validity of the card is verified by the (re)validation authority RA , which keeps track of the revoked cards, based on the **chip_number**. Note that the chip numbers of stolen and lost cards are sent to RA . If a card is revoked, RA will block the card such that further authentications are no longer possible. Otherwise, the **lastValTime** is updated with the current time. Whenever the card contacts a relying party, it ensures that the **lastValTime** lies within a certain time frame, that is acceptable for that party (see section 5.4). Table 1

shows in detail how this `lastValTime` is updated when the card is inserted in the card reader.

When the card has not been revalidated recently, user confirmation is requested³ to start the revalidation protocol (1–6). Next, an authenticated key agreement protocol is executed. Therefore, *TM* uses $(SK_{Co}, Cert_{Co})$ and *RA* uses $(SK_{RA}, Cert_{RA})$. This step results in a shared session key K_s (7). *TM* then sends the `chip_number` (encrypted with K_s) to *RA* (8). Next, the issuer checks the revocation status of the card, encrypts the current time with K_s and sends it to the card (9–11). Finally, upon receiving the encrypted time, the card decrypts it, and updates its `lastValTime` (12). The actual revalidation time is sent to the card since common trusted modules such as SIM and smart cards do not have a real-time clock. The `lastValTime` can be used to verify the validity period of certificates.

revalidateTrustedModule():

- (1) *TM* : inserted in reader
- (2) *TM* $\leftarrow$ *M* : "Hello", *currentTime*
- (3) *TM* $\rightarrow$ *M* : $reqRevalidation := (lastValTime < currentTime - \delta)$
- (4) *M* $\rightarrow$ $\boxed{U}$: **if** (*reqRevalidation*) **showRevalWindow()** **else** **abort()**
- (5) *M* $\leftarrow$ $\boxed{U}$: **response** [*assume Yes*; otherwise **abort()**]
- (6) *M* $\xrightarrow{\circlearrowright}$ *RA* : "RevalidationRequest"
- (7) *TM* $\xleftrightarrow{\circlearrowright}$ *M* $\xleftrightarrow{\circlearrowright}$ *RA* : $K_s := \text{authKeyAgreement}(SK_{Co}, Cert_{Co}; SK_{RA}, Cert_{RA})$
- (8) *TM* $\rightarrow$ *M* $\xrightarrow{\circlearrowright}$ *RA* : $\{chip_number\}_{K_s}$
- (9) *RA* : **if** (**not** *isValid(chip_number)*) **sendBlockCommand()**, **abort()**
- (10) *RA* : $time := \text{getCurrentTime}()$
- (11) *TM* $\leftarrow$ *M* $\xleftrightarrow{\circlearrowright}$ *RA* : $\{time\}_{K_s}$
- (12) *TM* : $lastValTime := time$

Table 1. The card is regularly revalidated by the government.

5.3 Authentication

During authentication, the card only proves to be genuine. Authentication between the card and relying party, which can be either a service provider or an identity provider, is accomplished as follows: if the card is not blocked, the card and relying party *P* initiate a standard authenticated key agreement protocol (1–2). The relying party, therefore, uses its certificate $Cert_P$, while the card uses the common certificate $Cert_{Co}$, preserving the privacy of the user. The resulting session key is used to encrypt all messages sent between *TM* and *P*.

After key agreement, a new session is started (3–4), and the session identifier *sesId*, is communicated to *P* (5). Then, *P* will send the earliest acceptable validation time, `accValTime`(6). The card verifies whether it has been revalidated more recently than `accValTime`(7). This ensures that the card was at least valid until `accValTime`, without revealing the precise value of `lastValTime`. If the

³ Note that steps 4–5 are optional. If they are omitted, the card will automatically be revalidated or blocked.

card has been revalidated recently enough, the card stores the name of the relying party, and if available, its access rights, in the temporary session object (8-9). The former can later be used to create a service specific nym.

$sesId := \text{Authenticate}():$	
(1) TM	: if (not $\text{isActivated}() \vee \text{isBlocked}()$) abort ()
(2) $TM \rightleftharpoons M \xrightarrow{\emptyset} P$	: $K_s := \text{authKeyAgreement}(SK_{Co}, Cert_{Co}; SK_P, Cert_P)$
(3) TM	: $sesId := \text{startNewSession}()$
(4) TM	: $\text{session}[sesId].K_s := K_s$
(5) $TM \rightarrow M \xrightarrow{\emptyset} P$	: $\{sesId\}_{K_s}$
(6) $TM \leftarrow M \xleftarrow{\emptyset} P$	: $sesId, \{ \text{"validatedAfter"}, \text{accValTime} \}_{K_s}$
(7) TM	: if ($\text{lastValTime} < \text{accValTime}$) abort ()
(8) TM	: $\text{session}[sesId].rights := Cert_P.rights$
(9) TM	: $\text{session}[sesId].subject := Cert_P.subject$

Table 2. Authentication between relying party P and card TM .

5.4 Access to (personalized) services

Before an individual can use an online service, the service provider may require the user to release certain personal attributes either stored in the card or available at identity providers. If no persistent pseudonym is used throughout several service consumptions, different attributes may be requested each time the individual wants to use that particular service. Moreover, this also allows the service provider to minimize data retention which reduces potential damage done by hackers. This is realized as follows and illustrated in Table 3.

The card and the service provider mutually authenticate (see 5.3) (1). Consequently, the service provider sends its **"attributeQuery"**-command to the card (2). This query can contain an explicit request to disclose the service specific pseudonym nym_{SP} . Note that not every service provider should be able to link different service requests by the same user. For instance, for some sites it is sufficient to prove that you are older than 18, in order to access the service. Next, the card verifies that the service provider is allowed to query this information, based on the access rights obtained during authentication (3). User-policies may further impose restrictions on the disclosure of these attributes. For instance, the user policy may specify that a valid PIN code is required before the attributes (or properties thereof) are released (4). Optionally, based on the user's policies the card may send the query together with ses_{SP} and SP 's name to the host M to be displayed to the user. After the user's consent, the (possibly modified) query (actually, the delta) is returned to the card (5-8). The card removes the delta from the query (9) and if a PIN was required, it is verified (10). The card then checks which attributes are locally available on the card (11) and which attributes have to be fetched from identity providers using **attributeMap**. This is a table that indicates which attributes can be retrieved from which identity providers. This strategy was proposed as a *Linking Service* in [4]. The service(-provider) specific nym_{SP} is generated locally and certain cached attributes that are still valid are retrieved

(12). Generating service(-provider) specific pseudonyms can be kept very simple: $nym_{SP} := \text{hash}(\mathcal{S}_C || \text{Cert}_{SP}.\text{subject})$ where **subject** is the name of the service provider as it is kept in its certificate; $\mathcal{S}_C$ is a card specific secret. Note that in case the service specific nym_{SP} is revealed during registration, disclosing this nym may be sufficient to use the service. The card then authenticates to all identity providers from which data has to be fetched (using the protocol defined in section 5.3), and sends the attribute request together with the identity provider specific pseudonym nym_{IP} . Based on nym_{IP} , the identity provider can fetch the data from its database and return it to the $TM(13-21)$. Finally, the card encrypts all requested attributes with K_s and sends them to the $SP(22)$.

Identify():

```

(1)  $TM \xleftrightarrow{\varnothing} M \xleftrightarrow{\varnothing} SP : ses_{SP} := \text{Authenticate}()$ 
(2)  $TM \leftarrow M \xleftrightarrow{\varnothing} SP : ses_{SP}, \{ \text{"attributeQuery"}, query \}_{\text{session}[ses_{SP}].K_s}$ 
(3)  $TM : \text{if (not verifyQuery(query, session[ses_{SP}].rights)) abort()}$ 
(4)  $TM : query_P^* := \text{applyPolicy(query)}$ 
(5)  $TM \rightarrow M : query_P^*, ses_{SP}, SP = \text{session}[ses_{SP}].\text{subject}$ 
(6)  $M \rightarrow \boxed{U} : \text{showQueryWin}(SP, query_P^*)$ 
(7)  $M \leftarrow \boxed{U} : \text{response [Assume OK [delta_U^*, PIN]; otherwise abort()]}$ 
(8)  $TM \leftarrow M : \text{"deltaQuery", ses_{SP}, delta_U^* [, PIN]}$ 
(9)  $TM : query_{P,U}^* := query_P^* - delta_U^*$ 
(10)  $TM : \text{if (PINincorrect(PIN)) handleWrongPIN()}$ 
(11)  $TM : \text{attsPerIP} := \text{resolveQuery}(query_{P,U}^*, \text{attributeMap})$ 
(12)  $TM : \text{atts} := \text{getLocalData}(\text{attsPerIP} [\text{"local"}])$ 
(13)  $TM : \text{forall (IP in attsPerIP):}$ 
(14)  $TM \xleftrightarrow{\varnothing} M \xleftrightarrow{\varnothing} IP : ses_{IP} := \text{Authenticate}()$ 
(15)  $TM : nym_{IP} := \text{getNym}(\mathcal{S}_C, \text{session}[ses_{IP}].\text{subject})$ 
(16)  $TM : qry := \text{makeQuery}(\text{attsPerIP} [IP])$ 
(17)  $TM \rightarrow M \xleftrightarrow{\varnothing} IP : \{ nym_{IP}, qry \}_{\text{session}[ses_{IP}].K_s}$ 
(18)  $IP : \text{atts}_{IP} := \text{getData}(qry)$ 
(19)  $TM \leftarrow M \xleftrightarrow{\varnothing} IP : \{ \text{atts}_{IP} \}_{\text{session}[ses_{IP}].K_s}$ 
(20)  $TM : \text{atts.add}(\text{atts}_{IP})$ 
(21)  $TM : \text{endfor}$ 
(22)  $TM \rightarrow M \xleftrightarrow{\varnothing} SP : \{ \text{atts} \}_{\text{session}[ses_{SP}].K_s}$ 

```

Table 3. The card releases attributes to the authenticated service provider.

Offline Use With some minor changes, the card can be used for proving attributes offline. If the service provider only requests immutable attributes stored in the card, the protocol remains unmodified. However, the service provider can request attributes that are not stored by default on the card. A simple solution consists of caching these attributes on the card before they need to be used. The identity provider can define which attributes may be cached together with a retention time. Additionally, the service may impose restrictions on the freshness of the (cached) data; such restrictions can easily be passed to the card by including them in SP 's certificate.

5.5 Enrollment

Similar to accessing personalized services, a user may enroll with an identity provider. If so, the card discloses the identity provider specific nym_{IP} and possibly attributes from other identity providers. The identity provider links nym_{IP} to the user's personal attributes. The same pseudonym is used during future requests for personal information by TM . The identity providers can update `attributeMap` when a connection with the card is established.

5.6 Deanonymization

It must be possible to revoke the user's anonymity for certain services. One strategy consists of encrypting the `chip.number` together with the service specific nym or a random number (in case no nym is revealed) using the public key of a trusted party entitled to deanonymization. An "`attributeQuery`"-command can include a request to disclose the encrypted `chip.number` (used for deanonymization purposes). In case of abuse (and when the deanonymization option was used), the service provider forwards the encrypted data to the trusted party, which can decrypt the data and use the `chip.number` to obtain the users real identity from the card issuer.

6 Evaluation

This section first matches the solution with the requirements defined before, followed by a discussion on the overall functionality and possible extensions. For an in-depth evaluation of a prototype implementation on a smart card, we refer to [18]. [18] demonstrates the practical feasibility of the proposed architecture, presents concrete protocols that realize the security and privacy requirements and provides insight in the most important design decisions made during the instantiation of the architecture.

6.1 Requirements Review

Functional Functional requirements F_1 and F_4 are realized by a caching table on the card. The table keeps a set of personal attributes and their retention time. The identity provider defines the validity interval. The retention time of the owner's `name` may be unlimited whereas the retention time of `student related information` may be limited (i.e. one year). Caching attributes allows offline use of the card. Note that mutual authentication and generating service(-provider) specific pseudonyms does not require communication with external entities. The user has substantially more control than in traditional identity management systems (cfr. F_2). This is accomplished through personalization and user-consent. Users can define the set of attributes that can be disclosed without explicit consent and the set of attributes for which a PIN code is required. Moreover, service and identity providers can be blacklisted or whitelisted, or a combination thereof (cfr. F_3). If whitelisting is used, new service and identity providers have to be added explicitly on a whitelist in the card.

Security and Privacy The security requirements S_1 and S_3 are trivially fulfilled using standard authentication and encryption mechanisms. However, many PKI systems use X.509 certificates. Verifying the validity of these certificates requires verification of certification chains, and revocation lists or OCSP requests. This verification may hamper the performance of the card. An alternative approach uses simpler card verifiable certificates (CVC) with a short life-time. As such, revocation lists or OCSP requests are no longer required. Moreover, to decrease the length of the certificate chain, the certificate authority could issue short-lived CVCs directly to service providers after authentication with a standard X.509 certificate and a proof of acquired access rights. Personalization, auditing and certification enable effective access control (cfr. S_2).

Multiple measures have been taken to realize the privacy requirements. In contrast to many commercial identity management systems, this solution does not require direct communication between identity providers and service providers. On the contrary, identity providers are unaware about the services that are consumed by the card holders.

Access rights for service providers enable an external control mechanism as specified in P_1 . These access rights can be coarse-grained, such as *access to all eHealth data of an individual*, or fine-grained, such as *only allow the verification of certain age properties (e.g. age > 18)*. Moreover, if several trusted identity providers can supply the requested information, the service provider does not know which identity provider was used. For instance, when a service provider queries whether the card owner is a student, proving that property does not reveal the university providing this information. Nevertheless, service providers can restrict the set of trusted identity providers. They can request a permission to retrieve attributes from a single identity provider or a group (or federation) of identity providers to an audit authority. Also, a level of assurance (LOA) can be assigned to each identity provider (coarse-grained) or to each attribute that is kept by the identity provider (fine-grained) as proposed in [1]. A service provider can impose a minimal LOA for the requested attributes. The LOA is then included in $Cert_{SP}$. Additionally, card personalization and user-consent gives the user control on the disclosure of personal attributes as specified in P_2 . The card supports one-time anonymity, pseudonymity and identifiability (cfr. P_3). Users are identifiable if uniquely identifying attributes are disclosed; pseudonymity is realized by service specific pseudonyms. Users remain anonymous if they do not reveal the service-specific pseudonym nor any uniquely identifying information. During card issuance and activation, the card issuer obtains only a minimal set of information (cfr. P_4). It cannot calculate the card's pseudonyms, which prevents impersonation of a specific person by the card issuer. Simple extensions to the scheme (such as deanonymization) allow to enforce accountability measures as required by P_5 . Finally, as specified in P_6 , it is not possible to prove invalid statements with a genuine card. The service provider can be sure about the disclosed information, without the need for verifiable proofs (e.g. signatures). Hence, the information has less economic value for third parties.

6.2 Discussion

The proposed scheme has several advantages but also constraints. In the following, we discuss the most important ones and show how to tackle them.

Trust The major constraint of the scheme, is the trust that all parties require in the Card Issuer. As he has the common keypair, it could impersonate users. However, the card-specific pseudonyms are not known by the card issuer. As such, he cannot perform directed attacks to a specific user, alleviating the required trust from the user-perspective. Furthermore, more advanced cryptosystems, such as anonymous credentials [19], could be used to prove that a genuine card is used. Anonymous credentials allow to prove the value of attributes included in a credential, or properties thereof. However, these proofs decrease the overall performance significantly. However, the credential could be used only for proving the validity of the card, since all processing happens in the trusted module. Hence, each card can keep a unique credential generated on the card itself. To mitigate the trust required by the user in his workstation, a card reader with pinpad can be used. Alternatively, the trusted module can be a SIM card. The middleware can be (partially) implemented on the mobile which serves as a proxy between the trusted module and the outside world. The mobile can inform the user about authenticating service providers and allow for card personalization.

Scalability One of the prerequisites of the scheme is that service providers have an agreement with identity providers, eventually intermediated by an audit authority. This authority is trusted by the identity providers to grant reasonable access rights to legitimate service providers. However, the scheme can be very flexible. Multiple audit authorities can authorize service providers to request specific attributes of a subset of identity providers. The authority can bind the privileges to the certificate (e.g. by signing the privileges and a hash of the server certificate), instead of including them in the certificate. Alternatively, an identity provider can also grant privileges to service providers directly. Hence, the scheme is very flexible: new parties can be added dynamically.

In the proposed system, card (re)validation and blocking are handled by a dedicated party. This is a significant advantage for service (and identity) providers since they no longer need to maintain certificate revocation lists or support OCSP responders from different CA's, especially since mobile tokens are easily lost or stolen. The latter results in large revocation lists. Moreover, security threats resulting from outdated certificate revocation lists are avoided. However, sensitive services should require a recent `accValTime` resulting in a short vulnerability window when credentials are compromised. Less sensitive or offline services could accept a more permissive window of vulnerability.

Flexibility Service(-provider) specific pseudonyms are generated based on a secret. If the secret can be transferred securely to other modules (of the same

individual), he can access the same service using multiple alternatives. For instance, the secret can be kept both on a SIM and on a smart card. Moreover, the personalized policy can be tuned for a specific module. For instance, the smart card can be used for financial transactions whereas the SIM cannot. Alternatively, the secret can be module-specific. The secret can be generated on the card and never leaves the module. This approach is more secure as users cannot delegate the secret to other devices (possibly owned by other users). However, users must be able to backup their pseudonyms. If the module is replaced (e.g. when the module was stolen or lost), a dedicated protocol must be foreseen to replace old pseudonyms by the new ones.

Personalization The proposed solution gives the user control over the disclosure of data. Audit authorities and user-policies assist the user in what data may (not) be disclosed in a specific context. Audit authorities define system-wide access rules, while user-policies may further restrict those rules. Certain service providers may require the user's consent/PIN for retrieving confidential data, while other service providers cannot access this information.

7 Conclusion

This paper presented an approach for a user-centric identity management system using trusted modules. We demonstrated the high flexibility and the superior privacy and security properties of the proposed architecture with respect to existing systems. The release of personal information is controlled at two levels. An audit authority defines system-wide access rules. Users can further restrict those access rules or explicitly ask for user consent. Future research will focus on how the system can be deployed in a real-world setting and evaluate the impact on performance, security and privacy.

References

1. Chadwick, D.W.: Federated identity management. In Aldini, A., Barthe, G., Gorrieri, R., eds.: FOSAD. Volume 5705 of Lecture Notes in Computer Science., Springer (2008) 96–120
2. Ahn, G.J., Ko, M.: User-centric privacy management for federated identity management. In: COLCOM '07: Proceedings of the 2007 International Conference on Collaborative Computing: Networking, Applications and Worksharing, Washington, DC, USA, IEEE Computer Society (2007) 187–195
3. Ingo Naumann, G.H.: Privacy Features of European eID Card Specifications. Technical report, ENISA (2009)
4. Chadwick, D.W., Inman, G., Klingenstein, N.: A conceptual model for attribute aggregation. *Future Generation Computer Systems* **26**(7) (2010) 1043 – 1052
5. Jøsang, A., Zomai, M.A., Suriadi, S.: Usability and privacy in identity management architectures. In Brankovic, L., Coddington, P.D., Roddick, J.F., Steketee, C., Warren, J.R., Wendelborn, A.L., eds.: ACSW Frontiers. Volume 68 of CRPIT., Australian Computer Society (2007) 143–152

6. Jøsang, A., Pope, S.: User centric identity management. In: Asia Pacific Information Technology Security Conference, AusCERT2005, Australia. (2005) 77–89
7. Recordon, D., Reed, D.: OpenID 2.0: a platform for user-centric identity management. In: DIM '06: Proceedings of the second ACM workshop on Digital identity management, New York, NY, USA, ACM (2006) 11–16
8. Morgan, R.L., Cantor, S., Carmody, S., Hoehn, W., Klingenstein, K.: Federated security : The shibboleth approach. *EDUCAUSE Quarterly* **27**(4) (2004)
9. Bertocci, V., Serack, G., Baker, C.: Understanding windows cardspace: an introduction to the concepts and challenges of digital identities. Addison-Wesley Professional (2007)
10. Leitold, H., Hollosi, A., Posch, R.: Security architecture of the austrian citizen card concept. In: ACSAC '02: Proceedings of the 18th Annual Computer Security Applications Conference, Washington, DC, USA, IEEE Computer Society (2002)
11. Suriadi, S., Foo, E., Jøsang, A.: A user-centric federated single sign-on system. *Journal of Network and Computer Applications* **32**(2) (2009) 388 – 401
12. Bangerter, E., Camenisch, J., Lysyanskaya, A.: A cryptographic framework for the controlled release of certified data. In Christianson, B., Crispo, B., Malcolm, J.A., Roe, M., eds.: Security Protocols Workshop. Volume 3957 of Lecture Notes in Computer Science., Springer (2004) 20–42
13. Camenisch, J., Shelat, A., Sommer, D., Fischer-Hübner, S., Hansen, M., Krasemann, H., Lacoste, G., Leenes, R., Tseng, J.: Privacy and identity management for everyone. In: DIM '05: Proceedings of the 2005 workshop on Digital identity management, New York, NY, USA, ACM (2005) 20–27
14. Leenes, R., Schallaböck, J., Hansen, M.: Privacy and identity management for europe. https://www.prime-project.eu/prime_products/whitepaper/PRIME-Whitepaper-V3.pdf (May 2008)
15. Pearson, S., Mont, M.C., Crane, S.: Persistent and dynamic trust: Analysis and the related impact of trusted platforms. In: iTrust. (2005) 355–363
16. Pearson, S.: Trusted computing: Strengths, weaknesses and further opportunities for enhancing privacy. In: iTrust. (2005) 305–320
17. Syverson, P.F., Goldschlag, D.M., Reed, M.G.: Anonymous connections and onion routing. In: IEEE Symposium on Security and Privacy, IEEE Computer Society (1997) 44–54
18. Vossaert, J., Verhaeghe, P., De Decker, B., Naessens, V.: A smart card based solution for user-centric identity management. In pre-proceedings of the Sixth International PrimeLife/IFIP Summer School on Privacy and Identity Management for Life (August 2010)
19. Camenisch, J., Herreweghen, E.V.: Design and implementation of the *idemix* anonymous credential system. In Atluri, V., ed.: ACM Conference on Computer and Communications Security, ACM (2002) 21–30