# On Privacy-Preserving Ways to Porting the Austrian eID System to the Public Cloud

Bernd Zwattendorfer and Daniel Slamanig

Institute for Applied Information Processing and Communications (IAIK), Graz University of Technology (TUG), Inffeldgasse 16a, 8010 Graz, Austria
{bernd.zwattendorfer,daniel.slamanig}@iaik.tugraz.at

**Abstract.** Secure authentication and unique identification of Austrian citizens are the main functions of the Austrian eID system. To facilitate the adoption of this eID system at online applications, the open source module MOA-ID has been developed, which manages identification and authentication based on the Austrian citizen card (the official Austrian eID) for service providers. Currently, the Austrian eID system treats MOA-ID as a trusted entity, which is locally deployed in every service provider's domain. While this model has indeed some benefits, in some situations a centralized deployment approach of MOA-ID may be preferable. In this paper, we therefore propose a centralized deployment approach of MOA-ID in the public cloud. However, the move of a trusted service into the public cloud brings up new obstacles since the cloud can not be considered trustworthy. We encounter these obstacles by introducing and evaluating three distinct approaches, thereby retaining the workflow of the current Austrian eID system and preserving citizens' privacy when assuming that MOA-ID acts honest but curious.

## 1 Introduction

The Austrian eID system constitutes one major building block within the Austrian e-Government strategy. Secure authentication and unique identification of Austrian citizens – by still preserving citizens' privacy – are the main functions of the Austrian eID system. The basic building block for secure authentication and unique identification in the Austrian eID system is the Austrian citizen card [10], the official eID in Austria.

To facilitate the adoption of this eID concept at online applications, the open source module MOA-ID has been developed. Basically, MOA-ID manages the identification and authentication process based on the Austrian citizen card for various service providers. Currently, the Austrian eID concept treats MOA-ID as a trusted entity, which is deployed locally in every service provider's domain. While this model has indeed some benefits, in some situations a centralized deployment approach of MOA-ID may be preferable. For instance, a centralized MOA-ID can save service providers a lot of operational and maintenance costs. However, in terms of scalability – theoretically the whole Austrian population could use this central service for identification and authentication at service providers – the existing approach is advantageous.

To bypass the issue of scalability, in this paper, we propose a centralized deployment approach of MOA-ID in the public cloud. The public cloud is able to provide nearly unlimited computing resources and hence the scalability problem can easily be compensated. However, the move of a trusted service into the public cloud brings up new obstacles. In particular, MOA-ID, since now running in the public cloud, can no longer be considered a trustworthy entity. We encounter these obstacles by introducing three different approaches, each describing how the current Austrian eID system can be securely migrated into the public cloud. All approaches retain the workflow of the current Austrian eID system and preserve citizens' privacy when assuming that MOA-ID acts honest but curious. The first approach uses both proxy re-encryption and redactable signatures, the second one relies on anonymous credentials, and the third one sets up on fully homomorphic encryption.

## 2 The Current Austrian eID System

In the following subsections we describe the basic ideas of the Austrian eID concept by presenting involved components and processes.

### 2.1 The Austrian Citizen Card Concept

Unique identification and secure authentication are essential processes in e-Government. Particularly, unique identification is essential when a large amount of users comes into play, such as the population of a whole country. In such a huge population, identification of citizens based on first name, last name, and date of birth may be ambiguous. To mitigate this problem, each Austrian citizen is registered in a central register and is assigned a unique identification number. Furthermore, another unique identifier is computed from this number and stored on each citizen card. This so-called sourcePIN is created by a trusted entity, the so-called SourcePIN Register Authority (SRA), and can be used for unique citizen identification at online applications. However, the sourcePIN requires special protection as it is forbidden by law to permanently store the sourcePIN outside the citizen card. Therefore, the Austrian eID concept uses a sector-specific model for identification at online applications. In this sector-specific model, the sourcePIN is used to derive unique sector-specific identifiers, so called sector-specific PINs (ssPINs) for every different governmental sector, e.g., tax, finance, etc. Thereby, citizens' privacy is assured as the sourcePIN cannot be derived from a given ssPIN and different ssPINs of one citizen cannot be linked together.

The key element of the Austrian eID concept constitutes the Austrian citizen card [10], which is basically an abstract definition of a secure eID token possessed by every Austrian citizen. Due to this abstract definition, the Austrian citizen card is a technology-neutral concept, which allows for different implementations. Currently, implementations based on smart cards and mobile phones are in use. In general, the main functions of the Austrian citizen card are 1) *identification and authentication of citizens* and 2) *secure and qualified electronic signature* *creation*. Citizen identification is based on a special data structure (the *Identity Link*), which is solely stored on the Austrian citizen card. This special data structure contains the citizen's first name, last name, date of birth, a unique identifier (sourcePIN), and the citizen's qualified signature certificate. To guarantee its integrity and authenticity, the *Identity Link* is digitally signed by the SourcePIN Register Authority at issuance. Citizen authentication is carried out by creating a qualified electronic signature according to the EU Signature Directive.

### 2.2 Identification and Authentication at Online Services

To facilitate the integration of the citizen card's identification and authentication functionality into online services, the open source module MOA-ID is available. The current Austrian eID system relies on a local deployment model, where MOA-ID is deployed and operated in basically every service provider's domain. Due to that fact, MOA-ID is assumed to be trusted, i.e., it will not leak sensitive information such as the citizen's sourcePIN. Figure 1 illustrates in an abstract way the typical identification and authentication scenario of Austrian citizens using MOA-ID.

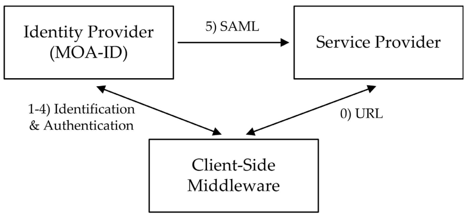
Identity Provider (MOA-ID)
5) SAML
Service Provider
1-4) Identification & Authentication
0) URL
Client-Side Middleware

**Fig. 1.** Simplified illustration of MOA-ID based authentication

**Service Provider:** The service provider usually provides web-based services, which require unique identification and secure authentication by using the Austrian citizen card. This organization can be either a public authority or a private sector company.

**Client-Side Middleware:** The Austrian eID concept foresees an abstract and generic access layer to the citizen card, irrespective of its implementation. The client-side middleware implements this interface, which provides online applications easy access to citizen card functionality without the need of knowing any citizen card specifics. The identity provider MOA-ID uses this interface for accessing diverse citizen card functions.

**Identity Provider (MOA-ID):** MOA-ID represents an identity provider for governmental or private sector service providers. On the one hand, MOA-ID manages the communication with the citizen and her citizen card and, on the other hand, MOA-ID provides specific and authentic citizen card attributes to the service provider for further processing.

In the following we briefly explain an authentication process flow at online services, whereas steps 1 and 2 represent the identification and steps 3 and 4 the authentication process of the Austrian citizen.

**Setup:** The SRA as trusted entity is responsible for managing citizens' Identity Links. Identity Links can be stored on smart card-based citizen card implementations or server-based (in a hardware security module) using the Austrian Mobile Phone Signature.

**Citizen registration:** All Austrian citizens are registered in a central register. In order to activate the citizen card, a citizen must prove her identity, e.g. by using a personal ID. This can be done through various channels, either proving the identity personally in a registration office or via certified mail.

**Service provider registration:** Governmental service providers can be identified either by a special domain ending ("gv.at") or by including a specific object identifier in the service provider's SSL certificate.

**Authentication at online services:**

**1** *Reading and verifying citizen's Identity Link:* After having received an authentication request from the service provider, MOA-ID starts the citizen identification process by requesting the citizen's *Identity Link* through the citizen's client-side middleware. After that, MOA-ID verifies the signature of the returned *Identity Link* to check its integrity and authenticity.
**2** *Calculation of the citizen's* $\mathsf{ssPIN}$ *according to the Austrian eID concept:* MOA-ID calculates the $\mathsf{ssPIN}$ by applying a cryptographic hash function $H$ (SHA-1) to the concatenation of the $\mathsf{sourcePIN}$ and a sector-specific identifier $\mathsf{s}$ of the service provider, i.e., $\mathsf{ssPIN} = H(\mathsf{sourcePIN}\|\mathsf{s})$.
**3** *Requesting the generation of a qualified electronic signature of the citizen:* MOA-ID requests a qualified electronic signature from the citizen through her client-side middleware. By signing a specific message, the citizen gives her consent that she is willing to authenticate at the respective service provider.
**4** *Verification of the citizen signature:* MOA-ID verifies the citizen's qualified signature.
**5** *Assembling citizen identification and authentication data in a structured way and providing it to the service provider:* MOA-ID assembles a special data structure including authentic identity information of the citizen from the *Identity Link*. These data are structured according to the specifications of the Security Assertion Markup Language (SAML, `http://saml.xml.org`) and are delivered to the authentication requesting service provider using a SAML defined protocol, thereby ensuring integrity and authenticity of the data transfer.

## 3 Cryptographic Building Blocks

In this section we introduce the cryptographic building blocks. We note that we do not provide an explicit description of a conventional digital signature scheme ($\mathsf{DSS}$) since this should be clear from the other signature primitives.

### 3.1 Redactable Signatures

A conventional digital signature does not allow for alterations of a signed document without invalidating the signature. However, there are scenarios where it would be valuable to have the possibility to replace or remove (specified) parts of a message after signature creation such that the original signature stays valid (and no interaction with the original signer is required). Signature schemes which allow *removal* of content (replacement by some special symbol $\perp$) by *any* party are called redactable [8], while signature schemes which allow (arbitrary) *replacements* of *admissible* parts by a *designated* party are called sanitizable signature schemes [2]. Below, we present an abstract definition of redactable signatures:

**RS.KeyGen:** This probabilistic key generation algorithm takes a security parameter and produces and outputs a public (verification) key $pk$ and a private (signing) key $sk$.
**RS.Sign:** This (probabilistic) signing algorithm gets as input the signing key $sk$ and a message $m = (m[1], \dots, m[\ell])$, $m[i] \in \{0,1\}^*$ and outputs a signature $\sigma = \mathsf{RS.Sign}(sk, m)$.
**RS.Verify:** This deterministic signature verification algorithm gets as input a public key $pk$, a message $m = (m[1], \dots, m[\ell])$, $m[i] \in \{0,1\}^*$, and a signature $\sigma$ and outputs a single bit $b = \mathsf{RS.Verify}(pk, m, \sigma)$, $b \in \{\mathtt{true}, \mathtt{false}\}$, indicating whether $\sigma$ is a valid signature for $m$.
**RS.Redact:** This (probabilistic) redaction algorithm takes as input a message $m = (m[1], \dots, m[\ell])$, $m[i] \in \{0,1\}^*$, the public key $pk$, a signature $\sigma$, and a list MOD of indices of blocks to be redacted. It returns a modified message and signature pair $(\hat{m}, \hat{\sigma}) = \mathsf{RS.Redact}(m, pk, \sigma, \mathsf{MOD})$ or an error. Note that for any such signature $(\hat{m}, \hat{\sigma})$ we have $\mathsf{RS.Verify}(pk, \hat{m}, \hat{\sigma}) = \mathtt{true}$

## 3.2 Anonymous Signatures

Anonymous signature schemes allow group members to issue signatures on behalf of a group, while hiding for each signature which group member actually produced it. There are several flavors of anonymous signatures: *Group signatures* [1] which involve a dedicated entity (the group manager), who runs a setup and an explicit join protocol for every group member to create the respective members signing key. Furthermore, the group manager is able to open signatures issued by group members to identify the respective signer.

*Ring signatures* [11] are conceptually similar to group signatures, but there is no group manager and the anonymity provided is unconditional. They are "ad-hoc", meaning that a user may take an arbitrary set (ring) of valid public keys to construct a ring signature and the ring represents the anonymity set. We choose to use ring signatures for one of our approaches and present an abstract definition of this signature scheme below, where the key generation is that of a standard digital signature scheme (DSS) and hence omitted here:

**AS.Sign:** This (probabilistic) signing algorithm gets as input the signing key $sk_i$ s.t. $pk_i \in R$, a ring of public keys $R = (pk_1, \dots, pk_n)$, a message $m$ and outputs a signature $\sigma = \mathsf{AS.Sign}(sk_i, R, m)$.
**AS.Verify:** This deterministic signature verification algorithm gets as input a ring of public keys $R = (pk_1, \dots, pk_n)$, a message $m$, and a signature $\sigma$ and outputs a single bit $b = \mathsf{AS.Verify}(R, m, \sigma)$, $b \in \{\mathtt{true}, \mathtt{false}\}$, indicating whether $\sigma$ is a valid signature for $m$ under $R$.

## 3.3 Proxy Re-Encryption

Proxy re-encryption is a public key encryption paradigm where a semi-trusted proxy can transform a message encrypted under the key of party $A$ into another ciphertext, containing the initial plaintext, such that another party $B$ can decrypt with its key. Although the proxy can perform this re-encryption operation, it neither gets access to the plaintext nor to the decryption keys. According to the direction of this re-encryption, such schemes can be classified into bidirectional, i.e., the proxy can transform from $A$ to $B$ and vice versa, and unidirectional, i.e., the proxy can convert in one direction only, schemes. Furthermore, one can distinguish between multi-use schemes, i.e., the ciphertext can be transformed from $A$ to $B$ to $C$ etc., and single-use schemes, i.e., the ciphertext can be transformed only once. We use the unidirectional single-use identity-based proxy re-encryption scheme of [7], but note that we could also use non-identity-based ones.

**RE.Setup:** This probabilistic algorithm gets a security parameter and a value MaxLevel indicating the maximum number of consecutive re-encryptions permitted by the scheme (in case of single-use we set MaxLevel=2). It outputs the master public parameters $params$, which are distributed to users, and the master private key $msk$, which is kept private.

**RE.KeyGen:** This probabilistic key generation algorithm gets $params$, the master private key $msk$, and an identity $id \in \{0,1\}^*$ and outputs a private key $sk_{id}$ corresponding to that identity.

**RE.Enc:** This probabilistic encryption algorithm gets $params$, an identity $id \in \{0,1\}^*$, and a plaintext $m$ and outputs $c_{id} = \mathsf{RE.Enc}(params, id, m)$.

**RE.RKGen:** This probabilistic re-encryption key generation algorithm gets $params$, a private key $sk_{id_1}$ (derived via RE.KeyGen), and two identities $(id_1, id_2) \in \{0,1\}^*$ and outputs a re-encryption key $rk_{id_1 \to id_2} = \mathsf{RE.RKGen}(params, sk_{id_1}, id_1, id_2)$.

**RE.ReEnc:** This (probabilistic) re-encryption algorithm gets as input a ciphertext $c_{id_1}$ under identity $id_1$ and a re-encryption key $rk_{id_1 \to id_2}$ (generated by RE.RKGen) and outputs a re-encrypted ciphertext $c_{id_2} = \mathsf{RE.ReEnc}(c_{id_1}, rk_{id_1 \to id_2})$.

**RE.Dec:** This decryption algorithm gets $params$, a private key $sk_{id}$, and a ciphertext $c_{id}$ and outputs $m = \mathsf{RE.Dec}(params, sk_{id}, c_{id})$ or an error.

## 3.4 Anonymous Credentials

Anonymous credential systems [3,4] enable anonymous attribute-based authentication, i.e., they hide the identity of the credential's owner. Multi-show approaches support unlinkability, i.e., different showings of a credential remain unlinkable and are unlinkable to the issuing [4], while others are one-show [3]. Anonymous credentials are very expressive since they allow to encode arbitrary attributes into the credential. Additonally, during the proof of possession of a credential a user can selectively reveal values of attributes or prove that certain relations among attributes hold, without revealing the attribute values. We use an abstract definition of an anonymous credential system as follows:

**AC.KeyGen:** This probabilistic key generation algorithm is run by an authority and takes a security parameter and produces and outputs a public key $pk$ and a private key $sk$.

**AC.Issue:** This interactive algorithm is run between a user $U$ and an authority $A$. $U$ has as input a list of attributes with corresponding values `attr` and wants to obtain a credential for `attr` ($U$ may also have as input a long term secret). $U$ executes the credential issuing protocol for `attr` with $A$ by using $U$'s input `attr` and $A$ has as input it's private key $sk$. Both algorithms have as input $pk$ and at the end of this interaction $U$ obtains a credential `Cred` corresponding to `attr`.

**AC.Prove:** This interactive algorithm is run between a user $U$ and a verifier $V$. $U$ proves the possession of `Cred` for `attr'`, which represents some subset of `attr`, to a verifier $V$. At the end of the protocol, $V$ outputs `accept` if $U$ has a valid credential `Cred` for `attr'`, otherwise $V$ outputs `reject`.

We note that the Prove algorithm may also be non-interactive, i.e., the credential holder produces a signature of knowledge which can then be given to the verifier to check the validity of the proof locally.

## 3.5 Fully Homomorphic Encryption

Fully homomorphic encryption (FHE) schemes are semantically secure (public-key) encryption schemes which allow arbitrary functions to be evaluated on ciphertexts given the (public) key and the ciphertext. Gentry [5] provided the first construction along with a general blue-print to construct (bootstrap) such schemes from less powerful ones. Since then lots of improvements and alternate approaches have been proposed (cf. [12]). However, it seems to require some more years of research to make them practical in general [6]. A fully homomorphic (public-key) encryption scheme is defined by the following efficient algorithms.

**FHE.KeyGen:** This probabilistic key generation algorithm takes a security parameter and produces and outputs a public-key $pk$, a public evaluation key $evk$, and a private key $sk$.

**FHE.Enc:** This probabilistic encryption algorithm takes a message $m \in \{0,1\}^n$ and a public-key $pk$ and outputs a ciphertext $c = \mathsf{FHE.Enc}(m, pk)$.

**FHE.Dec:** This deterministic algorithm takes a ciphertext $c$ and a private key $sk$ and outputs $m = \mathsf{FHE.Dec}(c, sk)$.

**FHE.Eval:** This homomorphic evaluation algorithm takes an evaluation key $evk$, a function $f : \{0,1\}^n \to \{0,1\}$ and $k$ ciphertexts and outputs a ciphertext $c_f = \mathsf{FHE.Eval}(f, c_1, \ldots, c_k, evk)$.

In this definition messages are bits, but this can easily be generalized to larger spaces. Let us consider arbitrary message spaces in the following. For one approach we need to assume that FHE schemes exists which are "key-homomorphic". Loosely speaking, this means that for each pair of public keys $pk_1$ and $pk_2$ one can derive $f_{1,2}$ and $evk_{1,2}$ such that

$$m = \mathsf{FHE.Dec}(\mathsf{FHE.Eval}(f_{1,2}, \mathsf{FHE.Enc}(m, pk_1), evk_{1,2}), sk_2).$$

This means that by using $f_{1,2}$ one performs a "re-encryption" of $m$ encrypted under $pk_1$ to another ciphertext under $pk_2$, which can then be decrypted using $sk_2$. Such a scheme can trivially be realized using any FHE scheme by letting $f_{1,2}$ represent the circuit, which firstly decrypts the ciphertext $c$ using $sk_1$ obtaining $m$ and then encrypts $m$ using $pk_2$ and $evk_{1,2} = evk_1$. However, since now $sk_1$ would be explicitly wired in the circuit, this would reveal the secret key which is clearly undesirable. Since we are currently not aware of an FHE construction which supports this (loosely defined) property, we need to assume that such a scheme will be available in the future.

## 4 Porting the Austrian eID System to the Public Cloud

The current local deployment model of MOA-ID has some benefits in terms of end-to-end security or scalability, but still some issues can be identified compared to a centralized deployment model of MOA-ID. The adoption of a centralized model may have the following advantages and disadvantages:

On the one hand, the use of one single and central instance of MOA-ID has a clear advantage for citizens as they only need to trust one specific identity provider. In addition, users could benefit from a comfortable single sign-on (SSO). On the other hand, especially service providers can save a lot of costs because they do not need to operate and maintain a separate MOA-ID installation. Nevertheless, still some disadvantages can be identified. Namely, a single instance of MOA-ID constitutes a single point of failure or attack. Particularly, scalability may be an issue as all citizen authentications will run through this centralized system. This is probably the main issue, as theoretically the whole

Austrian population could use this service for identification and authentication at service providers. However, the issue on scalability can be tackled by moving MOA-ID into a public cloud, which is able to theoretically provide unlimited computing resources. Needless to say, a move of a trusted service into the public cloud, however, brings up some new obstacles.

In order to make a migration of the Austrian eID system and MOA-ID into the public cloud possible, we have identified three approaches to adapt the existing Austrian eID system for running it in the public cloud. The adapted Austrian eID system of the respective solution will provide all functions of MOA-ID (identification, $\mathsf{ssPIN}$ generation, and authentication) as in the current status, but protects citizen's privacy with respect to the cloud provider. For providing compact descriptions, we denote the SourcePIN Register Authority by SRA and the Identity Link by $\mathcal{I} = ((A_1, a_1), \ldots, (A_k, a_k))$ as a sequence of attribute labels and attribute values. Let the set of citizens be $C = \{C_1, \ldots, C_n\}$ and the set of service providers be $S = \{S_1, \ldots, S_\ell\}$ as well as the citizen's client-side middleware be denoted as $M$. Moreover, let us assume that Citizen $C_i$ wants to authenticate at service provider $S_j$ who requires the set of attributes $\mathcal{A}_j$ from $\mathcal{I}$ and exactly one "pseudonym", i.e., the $\mathsf{ssPIN}$ for the sector $s$ the service provider $S_j$ is associated to. Additionally, recall that every citizen $C_i$ has a signing key $sk_{C_i}$ stored on the card and the public key $pk_{C_i}$ is publicly available.

## 4.1 Using Proxy Re-Encryption and Redactable Signatures

Here, the Identity Link $\mathcal{I}$ is modified in a way that it does not include the $\mathsf{sourcePIN}$, but additionally all $\mathsf{ssPINs}$ according to all possible governmental sectors. In this augmented Identity Link $\mathcal{I}'$, every attribute $a_i$ is encrypted using an uni-directional single-use proxy re-encryption scheme under a public key (the identity of MOA-ID) such that the corresponding private key is *not* available to MOA-ID and is only known to the SRA. Furthermore, instead of using a conventional digital signature scheme, $\mathcal{I}'$ is signed by the SRA using a redactable signature scheme such that every $a_i$ from $\mathcal{I}'$ can be redacted. The public verification key is available to MOA-ID. Every service provider $S_j$ obtains a key pair for the proxy re-encryption scheme when registering at the SRA. The latter entity produces a re-encryption key, which allows to re-encrypt ciphertexts intended for MOA-ID to $S_j$, and gives it to MOA-ID. Below we present the detailed workflow:

**Setup:** SRA generates $(pk_{\text{SRA}}, sk_{\text{SRA}}) = \mathsf{RS.KeyGen}(\kappa)$, $(params_{\mathsf{RE}}, msk_{\mathsf{RE}}) = \mathsf{RE.Setup}(\kappa, 1)$ as well as $sk_{\text{MOA-ID}} = \mathsf{RE.KeyGen}(params_{\mathsf{RE}}, msk_{\mathsf{RE}}, id_{\text{MOA-ID}})$. It keeps secret $(sk_{\mathsf{RS}}, msk_{\mathsf{RE}}, sk_{\text{MOA-ID}})$ and publishes $params_{\mathsf{RE}}$ as well as $pk_{\mathsf{RS}}$.

**Citizen registration:** The registration of a citizen $C_i$ at the SRA works as it is done now with the exception that $\mathcal{I}'$ includes additional attributes $a_{k+1}, \ldots, a_m$ representing $\mathsf{ssPINs}$ for all sectors. Furthermore, for every $(A_i, a_i) \in \mathcal{I}'$ the SRA replaces $a_i$ by $c_{a_i} = \mathsf{RE.Enc}(params, a_i, id_{\text{MOA-ID}})$ and produces a redactable signature $\sigma_{\mathcal{I}'} = \mathsf{RS.Sign}(sk_{\text{SRA}}, \mathcal{I}')$. Then, $(\sigma_{\mathcal{I}'}, \mathcal{I}')$ is stored on $C_i$'s citizen card.

**Service provider registration:** The registration for service provider $S_j$ at the SRA works as follows. SRA produces a private key $sk_{S_j} = \mathsf{RE.KeyGen}(params_{\mathsf{RE}}, msk_{\mathsf{RE}}, id_{S_j})$ for $S_j$ and a re-encryption key $rk_{\text{MOA-ID} \to S_j} = \mathsf{RE.RKGen}(params, sk_{\text{MOA-ID}}, \text{MOA-ID}, S_j)$ and gives $sk_{S_j}$ to $S_j$ and $rk_{\text{MOA-ID} \to S_j}$ to MOA-ID respectively.

**Authentication at online services:**

**1 & 2:** After having received an authentication request from $S_j$, MOA-ID starts the citizen identification process by requesting $C_i$'s Identity Link $\mathcal{I}'$ through $M$. Thereby, we have two possibilities:

1. If MOA-ID tells $M$ which attributes $\mathcal{A}_j$ are required by $S_j$, then $M$ runs $(\hat{\mathcal{I}'}, \hat{\sigma_{\mathcal{I}'}}) = \mathsf{RS.Redact}(\mathcal{I}', pk_{\mathsf{RS}}, \sigma_{\mathcal{I}'}, \mathsf{MOD})$ wheres $\mathsf{MOD}$ contains all the indices of $c_{a_i}$ from $\mathcal{I}'$ with exception of $\mathcal{A}_j$ (including the $\mathsf{ssPIN}$ required by $S_j$). Then, $M$ sends $(\hat{\mathcal{I}'}, \hat{\sigma_{\mathcal{I}'}})$ to MOA-ID which runs $b = \mathsf{RS.Verify}(pk_{\mathsf{RS}}, \hat{\mathcal{I}'}, \hat{\sigma_{\mathcal{I}'}})$ and proceeds if $b = \mathtt{true}$ and aborts otherwise.
2. $M$ sends $(\mathcal{I}', \sigma_{\mathcal{I}'})$ to MOA-ID which runs $b = \mathsf{RS.Verify}(pk_{\mathsf{RS}}, \mathcal{I}', \sigma_{\mathcal{I}'})$ and proceeds if $b = \mathtt{true}$ and aborts otherwise. Then, MOA-ID runs $(\hat{\mathcal{I}'}, \hat{\sigma_{\mathcal{I}'}}) = \mathsf{RS.Redact}(\mathcal{I}', pk_{\mathsf{RS}}, \sigma_{\mathcal{I}'}, \mathsf{MOD})$, whereas $\mathsf{MOD}$ contains the indices of all attributes in $\mathcal{I}'$ with exception of $\mathcal{A}_j$ (including the $\mathsf{ssPIN}$ required by $S_j$).

**3:** In this step, MOA-ID usually requests the generation of a qualified electronic signature from $C_i$. Here we have the following possibilities:

1. MOA-ID requests no signature, since $\mathcal{I}'$ is signed and only available to $C_i$.
2. $M$ produces a standard signature $\sigma = \mathsf{DSS.Sign}(sk_{C_i}, m^*)$ for a special message $m^*$ on behalf of $C_i$ (which, however, allows unique identification of $C_i$ by MOA-ID).
3. $M$ produces a ring signature $\sigma = \mathsf{AS.Sign}(sk_{C_i}, R, m^*)$ for a special message $m^*$ on behalf of ring $R$ including $pk_{C_i}$.

**4:** MOA-ID verifies the validity of signature $\sigma$ either by running $b = \mathsf{DSS.Verify}(pk_{C_i}, m^*, \sigma)$ or $b = \mathsf{AS.Verify}(R, m^*, \sigma)$ (note that due to $\sigma_{\mathcal{I}'}$ and it's potentially redacted version can always be linked together, it is advisable that every citizen $C_i$ uses a fixed ring all the time, i.e., all citizens in $R$ use the same ring, since otherwise, e.g., when they are sampled uniform at random, then intersection attacks on the rings will soon reveal $C_i$.).

**5:** MOA-ID takes all remaining attributes $c_{a_i}$ from $\mathcal{I}'$ (or $\hat{\mathcal{I}'}$) and computes for every such attribute $c'_{a_i} = \mathsf{RE.ReEnc}(c_{a_i}, rk_{\text{MOA-ID}\to S_j})$ and assembles all these resulting $c'_{a_i}$ into the SAML structure, which is then communicated to $S_j$. $S_j$ can then decrypt all the attributes using $sk_{S_j}$.

## 4.2 Using Anonymous Credentials

The Identity Link $\mathcal{I}$ is augmented to $\mathcal{I}'$ in a way that it does not include the $\mathsf{sourcePIN}$, but additionally all $\mathsf{ssPIN}$'s. Now, the SRA issues an anonymous credential `Cred` to every citizen for `attr` being all attributes in $\mathcal{I}'$. Essentially, a citizen then authenticates to a service provider by proving to MOA-ID the possession of a valid credential, i.e., MOA-ID checks whether the credential has been revoked or not. Note that for one show credentials, if the entire credential `Cred` is shown to MOA-ID, this amounts to a simple lookup in a blacklist. If the credential is not revoked, MOA-ID signs the credential to confirm that it is not revoked and the citizen performs via $M$ a (non-interactive) proof by revealing the necessary attributes $\mathcal{A}_j$ including the required $\mathsf{ssPIN}$ to $S_j$, who can then in turn verify the proof(s) as well as MOA-ID's signature.

**Setup:** SRA generates $(pk_{\text{SRA}}, sk_{\text{SRA}}) = \mathsf{AC.KeyGen}(\kappa)$ and keeps secret $sk_{\text{SRA}}$ and publishes $pk_{\text{SRA}}$. Furthermore, MOA-ID produces a key pair for a digital signature scheme $(pk_{\text{MOA-ID}}, sk_{\text{MOA-ID}}) = \mathsf{DSS.KeyGen}(\kappa)$ and publishes $pk_{\text{MOA-ID}}$.

**Citizen registration:** At registration of citizen $C_i$ at the SRA a modified Identity Link $\mathcal{I}'$ is generated, which includes additional attributes $a_{k+1}, \ldots, a_m$ representing $\mathsf{ssPIN}$s for all sectors and other citizen attributes. Then, SRA and $C_i$ run $\mathsf{AC.Issue}$ and the resulting credential `Cred` is stored on $C_i$'s Citizen Card.

**Service provider registration:** The registration for service provider $S_j$ works as it is done now.

**Authentication at online services:**

**1, 2 & 3:** After having received an authentication request from $S_j$, MOA-ID starts the citizen identification process by requesting $C_i$'s credential $\mathtt{Cred}$ and checks whether $\mathtt{Cred}$ has not been revoked. If $\mathtt{Cred}$ has not been revoked MOA-ID produces a signature $\sigma = \mathsf{DSS.Sign}(sk_{\text{MOA-ID}}, \mathtt{Cred}, \sigma)$ and sends $\sigma$ along with a description of $\mathcal{A}_j$ to $M$.

**4:** $M$ runs $b = \mathsf{DSS.Verify}(pk_{\text{MOA-ID}}, \mathtt{Cred}, \sigma)$ and if $b = \mathtt{true}$ produces a non-interactive proof $\pi$ which opens all attribute values of $\mathcal{A}_j$ including the ssPIN required by $S_j$ and sends $(\mathtt{Cred}, \pi, \sigma)$ to $S_j$. Otherwise, $M$ aborts.

**5:** $S_j$ computes $b = \mathsf{DSS.Verify}(pk_{\text{MOA-ID}}, \mathtt{Cred}, \sigma)$ and if $b = \mathtt{true}$ verifies the proof $\pi$. If both checks verify, $C_i$ is authenticated, otherwise $S_j$ aborts.

Note that in this approach Cred is shown to MOA-ID, which however does not reveal the attribute values but makes revocation easier, since it only requires blacklist lookups. One could also use multi-show credentials, whereas $M$ would then have to perform a proof with MOA-ID which convinces MOA-ID that the credentials are not revoked [9], which provides stronger privacy guarantees.

## 4.3 Using Fully Homomorphic Encryption

This approach is a rather theoretic one and requires an FHE scheme which is also "key-homomorphic" as already discussed before. The idea for this approach is the following: The Identity Link $\mathcal{I}$ of a citizen holds the same attributes as now (and in particular the sourcePIN), but every attribute $a_i$ is encrypted using an FHE scheme with the above described property under MOA-ID's public key for which MOA-ID does *not* hold the private key. Furthermore, this resulting $\mathcal{I}'$ is conventionally signed by the SRA. Then, for authentication at $S_j$, the resulting $\mathcal{I}'$ and the signature $\sigma$ are sent to MOA-ID who checks the signature and homomorphically computes the respective ssPIN from the encrypted sourcePIN (without learning neither sourcePIN nor ssPIN). Then, for all encrypted attributes required by $S_j$ (including the afore computed encrypted ssPIN), MOA-ID performs the "FHE re-encryption" to $S_j$'s public key. On receiving the respective information from MOA-ID, the service provider can decrypt all attribute values.

**Setup:** SRA generates $(pk_{\text{MOA-ID}}, evk_{\text{MOA-ID}}, sk_{\text{MOA-ID}}) = \mathsf{FEH.KeyGen}(\kappa)$ and keeps secret $sk_{\text{MOA-ID}}$ and publishes $(pk_{\text{MOA-ID}}, evk_{\text{MOA-ID}})$. Furthermore, SRA produces a key pair for a digital signature scheme $(pk_{\text{SRA}}, sk_{\text{SRA}}) = \mathsf{DSS.KeyGen}(\kappa)$ and publishes $pk_{\text{SRA}}$.

**Citizen registration:** During registration of citizen $C_i$ at the SRA, for every $(A_i, a_i) \in \mathcal{I}$ SRA replaces $a_i$ by $c_{a_i} = \mathsf{FHE.Enc}(a_i, pk_{\text{MOA-ID}})$ and produces a signature $\sigma_{\mathcal{I}'} = \mathsf{DSS.Sign}(sk_{\text{SRA}}, \mathcal{I}')$. Then, $(\sigma_{\mathcal{I}'}, \mathcal{I}')$ is stored on $C_i$'s citizen card.

**Service provider registration:** For the registration of service provider $S_j$, SRA computes $(pk_{S_j}, evk_{S_j}, sk_{S_j}) = \mathsf{FEH.KeyGen}(\kappa)$ as well as $evk_{\text{MOA-ID},S_j}$ and $f_{\text{MOA-ID},S_j}$, and gives $sk_{S_j}$ to $S_j$ as well as $evk_{\text{MOA-ID},S_j}$ and $f_{\text{MOA-ID},S_j}$ to MOA-ID.

**Authentication at online services:**

**1 & 2:** After having received an authentication request from $S_j$, MOA-ID starts the citizen identification process by requesting $C_i$'s Identity Link $\mathcal{I}'$ and its corresponding signature $\sigma_{\mathcal{I}'}$. MOA-ID runs $b = \mathsf{DSS.Verify}(pk_{\text{SRA}}, \mathcal{I}', \sigma_{\mathcal{I}'})$ and proceeds if $b = \mathtt{true}$ and aborts otherwise. Let $c_{a_k}$ be the encrypted sourcePIN, then MOA-ID computes $c'_{a_k} = \mathsf{FHE.Eval}(f_H, c_{a_k} \| \mathsf{FHE.Enc}(s_j, pk_{\text{MOA-ID}}), evk_{\text{MOA-ID}})$ where $s_j$ is the sector specific identifier required by $S_j$ and $f_H$ is a circuit representing the evaluation of the SHA-1 hash function, which is used for ssPIN generation.

**3:** In this step MOA-ID requests the generation of a qualified electronic signature from $C_i$. Here we have the following possibilities:
  1. MOA-ID requests no signature, since $\mathcal{I}'$ is signed and only available to $C_i$.
  2. $M$ produces a standard signature $\sigma = \mathsf{DSS.Sign}(sk_{C_i}, m^*)$ for a special message $m^*$ on behalf of $C_i$ (which, however, allows unique identification of $C_i$ by MOA-ID).
  3. $M$ produces a ring signature $\sigma = \mathsf{AS.Sign}(sk_{C_i}, R, m^*)$ for a special message $m^*$ on behalf of ring $R$ including $pk_{C_i}$.

**4:** MOA-ID verifies the validity of signature $\sigma$ either by running $b = \mathsf{DSS.Verify}(pk_{C_i}, m^*, \sigma)$ or $b = \mathsf{AS.Verify}(R, m^*, \sigma)$.

**5:** MOA-ID takes all attributes $c_{a_i}$ in $\mathcal{A}_j$ from $\mathcal{I}'$ including $c_{a_k}$ and computes for every such attribute $\hat{c}_{a_i} = \mathsf{FHE.Eval}(f_{\text{MOA-ID},S_j}, c_{a_i}, evk_{\text{MOA-ID},S_j})$, thus performing a re-encryption to $pk_{S_j}$, and assembles all these resulting $\hat{c}_{a_i}$ into the SAML structure, which is then communicated to $S_j$. $S_j$ can now decrypt all attributes using $sk_{S_j}$.

## 5 Evaluation

In this section we evaluate the different approaches based on selected criteria targeting several aspects, e.g. evaluating the overall architecture or aspects

**Table 1.** Evaluation of the various approaches. We use ✓ to indicate as the criterion being full applicable, × as not applicable, and ≈ as partly applicable. For quantitative criteria we use L for *low*, M for *medium*, and H for *high*.

**Re-use of existing infrastructure:** How much of the existing infrastructure of the Austrian eID system can be re-used or do a lot of parts need to be exchanged or modified?
**Conformance to current workflow:** Is the authentication process flow of the approach conform to the existing citizen card authentication process flow?
**Scalability:** Is the approach applicable in a large scale or not?
**Practicability:** Can the authentication process be carried out within a reasonable time frame?
**Extensibility:** Is the applied infrastructure of the approach easily extensible to new requirements, e.g., adding new sectors and thus requiring new ssPINs.
**Middleware complexity:** Does the approach require high complexity or computational power from the client-side middleware?
**Service provider effort:** How much effort is required by the service provider adopting a particular approach?
**Trust in MOA-ID:** Does the approach require MOA-ID being trusted?
**Anonymity:** Does the approach support citizens to be anonymous with respect to MOA-ID?
**Unlinkability:** Are users unlinkable to MOA-ID, i.e., can different authentications of one citizen be linked together?
**Authentication without prior registration:** The current Austrian eID system allows registration-less authentications. Hence, is this feature still possible or not?

| **Criterion** | **Approach 1** | **Approach 2** | **Approach 3** |
|---|---|---|---|
| Re-use of existing infrastructure | ≈ | ≈ | ≈ |
| Conformance to current workflow | ✓ | ≈ | ✓ |
| Scalability | ✓ | ✓, ≈ | ✓ |
| Practicability | ✓ | ✓, ≈ | × |
| Extensibility | × | ≈ | ✓ |
| Middleware complexity | L | L, H | L |
| Service provider effort | L | M | H |
| Trust in MOA-ID | L | L | L |
| Anonymity | ×, ✓ | ✓ | ×, ✓ |
| Unlinkability | × | ×, ✓ | × |
| Authentication without prior registration | ✓ | ✓ | ✓ |

regarding the individual entities. We briefly describe the selected criteria for evaluation below and Table 1 shows a comparison of our three approaches.

In the following, we give some explanations why specific criteria could be fulfilled, partly fulfilled, or not-fulfilled by the respective approach.

**Re-Use of Existing Infrastructure:** This criterion can only be partly fulfilled by all approaches since all approaches require some modification of the existing Austrian eID infrastructure. Approach 1 and 3 require some kind of additional governance structure, as proxy re-encryption keys for service providers have to be generated and managed by SRA. Additionally, the attribute values of the existing Identity Link structure must be exchanged by encrypted values and the Identity Link needs to be augmented. For approach 1, the conventional signature of the Identity Link must also be exchanged by a redactable signature. In contrast to that, Approach 2 using anonymous credentials requires a complete re-structuring of the Identity Link. However, all approaches can still rely on the same basic architectural concept of the Austrian eID infrastructure, using MOA-ID as identity provider.

**Conformance to Current Workflow:** Approach 1 and 3 fully comply with the current citizen card authentication process flow, hence they follow the steps identification, ssPIN provision, and authentication. Approach 2 is slightly different, as MOA-ID just checks if a provided credential is not revoked. The actual verification of the credential is carried out directly at the service provider.

**Scalability:** Basically, all approaches can be adopted in a large scale. Approach 1 and 3 are similar to the existing Austrian eID system, as only a few attributes need to be exchanged within the Identity Link and the computational requirements for the middleware remain low. For approach 2, it must be distinguished whether one-show or multi-show anonymous credentials will be used. For one-show credentials, revocation checking is a very light-weight process and hence easy adoptable. In contrast to that, revocation for multi-show credentials is much more complex and not easily applicable for a large amount of users such as the Austrian population. Finally, any scalability doubts concerning MOA-ID can be neglected as it is running in a public cloud providing nearly unlimited resources.

**Practicability:** Approach 1 and 2 seem to be to date the most promising practical approaches. Approach 1 relies only on cryptographic mechanisms, which can already efficiently implemented. For approach 2, again we must distinguish between one-show and multi-show credentials. For one-show credentials, proof generation requires moderate effort. For multi-show credentials, proof generation for non-revocation proofs is complex and computational expensive. This gives a lot of load to the client-side middleware, which makes approach 2 using multi-show credentials quite impracticable. For approach 3, the assumptions we made for FHE still require further research activities and are far away from any implementation. Although we rely on public clouds, FHE is currently not practicable.

**Extensibility:** For adding new sectors, approach 1 would require a full exchange of the Identity Link, as it must be re-signed when adding a new encrypted ssPIN. The same issue holds for approach 2, since a new credential incorporating the new ssPIN must be stored on the citzen card with exception when using scope-exclusive pseudonyms as proposed in ABC4Trust (`https://abc4trust.eu`). In approach 3, ssPIN's are computed from the encrypted version of the sourcePIN and no modifications of the Identity Link are required.

**Middleware Complexity:** In approach 1, client-side middleware complexity is low as only redaction of the Identity Link is required. Middleware complexity in approach 2 depends on the type of anonymous credentials used. Proof computation of multi-show credentials is computational expensive, which would impose a significant computational burden on $M$ [9] when taking into account that the system covers all citizens of Austria. For approach 3, middleware complexity is low again as its functionality is equal to current middleware implementations.

**Service Provider Effort:** The effort for service providers adopting approach 1 is low. Service providers just need to verify the data received by MOA-ID and do some decryption operations. For approach 2, the effort is slightly higher because service providers need to set up appropriate verification mechanisms for the claims provided by the user. The effort for service providers in approach 3 is the highest as FHE decryption is currently still computationally expensive.

**Trust in MOA-ID:** Since no sensitive citizen data such as the sourcePIN or any ssPIN are revealed to MOA-ID, no trust is required. In approach 1 and 3 MOA-ID only sees encrypted citizen data. In approach 2 MOA-ID does only see the credential but non of its attribute values. However, some trust assumptions are required that MOA-ID works correctly.

**Anonymity:** For approach 2, anonymity is obvious as the whole approach sets up on anonymous credentials. Achieving anonymity in approach 1 and 3 depends on the sub-processes to be chosen for citizen authentication (signature creation). Both approaches 1 and 3 rely on three similar alternative sub-processes. Sub-process 1 does not request a citizen signature and fully relies on the Identity Link's signature for citizen authentication, as the Identity Link is only available to the citizen. In this case, the citizen stays fully anonymous in the face of MOA-ID. In sub-process 2, citizen signature creation is requested by MOA-ID for citizen authentication. In this case, citizens are uniquely identifiable by MOA-ID due to $pk_{C_i}$. Finally, within sub-process 3 ring signatures are created and enable citizen anonymity with respect to the defined ring.

**Unlinkability:** For our approaches, it is very hard to achieve unlinkability with respect to MOA-ID. In approach 1 and 3 citizens are linkable because they always present the same Identity Link and corresponding signature. Citizens could only be unlinkable in approach 2, where one-show credentials provide linkability and multi-show credentials provide unlinkability.

**Authentication without Prior Registration:** This criterion can still be fulfilled by all of our approaches.

## 6 Conclusions

Based on the results of our evaluation, we conclude that all approaches might be feasible but not all of them might be really practical when considering an implementation of a cloud-based approach instead of the current Austrian eID system. Approach 1 might be the best as it could be quickly realized and requires less effort for the client-side middleware and the service provider. However, linkability and higher efforts for extensions are the drawbacks of this approach. Depending on the type of anonymous credential system, approach 2 might also be practicable and possible to implement. Although it provides more complexity and efforts for the client-side middleware, compared to approach 1 it could provide full anonymity and unlinkability. Finally, although approach 3 has its advantages, e.g., in terms of extensibility, and would be promising for the future, it is currently not practicable. Implementations of fully homomorphic encryption schemes are currently still in the early stages which definitely hinder a fast adoption of this approach.

**Acknowledgements.** We would like to thank the anonymous reviewers for their valuable suggestions. The second author has been supported by the European Commission through project FP7-FutureID, grant agreement number 318424.

## References

1. Ateniese, G., Camenisch, J.L., Joye, M., Tsudik, G.: A Practical and Provably Secure Coalition-Resistant Group Signature Scheme. In: Bellare, M. (ed.) CRYPTO 2000. LNCS, vol. 1880, pp. 255–270. Springer, Heidelberg (2000)
2. Ateniese, G., Chou, D.H., de Medeiros, B., Tsudik, G.: Sanitizable Signatures. In: de Capitani di Vimercati, S., Syverson, P.F., Gollmann, D. (eds.) ESORICS 2005. LNCS, vol. 3679, pp. 159–177. Springer, Heidelberg (2005)
3. Brands, S.: Rethinking Public Key Infrastructures and Digital Certificates: Building in Privacy. MIT Press (2000)
4. Camenisch, J., Lysyanskaya, A.: An Efficient System for Non-transferable Anonymous Credentials with Optional Anonymity Revocation. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 93–118. Springer, Heidelberg (2001)
5. Gentry, C.: Fully Homomorphic Encryption using Ideal Lattices. In: ACM STOC 2009, pp. 169–178. ACM (2009)
6. Gentry, C., Halevi, S., Smart, N.P.: Homomorphic evaluation of the AES circuit. In: Safavi-Naini, R. (ed.) CRYPTO 2012. LNCS, vol. 7417, pp. 850–867. Springer, Heidelberg (2012)
7. Green, M., Ateniese, G.: Identity-Based Proxy Re-encryption. In: Katz, J., Yung, M. (eds.) ACNS 2007. LNCS, vol. 4521, pp. 288–306. Springer, Heidelberg (2007)
8. Johnson, R., Molnar, D., Song, D., Wagner, D.: Homomorphic Signature Schemes. In: Preneel, B. (ed.) CT-RSA 2002. LNCS, vol. 2271, pp. 244–262. Springer, Heidelberg (2002)

9. Lapon, J., Kohlweiss, M., De Decker, B., Naessens, V.: Analysis of Revocation Strategies for Anonymous Idemix Credentials. In: De Decker, B., Lapon, J., Naessens, V., Uhl, A. (eds.) CMS 2011. LNCS, vol. 7025, pp. 3–17. Springer, Heidelberg (2011)
10. Leitold, H., Hollosi, A., Posch, R.: Security Architecture of the Austrian Citizen Card Concept. In: ACSAC 2002, pp. 391–402 (2002)
11. Rivest, R.L., Shamir, A., Tauman, Y.: How to leak a secret: Theory and applications of ring signatures. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 552–565. Springer, Heidelberg (2001)
12. Vaikuntanathan, V.: Computing Blindfolded: New Developments in Fully Homomorphic Encryption. In: IEEE FOCS 2011, pp. 5–16 (2011)