Programma eID
Afsprakenstelsel

Polymorphic Pseudonymization

programma eID

SAP working group

Version: 0.91

Date: 07 July 2014

Status: Draft

Date	Version	Change
20-05-2014	0.88	Initial version
22-05-2014	0.89	Inversion added in second part of proposition 2.1 (and related one on p.16). Remark added after proposition 2.1 regarding other ElGamal representations.
02-06-2014	0.90	Typo’s fixed in third part of proposition 2.1 and in equation (9) on p.15.
07-07-2014	0.91	• Improvements added based on work group discussions • Some further typos in formulas fixed. • Added *affine* PPCA and proofs for both basic PPCA as for affine PPCA in appendices A and B

Polymorphic Pseudonymization

(DRAFT)

Dutch e-ID programme
SAP Working group

07 July 2014

Abstract In this paper we introduce polymorphic pseudonymization based on the homomorphic properties of ElGamal encryption. We use this to develop an e-ID infrastructure with special security and privacy properties. We also describe an anonymous card application (PPCA). This allows for pseudonymous identification through a central identity provider with the paradoxical property that the provider cannot establish which card is actually used or link it with earlier usage of the card. This provides for anonymity in the scheme without losing computability with other means of authentication.

1 Introduction

In this paper we introduce polymorphic pseudonymization. This enables an electronic identity (e-ID) infrastructure based on existing electronic identity mechanisms where users are provided with consistent pseudonyms throughout the whole infrastructure. The scheme involves several redirections of the user and secure exchange of information between the e-ID parties. This part of the scheme could be based on SAML, cf. [18], but this is outside the scope of this paper. Below we introduce the parties that exist within the envisioned e-ID infrastructure and the main (security) requirements we require. Compare Figure 1 below.

- Users
  Users want to make use of an electronic service provided by a service provider. The cases relevant for this paper are those where the user needs to authenticate himself to the service provider. This can for instance be required if the electronic service rendered represents a monetary value as in video/audio streaming services. It can also be required when the service gives access to personal data of the user. The role of user authentication within the e-ID infrastructure takes place at identity providers.
  In many cases it is not required that the service provider needs to know the actual identity of the user or indeed other personal information (*attributes*) of the user. To this end, in the designed e-ID infrastructure the service provider is only provided with a pseudonym of the user by an identity provider after successful authentication. The e-ID infrastructure also supports that

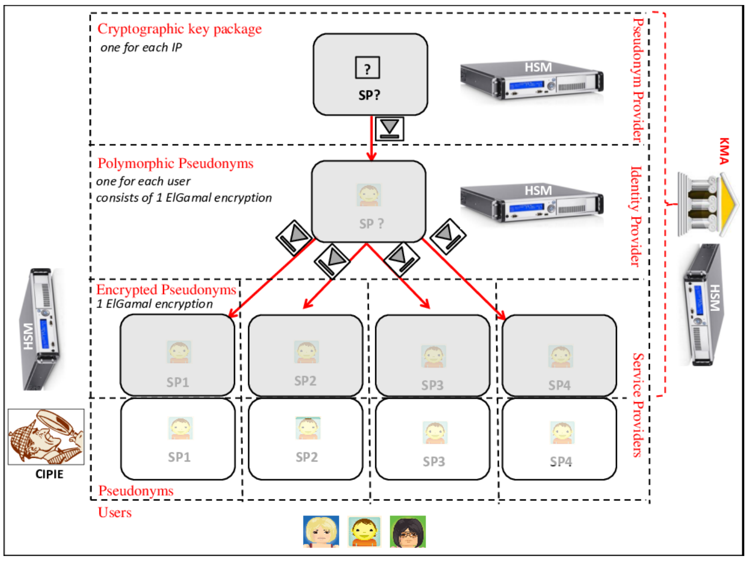

Figure 1. The e-ID infrastructure envisioned

pseudonyms - with user consent - are supplemented with attributes or (legal) authorisations to perform certain acts for someone else, see below. A functional requirement is that the user pseudonym the service provider is given, is independent of the identity provider that is used. That is, login on to a service provider by the user through different identity providers results in the same pseudonym. The number of users will typically be very large, e.g. several millions.

- Service Providers (SPs)
  Service providers identify their clients using pseudonyms from the e-ID infrastructure. Service providers receive pseudonyms in an encrypted form from the e-ID infrastructure possibly supplemented with attributes and authorizations related to the pseudonym, i.e. the user. As part of their registration process service providers are given cryptographic keys from the e-ID KMA allowing them to decrypt the messages from the identity providers holding the user pseudonyms. The number of service providers will typically be large, e.g. $\geq$ 100.000.
- e-ID Brokers, Attribute Providers, Authorization Providers
  Several parties can assist service providers to supplement its information. A service provider can - with user consent - request encrypted pseudonyms for the user from identity providers. It will typically ask one for itself but, with consent of the user, it can also ask encrypted pseudonyms under which the user is known by other parties such as attribute and authorization providers. By using these the service provider is able to retrieve user attributes and authorizations. An e-ID broker sits between the various e-ID parties and the

service provider and facilitates the latter. Typically an e-ID broker works for many service providers.

- Identity Providers (IPs)
  Identity providers have means to electronically identify their clients and facilitate their clients to identify themselves to the service providers by pseudonyms. Only parties that deploy suitable reliable client registration processes and have suitable strong authentication mechanisms can participate in the e-ID infrastructure we envision. Compare the STORK requirements in [21]. Other than that we want to impose as little as possible technical limitations on organisations to participate as identity providers in the e-ID infrastructure. Technically we only require that as a part of the user registration process the participating organisation records sufficient information and is able to uniquely identify the user in the e-ID infrastructure. That is, we require participating identity providers to be able to compile an e-ID wide *U-id* of a user. A naive e-ID scheme could be formed by letting these identity providers send this *U-id* to the service providers. However, this would not be very privacy friendly as the service providers would then get the identity of the users and the service providers would also be able to link their client databases. Our scheme effectively adds a privacy enhancing layer to this simple setup by encrypting the U-ID in such away that many of the functional benefits of the naive scheme remain but the privacy issues are removed.
  To further explain, after successful authentication the identity provider compiles the *U-id*, sends this to the *pseudonym provider* (see below) and requests a *polymorphic pseudonym* related to the *U-id*. The identity provider typically records a polymorphic pseudonym for the user provided by the pseudonym provider so that it does not need to contact the pseudonym provider in future identifications. In other words, the identity providers receive a polymorphic pseudonym for the user as part of registering the user into the e-ID infrastructure and update that information periodically.
  We mention that it might be very helpful to let a social security number be part of the *U-id* as this will ensure uniqueness of the *U-id* and thus of the e-ID pseudonyms. However, not all potential Identification Providers have registered - or in fact are legally allowed to register - that number. Alternatively, as part of the e-ID infrastructure registration at an identity provider, the user could be requested to subsequently “login” to the governmental infrastructure so that the user *U-id* can be augmented with its social security number before it is sent to the pseudonym provider.
  Each identity provider is provided with cryptographic keys in protected form through an Hardware Security Module (HSM) facilitated by the e-ID KMA. With these keys and the identifier of the service provider, the identity provider performs two operations. One particular key and the identifier are used to implicitly form a pseudonym for the service provider inside the polymorphic pseudonym. Another key and the identifier is next used to transform this result to an encrypted pseudonym that can only be used by the service provider.That is, before the transformation the polymorphic pseudonym is related to the user but not related to and decryptable for any service provider

(or identity provider). After the transformation an encrypted pseudonym is generated that holds the pseudonym of the user in the domain of the service provider in an encrypted form. That is, only the service provider is able to decrypt this. The number of identity providers in the e-ID infrastructure will be small, e.g. less than a hundred.

- Pseudonym Provider (PP)
  The pseudonym provider receives the unique user identification *U-id* from an identity provider after successful user identification. The pseudonym provider is provided by the KMA with a specific public key for each participating identity provider. The pseudonym provider is also provided cryptographic keys in its HSM by the KMA and in fact generates some keys on its own. The pseudonym provider then uses its HSM to form a polymorphic pseudonym for the identity provider. This is both specific for the identity provider and the user. We envision only one pseudonym provider in the e-ID infrastructure, although in principle there could be several.
- Key Management Authority (KMA)
  The KMA generates and manages the cryptographic keys used by all parties within the scheme. We envision only one KMA in the e-ID infrastructure, although in principle there could be several. Specifically, we envision the situation that there are sector specific KMAs and KMAs for HSM based service providers and non-HSM based service providers.
- Central Information Point e-ID Investigations(CIPEI)
  It needs to be possible that law enforcement agencies can efficiently investigate criminal abuse of the e-ID infrastructure pseudonymity features. We envision that this implies that functionality related to (re-)pseudonymization needs to be available for law enforcement agencies. In our scheme this is facilitated by the CIPEI. The operations of the CIPEI need to be legally controlled, e.g. by using court orders, and independently audited and supervised. Although we technically facilitate control, the further setup is outside the scope of this paper. One might argue that the existence of law enforcement access introduces the risk of misuse itself and is therefore harming the privacy properties of the e-ID scheme. On the other hand one might also argue that a service provider that knows that law enforcement access is possible, e.g. in case of fraud, will be less inclined to ask personal data of its users. That is, one can argue that the CIPEI can also contribute to the privacy of the e-ID scheme.
  The CIPEI we envision can handle two kinds of requests. In the basic request ("de-pseudonymization") a law enforcement agency presents a pseudonym and the service provider domain it comes from to the CIPEI and requests the identity of the user behind the pseudonym. A use case here could be a criminal complaint from a service provider accusing a user of fraud. The law enforcement agency would then be able to determine the identity of the user, e.g., to question him. In the second type of request ("pseudonymization on request") the law enforcement agency presents a *U-id* of a user and a reference to a service provider and requests the pseudonym of that user in the given domain. A use case here could be a criminal investigation against a

user of child grooming and the law enforcement agency wants to investigate if the user has been active on certain other websites as well. Cryptographically, both request types require different cryptographic information and can be handled independently. This implies that the CIPEI could be split into two parts.

Based on the above description we formulate the main pseudonymization security requirements in an informal manner:

1. A pseudonym does not allow the identification of the user through cryptographic means.
2. Pseudonym domains of the service provider are not cryptographically linkable. That is, at different service providers the user gets different pseudonyms which are not inter-linkable through cryptographic means.
3. Possession of many pseudonyms and the corresponding user identities does not allow the creation of an additional user pseudonym.
4. Encrypted pseudonyms are *indistinguishable* for the e-ID parties, most notably for the brokers. That is, the encrypted pseudonyms for the same user at the same service provider should not be linkable by other parties than the intended service provider and possibly the KMA and CIPEI.
5. Although the identity provider is performing the transformation from polymorphic pseudonym to encrypted pseudonym he is not able to access the pseudonym it helps to form.
6. An identity provider cannot use the polymorphic pseudonyms of another identity provider to extract encrypted pseudonyms from.

Outline of the paper
In Section 2 we introduce the cryptographic building blocks for the scheme based on the ElGamal encryption scheme. Section 3 describes the cryptographic key setup of our scheme and Section 4 describes the scheme itself. Section 5 contains the arguments that the scheme is meeting the security requirements mentioned above. In Section 6 we sketch various extensions of our scheme. Section 7 contains the conclusions of this paper. Finally, in Appendices A and B we have further formalized the security properties related to a card application (PPCA) which can provide anonymity in the scheme without losing computability with other means of authentication.

2 Notation and preliminaries

Throughout this paper we let $G = \langle g \rangle$ be a multiplicative group of prime order q generated by a generator element g. By GF(q) we denote the Galois field of the integers modulo q. The cryptographic security of G can be formulated in four problems in the context of the Diffie-Hellman key agreement protocol with respect to g. The first one is the *Diffie-Hellman problem*, which consists of computing the values of the function $DH_g(g^x, g^y) = g^{xy}$. Two other problems are related to the Diffie-Hellman problem. The first one is the *Decision Diffie-Hellman* (DDH) problem with respect to g: given $\alpha, \beta, \delta \in G$ decide whether $\delta = DH_g(\alpha, \beta)$ or not. The DH problem with respect to g is at least as difficult as the DDH problem with respect to g. The second related problem is the *discrete logarithm* (DL) problem in G with respect to g: given $\alpha = g^x \in G$, with $x \in \mathrm{GF}(q)$ then find $x = DL_g(\alpha)$. The DL problem with respect to g is at least as difficult as the DH problem with respect to g.

One can easily show that if one can solve the discrete logarithms with respect to one generator, one can solve it with to any generator of G. That is, the hardness of the discrete logarithm problem is independent of the generator of the group. In [22] a similar property is shown for the Diffie-Hellman problem. It seems very unlikely that the hardness of the Decision Diffie-Hellman is dependent of the generator of the group. However, as far as we know such a result is not known to be provable. To this end, we say that one can solve the Decision Diffie-Hellman problem with respect to the group G if one can solve the Decision Diffie-Hellman problem with respect to any generator of the group. We assume that all four introduced problems in G are intractable.

For practical implementations one can think of G being a group of points on an elliptic curve such as brainpoolP320r1, including the standard generator from [12]. Here the size of q is 320 bits. Throughout this paper we will let $\mathcal{M}(K, \text{string})$ represent a key derivation function (KDF) that maps a string into secret key in $\mathrm{GF}(q)^*$. One can think of the KDF functions from [13] but also of a HMAC based function modulo q where HMAC is specified in [1]. For easy reference we simply refer to such keys as *KDF keys*.

We will also distinguish a secure hash function $\mathcal{I}(\text{string})$ that maps a string into the group G. In the context of an elliptic curve group $E(\mathrm{GF}(p))$ over a finite field $\mathrm{GF}(p)$ two approaches exist for such an embedding. A straightforward approach, cf. [14], is probabilistic. Here one uses a standard secure hash function to map the string to an element $x \in \mathrm{GF}(p)$ and verifies there exists a curve point with this x-coordinate. If this is not the case one varies the string in a deterministic fashion, e.g. by concatenating a string corresponding to an incrementing counter that starts with 1 and tries again. Each try has a fifty percent of success so eventually one will find a point on the curve. A deterministic polynomial-time algorithm to embed strings in elliptic curves can be found in [20].

For $S \in G$, $x, k \in \mathrm{GF}(q)$ and $y = g^x$ we let $\mathcal{EG}(S, y, k)$ denote the ElGamal encryption [6] of *plaintext* $S \in G$ with respect to the *public key* y and *private* key x. Technically, an ElGamal encryption consists of a pair of points in G of the form $(g^k, S \cdot y^k)$. The number k is called the *randomization exponent*. As can be easily verified, the decryption of an ElGamal encryption (A, B) is given by B/A^x. Throughout the paper we consider the generator g as the basis for all ElGamal encryptions which is why we do not explicitly include g as a parameter in $\mathcal{EG}(.)$. We consider g and in fact the specifications of the group G to be implicitly defined in the scheme specifications.

We remark that strictly speaking the public key y does not need to be included in the ElGamal encryption $\mathcal{EG}$ specification. Indeed, the party for which the encryption is intended does not require it as he already possesses it (or can calculate it from the private key x). There are two reasons why we let the public key be part of the ElGamal encryption. The first, and most important, reason is that it allows for easy randomization of ElGamal encryptions (see the third part of Proposition 2.1 below) which is a convenient tool to avoid linkability based on cryptograms in the e-ID infrastructure. The second reason is that including the public key facilitates easy look up of the required private key of the intended party. For these reasons we let the ElGamal encryption $\mathcal{EG}(S, y, k)$ have the form of the triple $(g^k, S \cdot y^k, y)$.

Below we have outlined the homomorphic properties of ElGamal encryption that are the building blocks of our scheme.

Proposition 2.1 *Let* $\mathcal{EG}(S, y, k) = (A, B; C)$ *be an ElGamal encryption of plaintext* S *under public key* $y = g^x$ *and let* z *be an element of* $\mathrm{GF}(q)^*$*. Then the following equalities hold:*

1. $(A^z, B^z, C) = \mathcal{EG}(S^z, y, k{\cdot}z)$,
2. $(A^z, B, C^{(z^{-1})}) = \mathcal{EG}(S, y^{(z^{-1})}, k{\cdot}z)$,
3. $(A \cdot g^z, B \cdot C^z, C) = \mathcal{EG}(S, y, k + z)$.

Proof: Easy verification. $\square$

From the first part of Proposition 2.1 it follows that anyone can perform an exponentiation on the plaintext S without knowing the value itself. Moreover, from the second part of Proposition 2.1 it follows that anyone can transform an ElGamal encryption under a public key y to another one of the form $y = y^z$ with related private key $x{\cdot}z$. Finally, the transformation in the last part of Proposition 2.1 is called the *randomization* of an ElGamal encryption. With this transformation anyone can transform an existing ElGamal encryption, only using the public information g and y, into a fresh one holding the same plaintext S but which is not linkable to the original one. This is due to the assumption that the Decision Diffie-Hellman problem is hard in G. See Appendices A and B. We note that one actually does not require the public key y for randomization but that any pair (g^r, y^r) for a non-zero r suffices. That is, to support randomization one could also define an ElGamal encryption as a quadruple $(g^k, S \cdot y^k, g^r, y^r)$ for some (random) k, r. Although this representation of an ElGamal encryption requires more storage (it consists of four points instead of three) it can have computational benefits. Most notably, it can be deployed to avoid the usage of an inverted exponent in the second part of Proposition 2.1 which can be beneficial, e.g. in HSM implementations.

3 Basic Scheme Key management

3.1 The role of HSMs

In this section we describe the cryptographic keys for the e-ID parties required for the envisioned polymorphic pseudonymization scheme. The e-ID parties possess secret keys that are security critical to the scheme. Although we have arranged that no party on its own can generate, transform or re-identify pseudonyms (cf. Section 5) we envision that these secret keys are generated and kept in secure environments, i.e. Hardware Security Modules or HSMs. These HSMs are regulated by the e-ID scheme and provided with the required cryptographic keys by the KMA. In such HSMs cryptographic keys are managed in non-exportable fashion. The cryptographic keys can be generated inside the secure environment or they can be transported (in encrypted form) from other secure environments, e.g. from the KMA. The software applications of the party can place calls to the HSM to use these keys to perform pre-defined cryptographic operations and to return the result.

Different with a regular HSM setup is that such calls will typically not consist of only one core cryptographic operation, e.g. as defined in the PKCS #11 standard [19], but of several of them. Only the end result is then returned to the requesting software applications. Compare Figure 2. Practically speaking this means that the HSMs need to have a secure execution environment in which a (web)service and an internal application can be implemented. The (web)service is the interface to the software application on the one hand and to the internal application to the other. The internal application places (e.g., PKCS #11 based) calls to the core HSM performing the atomic cryptographic calls. The HMS (web)services can only be executed after the software application has authenticated itself to the HSM, e.g. by proving possession of a certain (TLS client) key. We mention that this architecture is supported by most modern HSMs.

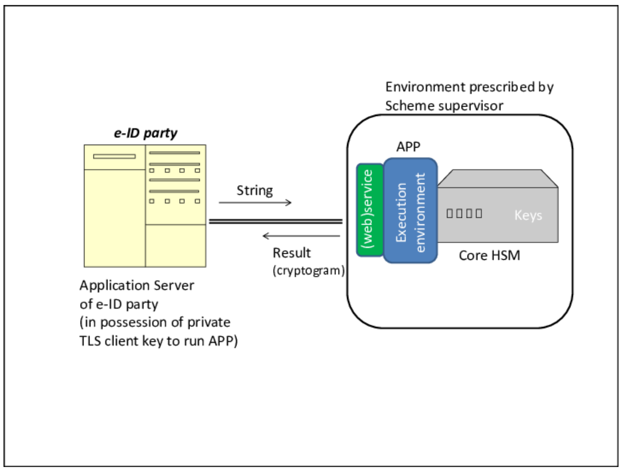
Environment prescribed by Scheme supervisor
e-ID party
APP
String
(web)service
Execution environment
Keys
Core HSM
Result (cryptogram)
Application Server of e-ID party (in possession of private TLS client key to run APP)

Figure 2. HSM setup

The HSMs we envision do not only provide preventive security but also detective security. The latter will also facilitate auditing and supervision of these parties. With respect to detective security we further envision that at HSM initialization a dedicated private response signing key is generated. Moreover, each HSM maintains an internal transaction counter similar to the EMV Application Transaction Counter in debit cards, cf. [7]. Any call to the HSM that uses secret information triggers an increment of an internal counter which is supplemented to the resulting cryptogram and the whole result is signed.

As we envision that the parties involved are required to maintain an administration of the signed cryptograms this facilitates easy control of the operations. Perhaps even more importantly, when one of the parties involved has a security breach this setup conveniently and reliably allows analysis of any rogue transactions by inspection of transaction counters. This HSM setup is currently not common practice or even specified in relevant (PKI) HSM standards such as FIPS 140-2 [15]. To further illustrate, in 2011 the security of the Dutch certification service provider DigiNotar was breached eventually leading to its bankruptcy [9]. One of the main issues in the forensic investigation was assessing if all rogue certificates could be accounted for. This was difficult to assess as the integrity of the logs was heavily based on software based systems. If the transaction counter setup had been used this investigation could have been simplified and its outcome might even have been different. We remark that a certification service provider cannot simply add such counters inside its HSM setup as this typically has impact on the HSMs FIPS 140-2 certification. However in our scheme we introduce functionality that is non-standard anyway, so adding transaction counters does probably not add for much additional complexity with respect to certification.

The above discussion leads to a private signing key and transaction counter for the KMA (KSK and KTC), for the CIPEI (CSK and CTC), for the pseudonym provider (PSK and PTC) and for the identity providers (ISK and ITC). See also Figure 3. To avoid creating obstacles in participation of the e-ID infrastructure we envision it is not mandatory for service providers to use an HSM. However as the usage of HSMs enhances the security of the service provider's operation we envision two types of service providers (and pseudonyms) corresponding to mandatory HSM usage or not.

3.2 KMA

At the start of its operation, the KMA sets up a Public Key Infrastructure to sign all public keys of the parties in the e-ID infrastructure including the operational keys of the KMA itself. Next, the KMA initialises its HSM with the generation of a private *KMA signing* key (KSK) inside the HSM and provides a signed copy of the public key to the e-ID parties. The HSM of the KMA also initialises a KMA Transaction Counter (KTC) and sets it to zero. See Section 3.1. The KMA then generates the following scheme wide parameters and secret keys which are managed in its HSM:

- *Secret Pseudonymization logarithm* $A \in_R \mathrm{GF}(q)^*$. This logarithm only resides in plaintext form in the HSM of the KMA and is implicitly used by the pseudonym provider in the user specific part of the pseudonyms.

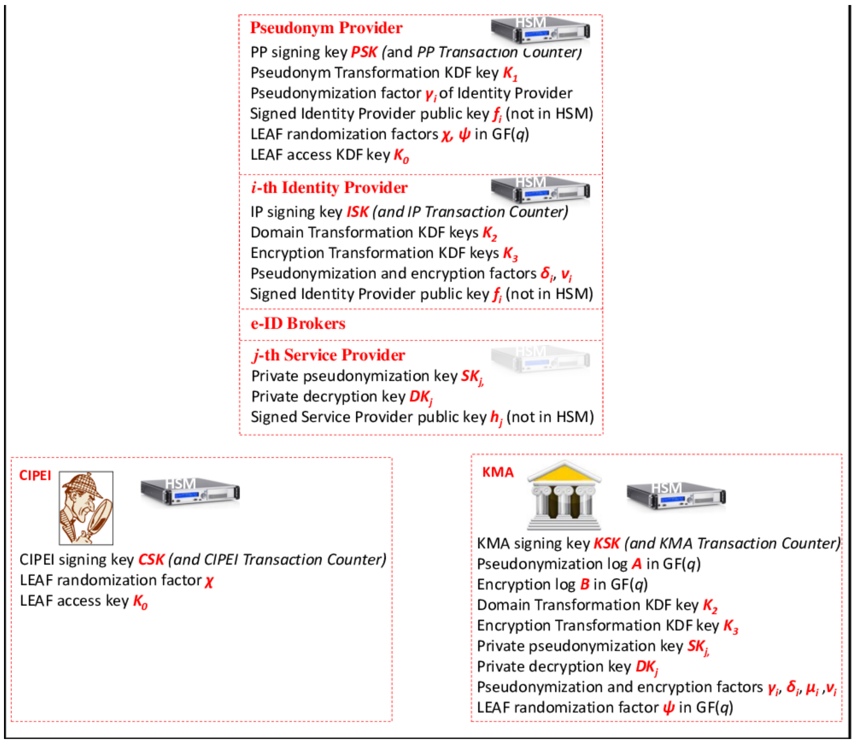
Pseudonym Provider
HSM
PP signing key PSK (and PP Transaction Counter)
Pseudonym Transformation KDF key K_1
Pseudonymization factor γ_i of Identity Provider
Signed Identity Provider public key f_i (not in HSM)
LEAF randomization factors χ, ψ in GF(q)
LEAF access KDF key K_0
i-th Identity Provider
HSM
IP signing key ISK (and IP Transaction Counter)
Domain Transformation KDF keys K_2
Encryption Transformation KDF keys K_3
Pseudonymization and encryption factors δ_i, ν_i
Signed Identity Provider public key f_i (not in HSM)
e-ID Brokers
j-th Service Provider
HSM
Private pseudonymization key SK_j
Private decryption key DK_j
Signed Service Provider public key h_j (not in HSM)
CIPEI
HSM
CIPEI signing key CSK (and CIPEI Transaction Counter)
LEAF randomization factor χ
LEAF access key K_0
KMA
HSM
KMA signing key KSK (and KMA Transaction Counter)
Pseudonymization log A in GF(q)
Encryption log B in GF(q)
Domain Transformation KDF key K_2
Encryption Transformation KDF key K_3
Private pseudonymization key SK_j
Private decryption key DK_j
Pseudonymization and encryption factors γ_i, δ_i, μ_i, ν_i
LEAF randomization factor ψ in GF(q)

Figure 3. The e-ID cryptographic key infrastructure

- *Secret encryption logarithm* $B \in_R \mathrm{GF}(q)^*$. This logarithm only resides in plaintext form in the HSM of the KMA and is implicitly used by the identity providers to allow the service provider to decrypt the encrypted pseudonyms.
- *Domain Transformation* key K_2. This is a KDF key that is securely transported from the HSM of the KMA to the HSMs of the identity providers. This key is used by the identity providers to transform their polymorphic pseudonyms into the domain of the service providers.
- *Encryption Transformation* key K_3. This is also a KDF key that is securely transported from the HSM of the KMA to the HSMs of the identity providers. This key is used by the identity providers to transform the encryption of the polymorphic pseudonyms into an encryption that the service provider has a private decryption key for.

We remark that KDF keys K_0, K_1 also exist in the scheme; these are generated by the pseudonym provider and not shared with the KMA. See Section 3.3.

For the i-th identity provider IP_i $(1 \leq i \leq m)$ the KMA generates the following factors which are managed in secure environments, i.e. Hardware Security Modules:

- Pseudonimization factors $\gamma_i \in_R \mathrm{GF}(q)^*$ and $\delta_i = A/\gamma_i$ or in other words $A = \gamma_i{\cdot}\delta_i$. The factor γ_i is securely transported to the HSM of the pseudonym provider, the factor δ_i is securely transported to the HSM of the i-th identity provider.
- Encryption factors $\mu_i \in_R \mathrm{GF}(q)^*$ and $\nu_i = \mu_i/B$ or in other words $B = \mu_i/\nu_i$. The factor ν_i is securely transported to the HSM of the i-th identity provider.

These factors ensure that polymorphic pseudonyms are non-interoperable among identity providers (see Section 5). The form of the factors is also chosen to facilitate easy software implementation. For the i-th identity provider the KMA then forms the public key $f_i = g^{\mu_i}$, signs it and sends a signed copy to both the i-th identity provider and the pseudonym provider. Note that the identity provider (and in fact no e-ID party) is not provided with the corresponding private key μ_i.

For the j-th service provider the KMA generates a private pseudonymization key $SK_j \in_R \mathrm{GF}(q)$ and a private ElGamal decryption key DK_j that takes the form:

$$DK_j = \frac{B}{\mathcal{M}(K_3, \mathit{SP\text{-}id}_j \,|\, \mathcal{DI})}. \tag{1}$$

Here $\mathcal{DI}$ represents distinguishing information, e.g. a serial number, allowing to periodically change the service provider decryption key DK_j. Both SK_j and DK_j are securely transported from the KMA HSM to the service provider or its HSM if the service provider is required to have one. Finally, each service provider is given its public key $h_j = g^{DK_j}$ by the KMA in a signed version.

Finally, to facilitate legal access, the KMA generates in its HSM a *LEAF randomization factor* $\psi \in_R \mathrm{GF}(q)$ this is also securely transported to the HSM of the pseudonym provider.

3.3 Pseudonym Provider (PP)

At the start of its operation, the KMA initialises its HSM with the generation of a private *PP signing* (PSK) key inside the HSM and the reliable transport of the corresponding (signed) public key to the e-ID parties. The HSM of the PP also initialises a PP Transaction Counter (PTC) and sets it to zero. See the beginning of Section 3.

The PP then securely generates a *User Pseudonymization* KDF key K_1 in its HSM on which the user specific part of the pseudonym is based. As user pseudonymization is not required by the KMA the key K_1 is not shared with the KMA or in fact any other party. That is, the PP is the only party in the e-ID infrastructure that is able to fill in the user specific part of the pseudonyms.

To facilitate legal access, the PP securely generates a *LEAF access key* KDF key K_0 and a *LEAF randomization factor* $\chi \in_R \mathrm{GF}(q)$ at the start of its operation. The LEAF access key and the LEAF randomization factor is securely exchanged to the HSM of the CIPEI.

Next the PP is provided with the public key f_i of the i-th identity provider ($1 \leq i \leq m$) signed by the KMA. Also the pseudonimization factor γ_i and the LEAF randomization factor ψ are securely transported from the HSM of the KMA to the HSM of the PP.

3.4 Identity Provider (IP)

At the start of its operation, each IP initialises its HSM with the generation of a private *IP signing* (ISK) key inside the HSM and the reliable transport of the corresponding (signed) public key to the e-ID parties. The HSM of the i-th identity provider also initialises an IP Transaction Counter (ITC) and sets it to zero. See the beginning of Section 3.

The HSM of each identity provider IP_i ($1 \leq i \leq m$) is securely provided by the HSM of the KMA:

- The Domain Transformation KDF key K_2.
- The Encryption Transformation KDF key K_3.
- The pseudonymization factor δ_i,
- The encryption factor ν_i.

3.5 Service Provider (SP)

Each j-th service provider with identifier $SP\text{-}id_j$ ($1 \leq j \leq n$) is provided by the KMA with a private pseudonymization key $SK_j \in_R \mathrm{GF}(q)$ and a private decryption key DK_j. See Equation (1). These private keys need to be protected by the service provider, depending on the service provider type these keys are protected in an HSM.

3.6 Central Information Point e-ID Investigations(CIPEI)

At the start of its operation, the CIPEI initialises its HSM with the generation of a private *CIPEI response signing* (CRS) key inside the HSM and the reliable transport of the corresponding (signed) public key to the e-ID parties. The HSM of the CIPEI also initialises a CIPEI Transaction Counter (CTC) and sets it to zero.

As we explained at the beginning of Section 3 any request to the HSM that uses secret information in the HSM triggers an increment of the CTC which is supplemented to the resulting cryptogram and the whole result is signed. The KMA is only allowed to respond to signed (and numbered) messages and a similar setup is applicable to the KMA (cf. Section 3.2). The CIPEI is also required to maintain an administration of the signed cryptograms allowing an auditor to easily validate if all such requests meet the legal requirements, e.g. if corresponding court orders exist. Finally the CIPEI securely imports the *LEAF access key* K_0 and the LEAF randomization factor χ in its HSM which are securely transported from the HSM of the PP.

4 Scheme Description

The pseudonym of user $U\text{-}id$ at the j-th service provider with identifier $SP\text{-}id_j$ takes the form:

$$P_{U,SP\text{-}id_j} = \mathcal{I}(U\text{-}id)^{A \cdot \mathcal{M}(K_1, U\text{-}id) \cdot \mathcal{M}(K_2, SP\text{-}id_j) \cdot SK_j}. \tag{2}$$

The pseudonym is an exponentiation of $\mathcal{I}(U\text{-}id)$, the identity of the user embedded in the group G, to a secret power which is the product of four parts applied in the following order:

- a user specific part, $\mathcal{M}(K_1, U\text{-}id)$) provided by the pseudonym provider,
- factor A provided in concert by the pseudonym provider and the identity provider,
- a service provider specific part, $\mathcal{M}(K_2, SP\text{-}id_j)$) provided by the identity provider,
- another service provider specific part, SK_j, provided by the j-th service provider itself.

For later reference we remark that the following intermediate pseudonym forms P_U and P_{U*} appear in the pseudonymization process

$$\begin{aligned} P_{U-} &= \mathcal{I}(U\text{-}id)^{\mathcal{M}(K_1, U\text{-}id)} \\ P_U &= \mathcal{I}(U\text{-}id)^{A \cdot \mathcal{M}(K_1, U\text{-}id)} \\ P_{U*} &= \mathcal{I}(U\text{-}id)^{A \cdot \mathcal{M}(K_1, U\text{-}id) \cdot \mathcal{M}(K_2, SP\text{-}id_j)} \end{aligned} \tag{3}$$

The pseudonym $P_{U,SP\text{-}id_j}$ is provided by an identity provider in ElGamal encrypted form under the public key of the service provider which takes the form

$$g^{DK_j} = g^{\frac{B}{\mathcal{M}(K_3, SP\text{-}id_j \mid \mathcal{DI})}}.$$

Here $\mathcal{DI}$ represents distinguishing information, e.g. a serial number, allowing to periodically change the service provider decryption key DK_j.

In the steps below we describe how the scheme provides the user pseudonym to the service provider in a series of computations and exchanges between the parties involved. To this end we first describe the scheme in a high level fashion and then we refer to cryptographic building blocks we describe in subsections below. The scheme involves several redirections of the user; we assume that the scheme has some kind of secure session concept which allows the secure linkage of these redirections and secure exchange of information. More specifically, the redirections and information exchanges between the parties involved could be based on SAML, cf. [18].

The scheme starts with a user with an e-ID wide unique identifier *U-id* that wants to authenticate himself to the j-th service provider identified by $\textit{SP-id}_j$. The user is shown several identity providers that can facilitate this. The user has an existing relation with at least one of these identity providers. That is, the user already possesses an authentication token for this identity provider. Such a token could for instance be a smartcard, a hardware token, a One-Time-Password mechanism based on SMS, or even something as prosaic as a user-id/password. The technical token quality, the process of registration, cf. [21], and the communication thereof to the service provider is outside the scope of this paper. As stated in the introduction, we only assume that the registration process is such that all identity providers in the scheme are able to identify a user with a unique string *U-id*.

The user selects an identity provider and is redirected to that by the service provider. Next the user authenticates himself to the identity provider. The rules and regulations of the scheme will provide for guidelines when authentication is not successful, e.g. by informing the service provider or blocking the token used. This is outside the scope of this paper. If authentication is successful we assume that the identity provider is able to deduce the unique *U-id* of the user from its administration. The identity provider then connects to the pseudonym provider and requests the Polymorphic Pseudonym corresponding with *U-id*. This effectively is the registration of the user to the e-ID infrastructure. Some kind of user consent is requested here, but this is outside the scope of this paper.

We also note that the identity provider could already possess the Polymorphic Pseudonym corresponding with the *U-id* if this registration was done on an earlier occasion. If this is not the case then the pseudonym provider registers the user into the e-ID infrastructure using the building block Polymorphic Pseudonym and LEAF Provisioning by PP, see Section 4.1. With this building block the pseudonym provider uses the identity provider specific parameters to transform the *U-id* of the user to a Polymorphic Pseudonym and a Law Enforcement Access Field (LEAF). The Polymorphic Pseudonym is provided to the identity provider and is specific for both the user and the identity provider. The LEAF is provided to the Central Information Point e-ID Investigations(CIPEI) to facilitate requests from law enforcement agencies. See Section 4.4.

On receipt of the Polymorphic Pseudonym the identity provider transforms this to an Encrypted Pseudonym for the requesting service provider using the building block Encrypted Pseudonym Provisioning by IP (Section 4.2). This holds the pseudonym of the user encrypted under a public key of which the service provider has the corresponding private key.

The Encrypted Pseudonym is then sent by the identity provider to the service provider which then decrypts this using the building block Encrypted Pseudonym Provisioning by IP (Section 4.3).

4.1 Polymorphic Pseudonym and LEAF Provisioning by PP

The starting point of this building block is that the pseudonym provider has received the $U\text{-}id$ from the i-th identity provider. For this the pseudonym provider application calculates the corresponding embedding $\mathcal{I}(U\text{-}id) \in G$. We remark that we let this embedding be calculated by the pseudonym provider application instead of its HSM for practical reasons; this kind of calculation is not common in HSMs. Next the pseudonym provider looks up the public key f_i of the i-th identity provider. The pseudonym provider software application then sends $\mathcal{I}(U\text{-}id)$, $U\text{-}id$ and f_i to its HSM together with a reference to the i-th identity provider and requests a Polymorphic Pseudonym and corresponding Law Enforcement Access Field (LEAF).

To this end, the HSM first increments its counter PTC and forms the Polymorphic Pseudonym and corresponding LEAF which takes the form:

$$\{E_1, E_2, E_3, PTC\}_{SIG}; \{L_1, L_2\}_{SIG}, \{L_3, L_4\}_{SIG}. \tag{4}$$

This consists of three signed structures the first one is the Polymorphic Pseudonym and the second and the third the LEAF each corresponding to the two CIPEI requests introduced in the introduction.

The Polymorphic Pseudonym consists of four elements the last of which is the PP Transaction Counter. The first three elements form an ElGamal encryption that is constructed as follows. The HSM first calculates the value $\gamma_i \cdot \mathcal{M}(K_1, U\text{-}id)$ based on the provided $U\text{-}id$ and the factor γ_i and KDF key K_1 inside its secure memory. This value is used to form:

$$\mathcal{I}(U\text{-}id)^{\gamma_i \cdot \mathcal{M}(K_1, U\text{-}id)} = P_U^{(\delta_i^{-1})},$$

where the equality follows from equation (3) and the fact that $A = \gamma_i \cdot \delta_i$. The HSM then generates $k \in_R \mathrm{GF}(q)$ and forms the ElGamal encryption

$$\mathcal{EG}(P_U^{(\delta_i^{-1})}, f_i, k) = (E_1, E_2, E_3). \tag{5}$$

This concludes the construction of the Polymorphic Pseudonym.

A LEAF consists of two signed structures. The HSM of the PP first calculates three intermediate values O_1, O_2, O_3:

$$\begin{aligned} O_1 &= \mathcal{I}(\mathit{U\text{-}id})^{\mathcal{M}(K_1,\,\mathit{U\text{-}id})\cdot\psi\cdot\chi} = P_{U-}^{\cdot\psi\cdot\chi} \\ O_2 &= \mathcal{H}(O_1), \\ O_3 &= \mathcal{M}(K_0,\,\mathit{U\text{-}id}). \end{aligned} \tag{6}$$

Compare formula (3). Here $\mathcal{H}(M)$ represents a secure hash of a message M, e.g. [17]. Next, the LEAF elements are calculated as:

$$\begin{aligned} L_1 &= \mathcal{H}(O_2), \\ L_2 &= \mathcal{E}(O_2,\,\mathit{U\text{-}id}), \\ L_3 &= \mathcal{H}(O_3)), \\ L_4 &= \mathcal{E}(O_3, O_1). \end{aligned} \tag{7}$$

Here $\mathcal{E}(K, M)$ represents a symmetric encryption of a message M under a key K, e.g. [16]. The elements L_1, L_2 and L_3, L_4 are each combined and signed resulting in the two LEAF structures. Then the HSM returns the polymorphic pseudonym and the LEAF in response to the request of the pseudonym provider. Finally, the pseudonym provider returns the polymorphic pseudonym to the requesting identity provider and the corresponding LEAF to the CIPEI.

4.2 Encrypted Pseudonym Provisioning by IP

The identity provider wants to provide the j-th service provider (with reference $\mathit{SP\text{-}id}_j$) an encrypted pseudonym of the user that has just successfully authenticated itself and wants to authenticate himself to the service provider. The identity provider first looks up the user's Polymorphic Pseudonym of the form $\{E_1, E_2, E_3, PTC\}_{SIG}$ from the pseudonym provider as part of an earlier e-ID registration. Next the identity provider software application sends this Polymorphic Pseudonym and reference $\mathit{SP\text{-}id}_j$ to its HSM and requests an encrypted pseudonym.

The HSM of the identity provider first checks the signature on the Polymorphic Pseudonym. Recall from Expression (5) that E_1, E_2, E_3 takes the form

$$(E_1, E_2, E_3) = \mathcal{EG}(P_U^{(\delta_i^{-1})}, f_i, k), \tag{8}$$

for some random $k \in \mathrm{GF}(q)$ placed by the HSM of the pseudonym provider.

Then the HSM uses its KDF keys K_2, K_3 and its encryption factors δ_i, ν_i to calculate $H_1 = \delta_i \cdot \mathcal{M}(K_2, \mathit{SP\text{-}id}_j)$ and $H_2 = \nu_i \cdot \mathcal{M}(K_3, \mathit{SP\text{-}id}_j \,|\, \mathcal{DI})$. Here $\mathcal{DI}$ represents distinguishing information, e.g. a serial number, allowing to periodically change the service provider decryption keys. Typically, the identifier of the service provider and the distinguishing information will be retrieved by the identity provider from a digital certificate belonging to the service provider.

Next the HSM generates a $l \in_R \mathrm{GF}(q)$ and calculates the encrypted pseudonym as:

$$\begin{aligned}&\{E_1^{H_1 \cdot H_2} \cdot g^l, E_2^{H_1 + l \cdot H_2^{-1}}, E_3^{(H_2^{-1})}, ITC\}_{SIG}\\ =\ &\{\mathcal{EG}(P_{U*}, h_j, \tilde{k}), ITC\}_{SIG}\end{aligned} \tag{9}$$

where $\tilde{k} = k \cdot H_1 \cdot H_2 + l$

To explain the working of this we will calculate this expression in three steps. In the first step the HSM incorporates the identifier $\mathit{SP}\text{-}\mathit{id}_j$ of the j-th service provider into the intermediate pseudonym as follows. To this end, the HSM transforms (E_1, E_2, E_3) into

$$\begin{aligned}(F_1, F_2, F_3) &= (E_1^{H_1}, E_2^{H_1}, E_3)\\ &= \mathcal{EG}(P_U^{(\delta_i^{-1} \cdot H_1)}, f_i, k \cdot H_1),\\ &= \mathcal{EG}(P_U^{(\delta_i^{-1} \cdot \delta_i \cdot \mathcal{M}(K_2, \mathit{SP}\text{-}\mathit{id}_j))}, f_i, k \cdot H_1),\\ &= \mathcal{EG}(P_U^{\mathcal{M}(K_2, \mathit{SP}\text{-}\mathit{id}_j)}, f_i, k \cdot H_1),\\ &= \mathcal{EG}(P_{U*}, f_i, k \cdot H_1)\end{aligned}$$

The second equality follows from the first part of Proposition 2.1, the third one from the definition of H_1 above and the fourth one from the definition of γ_i, δ_i in Section 3.2. The last equality follows from the definition of P_{U*} in the beginning of Section 3. This concludes the first step: we have ended up at an ElGamal encrypted pseudonym under the public key f_i of which no e-ID party has the private part.

In the second step the HSM transforms this ElGamal encryption to one related to the private key of the service provider. To this end, the HSM transforms (F_1, F_2, F_3) into

$$\begin{aligned}(G_1, G_2, G_3) &= (F_1^{H_2}, F_2, F_3^{(H_2^{-1})})\\ &= \mathcal{EG}(P_{U*}, f_i^{H_2^{-1}}, k \cdot H_1 \cdot H_2)\\ &= \mathcal{EG}(P_{U*}, f_i^{\frac{1}{\nu_i \cdot \mathcal{M}(K_3, \mathit{SP}\text{-}\mathit{id}_j \mid \mathcal{DI})}}, k \cdot H_1 \cdot H_2)\\ &= \mathcal{EG}(P_{U*}, g^{\frac{\mu_i}{\nu_i \cdot \mathcal{M}(K_3, \mathit{SP}\text{-}\mathit{id}_j \mid \mathcal{DI})}}, k \cdot H_1 \cdot H_2)\\ &= \mathcal{EG}(P_{U*}, g^{\frac{B}{\mathcal{M}(K_3, \mathit{SP}\text{-}\mathit{id}_j \mid \mathcal{DI})}}, k \cdot H_1 \cdot H_2)\\ &= \mathcal{EG}(P_{U*}, h_j, k \cdot H_1 \cdot H_2)\end{aligned}$$

The second equality follows from the second part of Proposition 2.1, the third one from the definition of H_2 above and the fourth one from the definition of f_i in Section 3.2. The fifth equality follows from the definition of μ_i, ν_i in Section 3.2 and the last one from the definition of h_j in Section 3.2. This concludes the second step: we have ended up at an ElGamal encrypted pseudonym under the public key h_j of the j-th service provider.

In the third and final step the HSM randomizes the ElGamal encryption from the second step, cf. the third part of Proposition 2.1. This will make various encrypted pseudonyms for the user at the same service provider indistinguishable. To this end, the HSM generates a $l \in_R \mathrm{GF}(q)$ and transforms (H_1, H_2, H_3) into

$$(H_1, H_2, H_3) = (G_1{\cdot}g^l, G_2{\cdot}G_3^l, G_3).$$

One can easily verify that the three steps explained above lead to the expression in Equation (9).

4.3 Pseudonym Provisioning by SP

The starting point of this building block is that the j-th service provider has received an encrypted pseudonym from an identity provider. According to Equation (9) this takes the form

$$(I_1, I_2, I_3) = \mathcal{EG}(P_{U*}, h_j, \tilde{k})$$

for some $\tilde{k} \in \mathrm{GF}(q)$. The service provider then first uses its private ElGamal decryption key DK_j to extract P_{U*} as

$$P_{U*} = I_2 / I_1^{DK_j}.$$

Then service provider uses its private pseudonymization key SK_j to form the user pseudonym in its final form (cf. formula (3)):

$$P_{U,SP\text{-}id_j} = P_{U*}^{SK_j}.$$

4.4 CIPEI handling of law enforcement agency request

The starting point of this building block is that the CIPEI receives a lawful request of a law enforcement agency. Two kinds of request can be handled.

Request Type 1 (“de-pseudonimization”) A pseudonym and a reference to the related service provider domain is given by the law enforcement agency; the identity of the person behind the pseudonym is requested.

Request Type 2 (“pseudonimization on request”) The *U-id* of a user is given by the law enforcement agency together with a reference to a service provider domain; the pseudonym of the person in the service provider domain is requested.

Request Type 1 is handled as follows. The CIPEI instructs its HSM to generate a KMA request and feeds it the received pseudonym and the reference $SP\text{-}id_j$ for the j-th service provider. The HSM then uses its LEAF randomization factor $\chi \in \mathrm{GF}(q)$ and forms (after increasing its counter CTC):

$$\{P^{\chi}_{U,SP\text{-}id_j}, SP\text{-}id_j, CTC\}_{SIG} = \{M_1, M_2, M_3\}_{SIG}. \tag{10}$$

This expression is sent to the KMA, requesting the LEAF opening value (O_2 in Formula (6)) related to the pseudonym. The HSM of the KMA then uses the LEAF factor ψ, pseudonymization logarithm A, the KDF key K_2 and the key SK_j, to transform expression (10) into

$$\begin{aligned} O_1 &= M_1^{\psi\cdot(A\cdot\mathcal{M}(K_2,SP\text{-}id_j)\cdot SK_j)^{-1}} = P_{U,SP\text{-}id_j}^{\chi\cdot\psi\cdot(A\cdot\mathcal{M}(K_2,SP\text{-}id_j)\cdot SK_j)^{-1}} = P_{U-}^{\psi\cdot\chi} \\ O_2 &= \mathcal{H}(O_1). \end{aligned} \tag{11}$$

Compare formula (3). The KMA now returns O_2 to the CIPEI in a signed and numbered form. The CIPEI then determines the corresponding LEAF identifier $L_1 = \mathcal{H}(O_2)$ and looks up the corresponding LEAF structure $\{L_1, L_2\}_{SIG}$ in its administration. Next the CIPIE uses O_2 to decrypt the L_2 part of the leaf to determine *U-id*. See Formula (7). This is then returned to the requesting law enforcement agency. In case of disputes, the CIPEI can also provide all related information.

Request type 2 is handled as follows. The CIPEI first sends *U-id* to its HSM and requests the value O_3 in formula (6) of the user. The HSM then uses the KDF key K_0 to calculate $O_3 = \mathcal{M}(K_0, U\text{-}id)$ and returns this as a signed and numbered message. The CIPEI then determines the corresponding LEAF identifier $L_3 = \mathcal{H}(O_3)$ and looks up the corresponding LEAF structure $\{L_3, L_4\}_{SIG}$ in its administration. The CIPEI then decrypts value L_4 using O_3 and determines the group element $O_1 = \mathcal{I}(U\text{-}id)$ corresponding to the user. The CIPEI then sends this group element to its HSM together with the reference to the service provider $SP\text{-}id_j$ domain the pseudonym of the user is requested for. The HSM then returns a signed numbered request including O_1 that is sent to the KMA by the CIPEI. Additionally one could let O_3 and the LEAF structure $\{L_3, L_4\}_{SIG}$ be part of the request enabling the KMA to validate the request. The HSM of the KMA exponentiates the value O_1 to the power

$$A\cdot\mathcal{M}(K_2, SP\text{-}id_j)\cdot SK_j \cdot \psi^{-1}$$

which leads to $P_{U,SP\text{-}id_j}^{\chi}$. This is then numbered and signed by the KMA HSM and sent to CIPEI by the KMA. The CIPEI then sends this to its HSM which removes the factor χ (arriving at $P_{U,SP\text{-}id_j}$ as desired) and returns this as part of a signed numbered message. The message is then sent to the requesting law enforcement agency.

5 Security arguments

Below we informally compare the proposed scheme against the main pseudonymization security requirements from the introduction. Further formalizations of the PPCA security properties are placed in Appendices A and B.

1. *A pseudonym does not allow the identification of the user through cryptographic means.* In this mis-use case an adversary has a pseudonym $P_{U,SP\text{-}id_j}$ for a service provider with identifier $SP\text{-}id_j$ and wants to find out to which

user *U-id* it belongs. As follows from formula (2) a user pseudonym is a function based on secret one way values (HMACs) from three e-ID parties, the PP, an IP and the service provider the pseudonym is meant for. This means that the only attack possible is based on brute-forcing the *U-id* for which the cooperation is required of all three parties involved. Regardless of the cryptographic techniques, a colluding identity provider and service provider could perform a brute-force on the *U-id* of the user, i.e. by asking polymorphic pseudonyms for all users at the pseudonym provider, transform them to encrypted pseudonyms and letting the service provider validate if it matches $P_{U,SP\text{-}id_j}$. This will be detectable by the HSM counters, however. It is an implicit assumption for the scheme that this risk can be made acceptable.

2. *Pseudonym domains of the service provider are not cryptographically linkable. That is, at different service providers the user gets different pseudonyms which are not inter-linkable through cryptographic means.* In this mis-use case an adversary has two pseudonyms at two different service provider domains and wants to decide whether these correspond to the same person. It follows from formula (2) that for this one needs access to the secret keys of an identity provider and the two service providers. That is, these three parties would need to collude. Note: even if this is the case, the HSM of the identity provider will not support the requested operation. Regardless of the cryptographic techniques, a colluding identity provider and two service providers could perform a brute-force on the *U-id* of the users involved and assess if the two pseudonyms correspond to the same user. This will be detectable by the HSM counters. It is an implicit assumption for the scheme that this risk can be made acceptable.
3. *Possession of many pseudonyms and the corresponding user identities does not allow the creation of an additional user pseudonym.* In this mis-use case we assume that an adversary can acquire all pseudonyms and corresponding identities at will and his task is to create an extra user pseudonym, i.e. one not yet asked. This effectively reduces to the task of leaving out one of the pseudonyms in

$$\{P_{U_1,SP\text{-}id_1}, P_{U_1,SP\text{-}id_2}, P_{U_2,SP\text{-}id_1}, P_{U_2,SP\text{-}id_2}\},$$

for certain user ids U_1, U_2 and service provider ids $SP\text{-}id_1, SP\text{-}id_2$ and construct the missing pseudonym from the other three. As the base of the pseudonym is formed by the embedding $\mathcal{I}(.)$ of the user identity *U-id* this amounts to a Decision Diffie-Hellman problem in G. Unless an identity provider and the two service providers $SP\text{-}id_1, SP\text{-}id_2$ collude and use their secret keys this problem is assumed to be intractable. In this case the colluding parties can transform pseudonyms between the two service provider domains. However, this is a risk we have encountered already. It is an implicit assumption for the scheme that this risk can be made acceptable.
4. *Encrypted pseudonyms are indistinguishable for the e-ID parties, most notably for the brokers. That is, the encrypted pseudonyms for the same user*

at the same service provider should not be linkable by other parties than the intended service provider and possibly the KMA and CIPEI. This property is due to the randomization property of ElGamal encryption and the assumed hardness of the Decision Diffie-Hellman problem in G. See the remarks following Proposition 2.1.

5. *Although the identity provider is performing the transformation from polymorphic pseudonym to encrypted pseudonym he is not able to access the pseudonym it helps to form.* Regardless of the security provided by the HSM, it follows from formula (2) that a user pseudonym also includes the usage of the key SK_j of the j-the service provider which the service provider does not possess.
6. *An identity provider cannot use the polymorphic pseudonyms of another identity provider to extract encrypted pseudonyms from.* The intermediate pseudonym for another identity provider is encrypted with a (different) pseudonym factor δ, the identity provider does not possess. Moreover, it is encrypted with a public key that the identity provider cannot transform to a service provider public key as it does not have the appropriate encryption factor ν.

6 Extensions

In this section we sketch some extensions to the basic polymorphic pseudonym system:

- Diversified Pseudonyms
- Polymorphic Pseudonym Card Application (PPCA)
- e-ID Token Status Services
- Polymorphic Attributes

6.1 Diversified Pseudonyms

By using additional information (e.g. sequence numbers) in the identifiers $U\text{-}id$ and $SP\text{-}id_j$ one can create several pseudonym domains at one service provider. This also allows for all kinds of pseudonyms, including temporary pseudonyms.

6.2 Polymorphic Pseudonym Card Application (PPCA)

In the basic setup of our scheme, the identity provider knows the identity of the user and it knows to which service provider the user is going. There might be service providers where the sole fact of being a client is privacy sensitive. Our scheme also simply supports the notion of a polymorphic pseudonymity card application (PPCA). The idea is that an identity provider places polymorphic pseudonyms for a user in a simple smartcard application. These polymorphic pseudonyms are a separate class (with their own randomization and encryption factors) to prevent replay attacks, see below.

The fundamental security property of the PPCA application is that it allows the user to present a polymorphic pseudonym without the identity provider knowing which card is used or even to be able to distinguish the various PPCA cards used. The basis for this property is that the PPCA application randomizes the polymorphic pseudonym before sending it out of the card (cf. the remarks following Proposition 2.1). In Proposition A.1 in Appendix A we show that the ability to distinguish issued PPCA cards is as hard as breaking the Decision Diffie-Hellman problem with respect to g, which is considered to be intractable in our setting.

To prevent likability the PPCA outputted polymorphic pseudonyms are not provided with a counter and signature. That is, they only consist of the triple $\{E_1, E_2, E_3\}$ as indicated in Formula (5). To further reduce communicational overhead, the public key of the identity provider, i.e. E_3, can be omitted in the output. The identity provider processes the polymorphic pseudonym as before but wipes them after use to prevent possible replay attacks, see below. In fact, the handling of the PPCA application could take directly take place in the HSM that generates the encrypted pseudonyms. The PPCA application could be placed on the chip next to other applications such as a regular eID application, a Qualified Electronic Signature (QES) application, an ISO compliant driving licence (IDL) ([8]), an ICAO compliant electronic passport ([10]). In fact, see below, the PPCA application resembles an electronic passport: the polymorphic pseudonyms can be read in a similar way as fingerprints are read from an European electronic passport.

Two important potential risk types exist in the context of PPCA.

Replay of PPCA polymorphic pseudonyms: an adversary is able to send a polymorphic pseudonym of an existing user to the identity provider and takes over the e-ID identity of that user.

Forging of PPCA polymorphic pseudonyms: an adversary is able to send a random polymorphic pseudonym, i.e. not relating to an existing user, and thus creates a fake e-ID identity.

To address the first risk, access to the randomized polymorphic pseudonym in PPCA should be restricted to the identity provider. This can for instance be arranged by using the technique of Card Verifiable Certificates (CVC) in PPCA. In this way, access to the randomized polymorphic pseudonym is only possible for a party having a certificate from a certain public key infrastructure and the related private key. This is the technique (extended access control, [2]) used to protect fingerprint templates in European electronic passports. See also [10]. The second risk can be addressed inside the e-ID infrastructure by introducing an token status service inside the e-ID infrastructure, see Section 6.3. Additionally, one could also use the setup of the German e-ID scheme [2] to authenticate the card of the user without making it linkable. Currently the German e-ID scheme supports two such mechanisms called Chip Authentication version 2/Restricted Identification [2], [3] and Chip Authentication version 3/Pseudonymous Signatures [4]. The latter is still in draft status at the writing of this document.

In the German e-ID setup, pseudonyms are read by the service provider directly from the e-ID card. The service provider needs a specific CVC certificate (and accompanying private key) for that also know as a *terminal certificate.* With Restricted Identification batches of cards share a private authentication key (Chip Authentication key) allowing the identity provider to validate the authenticity of the card and to setup a secure channel to the identity provider. The batch size is chosen such that the card is not linkable by its Chip Authentication key. A pseudonym is generated by the card as as secure hash value based on a secret key on the card and a public key of the service provider inside its terminal certificate. It is sent by the card over the secure channel. This setup is susceptible to both risk types mentioned above. Indeed, these pseudonyms are stored at the service providers and in principle can be stolen from there by an adversary. This implies that if the adversary also extracts the shared Chip Authentication key from a card he will be able to impose as any of the users corresponding to the stolen pseudonyms. With the Chip Authentication key the adversary will also be able to send fake pseudonyms to service providers.

With Pseudonymous Signatures pseudonyms are no longer sent to the service provider in hashed form. This allows the card to "prove" possession of the pseudonym through a Pseudonymous Signature effectively over a public key in the terminal certificate. This will mitigate the first risk. However the secret key used for the Pseudonymous Signature is still shared among cards, albeit in in pairs of cards. That is, compromise of the keys on two cards will allow extraction of the pseudonym signature key. If this is the case then the second risk type mentioned above might become manifest. We note that [5] indicates that in this case white lists will be published indicating the real pseudonyms and differentiating them from the fake ones. We finally note that the fact that pseudonyms are no longer sent to the service provider in a hashed form (as is the case in Restricted Identification) also introduces a new type of potential risk. Indeed, this setup makes it at least cryptographically possible for the party issuing the terminal certificates ("DVCA", see [4]) to perform pseudonym conversions on its own. This places extra weight at the cryptographic key protection at the DVCA.

One can use either techniques from the German e-ID to prove to the identity provider it is connected to an authentic card without providing linking information. To this end, in the context of Chip Authentication version 2 this would mean having a shared chip authentication key in a batch of cards and sending a randomized polymorphic pseudonym instead of the German pseudonym. In the context of Chip Authentication version 2 one would sign the randomized polymorphic pseudonym with a pseudonymous signature over the polymorphic pseudonym instead of over the service provider public key.

One can argue that the drawbacks of Chip Authentication version 2 are less relevant in the context of PPCA than in the German e-ID context. In the PPCA context compromise of the Chip Authentication key will allow the adversary to establish a secure tunnel with the identity provider. However, there will be no place where the adversary can harvest user polymorphic pseudonyms as we assumed they are wiped after use at the identity provider. In the context of PPCA it seems that both Geman e-ID chip authentication techniques are more or less equivalently secure. And as already stated both chip authentication techniques can be considered as only additional as the risk of fake identities can be mitigated in the e-ID infrastructure itself as well.

PPCA facilitates that a user can harvest attributes and authorizations in the "named world", i.e. under his formal identity, have them placed under a pseudonym at an attribute or authorization provider and then use them in the "pseudonymous world" through PPCA. This allows for a setup combining the best of both worlds. In the next bullet we decribe how we can also support revocation for PPCA.

We end this section with some results on practical PPCA implementation. The randomization operation takes as input an ElGamal encryption $(A, B, C) = (g^k, S \cdot y^k, y)$, generates an $r \in_R \mathrm{GF}(q)$ and then produces $(A \cdot g^r, B \cdot y^r)$. This takes two exponentiations and two multiplications. When implementing this in a (Java)card environment using elliptic curve arithmetic, then exponentiation (point multiplications on a curve) is easier available then multiplications (point additions on a curve). The reason for this is that exponentiation resembles the basic Diffie-Hellman operation available from a cryptographic processor in the card. In other words, the ability to perform the randomization operations with only exponentiations would be convenient from an implementation perspective.

The following construction, *affine PPCA*, provides this. In the affine PPCA application one places two randomizations of the same polymorphic pseudonym, e.g. $(A_1, B_1) = (g^r, S \cdot y^r)$ and $(A_2, B_2) = (g^s, S \cdot y^s)$ with $r, s \in_R \mathrm{GF}(q)$ different. Here we have simply denoted the intermediate pseudonym by $S \neq 1$. When the application is requested a randomized polymorphic pseudonym, it first generates $k \in_R \mathrm{GF}(q) \setminus \{0, 1\}$ and then returns two *random affine representatives*, i.e. the two pairs of $(a_1, b_1) = (A_1^k, B_1^k)$ and $(a_2, b_2) = (A_2^{1-k}, B_2^{1-k})$ based on (A_1, B_1) and (A_2, B_2) to the identity provider. Note that this only requires exponentiations. After receipt, the identity provider pairwise multiplies the output, i.e.:

$$\begin{aligned}(a_1 \cdot a_2, b_1 \cdot b_2) &= (g^{r \cdot k + s \cdot (1-k)}, S^{k+1-k} \cdot y^{r \cdot k + s \cdot (1-k)}) \\ &= (g^{r \cdot k + s \cdot (1-k)}, S \cdot y^{r \cdot k + s \cdot (1-k)}).\end{aligned}$$

Clearly this result is a randomization of the original ElGamal encryption. In essence it constitutes to a random 'point' on the 'line' through the exponents of the original randomized polymorphic pseudonyms, hence the name 'affine'. This end result will not allow the identity provider to identity or link the card by Proposition A.1 in Appendix A. However, the identity provider also gets access to the individual results (a_1, b_1) and (a_2, b_2) and the question arises if these will allow the identity provider to identity or link a card. In Proposition B.2 in Appendix B we sketch a proof that this is not the case either. Here we show that the ability to distinguish issued affine PPCA based cards is as hard as breaking the Decision Diffie-Hellman problem with respect to the whole group G, which is considered to be intractable in our setting.

6.3 e-ID Token Status Services

The setup of polymorphic pseudonyms also introduces a convenient setup for e-ID wide token status services. A token status service provider is just an e-ID service provider and as part of registration the user registers there as well. Recorded are the reference to the identity provider and an identification of the authentication token used. A token identification only needs to be unique in the context of the user. That is, it is not indirectly identifying the user. It could for instance simply be a sequence number. The user can inspect and maintain the records there through the e-ID infrastructure. When the user authenticates with an identity provider an encrypted pseudonym of the user is sent to the token status service and asks for the status. This could be done by the identity provider, the service provider or a broker.

This resembles the setup of the Online Certificate Status Protocol (OCSP) setup, cf. [11]. Of course the authentication tokens are administered by the identity provider so typically it knows the token status itself. However with this setup the identity provider could outsource its status services. Also, there could be other reasons - not known to the identity provider - why the token could be revoked.

To illustrate, if the identity provider is a bank that has deployed authentication based on debit cards (e.g., EMV CAP) then the record will contain the name of the bank, a reference to the use of debit card authentication and the sequence number of the user card. The sequence number is used to deal with replacement of cards: the old number is invalid but the new number is valid. As an illustrative use case, suppose a user has lost his wallet with several banking cards in it but has not lost its e-ID card. The user could authenticate itself with its e-ID card and then could be redirected to the token status service and revoke all the bank cards in one move. All service providers where the user is registered could periodically query the revocation service and update the revocation status of the authentication tokens and act accordingly, e.g. by physically sending out a new authentication token and letting it being activated with the e-ID.

The described e-ID revocation services also allow revocation of the PPCA application introduced above. Each such card application gets a user specific sequence number which is registered at the e-ID revocation service. Each time the user uses PPCA at the identity provider it asks for the status of the card. When the card needs to be revoked the user logs in to the revocation service (with PPCA or with a regular e-ID token) and revokes it. If one deploys a shared chip authentication key in PPCA, see above, this key could be used as an identification of the card (provided a user never gets the same authentication key twice). Other variants of this idea can be devised.

6.4 Polymorphic Attributes

Imagine an attribute provider that possess attributes of the user that enables indirect identification of the user, e.g. a full postal code, a full date of birth or a social security number. In a regular (SAML) setup, [18], the attribute provider would encrypt the attributes under a public key of the service provider it is meant for. But particulary with attributes that allow indirect identification this would enable the attribute provider to follow the movements of its users. Particularly in the context of PPCA this is not satisfactory.

One of the elements in our scheme is that one can encrypt information (an intermediate pseudonym) under a public key of which the private part is not known by the parties (e.g., identity providers, brokers) that use the encrypted information. Instead these parties have cryptographic keys to transform the cryptograms to other cryptograms for other parties (service providers) that do have access to the required private keys. One can easily use the same idea to devise 'polymorphic attributes' that are particularly interesting to use in combination with PPCA.

The idea is that a party that knows attributes of the user, encrypts those with such a transformable public key and sends these to an attribute provider using the pseudonym of the user in the attribute provider domain. If the user authenticates at an identity provider, particularly through PPCA, it can ask the attribute provider to transform its encrypted attribute to a form decryptable by the service provider and have them sent to it. In this way, only the service provider gets access to the attribute.

Provided strings are not too long, they can be efficiently bijectively embedded in elliptic curves by using a standard encoding of a string as a number. For instance, one can reserve some room, say one byte, in the string supporting that the whole encoded string has an x-coordinate has a matching y-coordinate on the curve. By proceeding in this way, one could in fact perform ElGamal encryptions directly on the encoded attributes. In this way the encrypted attributes could be randomized (cf. the remarks following Proposition 2.1) further improving the privacy properties. We finally remark that ElGamal encryption has very efficient properties with respect to encrypting the same plaintext under different public keys. In [23] is shown that the same randomization exponent can be used without security implications.

7 Conclusion

In this paper we have described polymorphic pseudonymization based on the homomorphic properties of ElGamal encryption. We have used this to design an e-ID infrastructure with special security and privacy properties. Pseudonyms for service providers are consistent throughout the infrastructure even if they were formed by different identity providers. The scheme limits the need of a centrally used pseudonym provider through the concept of polymorphic pseudonyms. These are centrally generated once during registration and allow the identity provider to let the user be identified at service providers through pseudonyms. While supporting e-ID parties can process (encrypted) pseudonyms and can associate pseudonyms with attributes and authorisations they are not known to them, nor is their use linkable. Only at the service provider pseudonyms and attributes are known.

This setup also conveniently supports token status services based on pseudonyms. The polymorphic pseudonyms can also be placed in a simple card application (PPCA) and randomized before sent out of the card over a secured channel similar to how fingerprints are read from European electronic passports. This allows for pseudonymous identification through a central identity provider with the paradoxical property that the provider cannot establish which card is actually used or link it with earlier usage of the card. PPCA provides for anonymity in the scheme without losing computability with other means of authentication. Also, PPCA cards can be conveniently revoked through the token status services. PPCA also allows for a setup where users can conveniently place attributes and authorizations from the "named world" under a pseudonym at an attribute provider and then use them in the "pseudonymous world" through PPCA. In this way PPCA combines the best of both worlds.

To further enhance societal acceptance of the scheme (and trust from the service providers in the scheme) we have incorporated access for law enforcement agencies in the scheme through a Central Information Point e-ID Investigations(CIPEI). We have also designed cryptographic controls around this access providing preventive security and which will also facilitate independent auditing and supervision of the CIPEI.

References

1. M. Bellare, R. Canetti, and H. Krawczyk, Keyed Hash Functions and Message Authentication, Proceedings of Crypto'96, LNCS 1109, pp. 1-15.
2. Bundesamt für Sicherheit in der Informationstechnik (BSI), Technical Guideline TR-03110-2 Advanced Security Mechanisms for Machine Readable Travel Documents Part 2 Extended Access Control Version 2 (EACv2), Password Authenticated Connection Establishment (PACE), and Restricted Identification (RI) Version 2.10, 20. March 2012. See https://www.bsi.bund.de/EN/Publications/TechnicalGuidelines/TR03110/BSITR03110.html.
3. Bundesamt für Sicherheit in der Informationstechnik (BSI), Technische Richtlinie TR-03127 Architektur elektronischer Personalausweis und elektronischer Aufenthaltstitel Version 1.15, 1. August 2012. See https://www.bsi.bund.de/DE/Publikationen/TechnischeRichtlinien/tr03127/tr-03127.html
4. Bundesamt für Sicherheit in der Informationstechnik (BSI), Technical Guideline TR-03110-2 Advanced Security Mechanisms for Machine Readable Travel Documents and eIDAS Token Part 2 Protocols for electronic IDentification, Authentication and trust Services (eIDAS) Version 2.20 beta2 11. March 2014 See https://www.bsi.bund.de/EN/Publications/TechnicalGuidelines/TR03110/BSITR03110.html.
5. Bundesamt für Sicherheit in der Informationstechnik (BSI), Technical Guideline TR-03110-3 Advanced Security Mechanisms forMachine Readable Travel Documents and eIDAS Token Part 3 Common Specifications Version 2.20 beta2 11. March 2014
6. T. ElGamal, A Public Key Cryptosystem and a Signature scheme Based on Discrete Logarithms, IEEE Transactions on Information Theory 31(4), 1985, pp. 469-472.
7. EMVCO, EMV Integrated Circuit Card Specifications for Payment Systems, Book 2, Security and Key Management,Version 4.3 November 2011. See https://www.emvco.com/.
8. European Commission, Laying down technical requirements with regard to driving licences which include a storage medium (microchip), COMMISSION REGULATION (EU) No 383/2012, 4 May 2012.
9. Fox-IT, Black Tulip Report of the investigation into the DigiNotar Certificate Authority breach, 13 August 2012.
10. ICAO, Machine Readable Travel Documents Part 3: Machine Readable Official Travel Documents, Doc 9303, Sixth Edition 2006. See http://www.icao.int/publications/pages/publication.aspx?docnum=9303
11. IETF, Online Certificate Status Protocol - OCSP, Network Working Group Request for Comments: 2560, June 1999. See http://www.ietf.org.
12. IETF, Request for Comments 5639, Elliptic Curve Cryptography (ECC) Brainpool Standard, Curves and Curve Generation, March 2010, see http://www.ietf.org.
13. ISO, ISO/IEC 18033-2:2006 Information technology - Security techniques - Encryption algorithms – Part 2: Asymmetric ciphers, 2006.
14. N. Koblitz. Elliptic curve cryptosystems. in Mathematics of Computation 48, 1987, pp. 203209
15. National Institute of Standards and Technology (NIST), Security Requirements for Cryptographic Modules, FIPS 140-2, May 25, 2001. See http://csrc.nist.gov.
16. National Institute of Standards and Technology (NIST), Advanced Encryption Standard (AES), FIPS 140-2, November 26, 2001. See http://csrc.nist.gov.
17. National Institute of Standards and Technology (NIST), Secure Hash Standard (SHS), FIPS 180-4, March 2012. See http://csrc.nist.gov.

18. OASIS, Security Assertion Markup Language, version 2.0. See https://wiki.oasis-open.org.
19. RSA Laboratories, PKCS #11: Cryptographic Token Interface Standard, 16 April 2009. See http://www.emc.com.
20. A. Shallue, A., C. van de Woestijne, Construction of rational points on elliptic curves over finite fields, ANTS , Lecture Notes in Computer Science, Volume 4076, Springer, 2006, pp. 510-524.
21. Project Stork, Quality authenticator scheme, 3 March 2009. Available from http://www.eid-stork.eu.
22. Eric Verheul, Evidence that XTR is more secure than supersingular elliptic curve cryptosystems, Journal of Cryptology (JOC) 17(4), pp. 277-296, 2004.
23. Eric Verheul, Binding ElGamal: A Fraud-Detectable Alternative to Key-Escrow Proposals, Proceedings of Eurocrypt 1996, LNCS 1233, pp. 119-133.

A Appendix: sketch of security proof for basic PPCA

The following is a sketch of the security proof that the identity provider is not able to distinguish issued PPCA cards through the randomized polymorphic pseudonyms they generate. The takes the form of a game in which two parties are involved. A party P and an adversary A. Party P has two arbitrary ElGamal encryptions $\mathcal{EG}_1 = \mathcal{EG}(S_1, y, k)$ and $\mathcal{EG}_2 = \mathcal{EG}(S_2, y, k)$ of plaintexts $S_1 \neq S_2$ under the same public key y of which neither party A or P has the private part. Two phases exist in the game: a learning phase and a final phase. During the learning phase, the adversary A is allowed to refer to either ElGamal encryptions, by either 1 or 2, and to get a randomized version of the corresponding ElGamal encryption $\mathcal{EG}_1$ or $\mathcal{EG}_2$. In the final phase, P flips a coin. If it is tails, P sends random affine representations based on $\mathcal{EG}_0$. Heads corresponds to Scenario 1: P sends two randomizations of either $\mathcal{EG}_1$ or $\mathcal{EG}_2$, he flips the coin again to decide which of the two. Tails correspond to Scenario 2: P sends randomizations of both $\mathcal{EG}_1$ and $\mathcal{EG}_2$ in fashion, he flips the coin again to decide the order. After sending the adversary A then has to state which of the scenarios P chose, i.e. Scenario 1 or Scenario 2.

This game corresponds to the context where an identity provider tries to link a user through its PPCA output. Indeed, the learning phase might in theory occur during PPCA card production and the final phase is a representation of the identity provider trying to distinguish PPCA cards during usage. The following result states that if the identity provider is able to distinghuish PPCA cards, he can break the Decision Diffie-Hellman problem with respect to the generator g which we assumed to be hard.

Proposition A.1 *If the adversary has a probability of winning the game substantially better than guessing he can break the Decision Diffie-Hellman problem with respect to the generator g*

Proof: Suppose that the adversary has an algorithm $\mathcal{A}()$ to win the game with a probability of winning substantially better than guessing. Let $g, h, K = g^k, L = h^l \in G$ be an arbitrary Decision Diffie-Hellman instance with respect to g. We will show that the adversary can decide whether $k = l$ or not with non-negligible advantage. To this end, the adversary introduces an ElGamal public key $y = h$ with private key $x = DL_g(h)$ unknown to all participants. Next the adversary chooses $1 \neq S \in_R G$ and forms the following two ElGamal encryptions under public key y:

$$\mathcal{EG}_1 = (g, S \cdot y, y)$$
$$\mathcal{EG}_2 = (K, S \cdot L, y).$$

It follows that the decryption of the second pair is $S \cdot L \cdot K^{-x} = S \cdot g^{x(l-k)}$. This is equal to S if and only if $k = l$. In the latter case $\mathcal{EG}_1$ is a randomized version of $\mathcal{EG}_2$. As the adversary can randomize ElGamal encryptions himself he can simulate the learning phase himself. The adversary's algorithm $\mathcal{A}()$ will then output Scenario 1 substantially more often Scenario 2 if and only if $k = l$. The adversary can also repeat this game by choosing another random S and by randomization of the K and L, i.e. by replacing them with $K' = K^{r_1} \cdot g^{r_2} = g^{k \cdot r_1 + r_2}$ and $L' = L^{r_1} \cdot h^{r_2} = g^{k \cdot r_1 + r_2}$ By repeating the game the adversary is able to solve the given Decision Diffie-Hellman problem with a probability arbitrarily close to one. $\square$

B Appendix: sketch of security proof for affine PPCA

In this appendix we sketch a proof that an identity provider is not able to link or identify a user based on affine PPCA, i.e. the variant that uses no multiplications but only exponentiations. This proof is similar to the one given in Appendix A but more complex.

We introduce a game between a party P and an adversary A similar to the one in Appendix A. Party P has a random ElGamal encryption $\mathcal{EG}_0 = \mathcal{EG}(S_0, y, k_0)$ of a plaintext $S_0 \neq 1$ under a public key y of which neither party A or P has the private part. P has also generated two randomizations $\mathcal{EG}_{0,1}$, $\mathcal{EG}_{0,2}$ of the encryption $\mathcal{EG}_0$. This corresponds to a specific affine PPCA card instance. Moreover, P also has a collection of many other ElGamal encryptions, $\mathcal{EG}_i = \mathcal{EG}(S_i, y, k_i)$ for $i = 1, 2, 3\ldots$ of other distinct plaintexts $S_i \neq 1$ but under the same public key y. For each of those P has also generated two randomizations. During the learning phase, the adversary is allowed to ask for random affine representations based on $\mathcal{EG}_{0,1}$ and $\mathcal{EG}_{0,2}$. In the final phase, P flips a coin. If it is heads, P randomly selects any of the other ElGamal encryptions and sends random affine representations based on that (Scenario 2). P then asks the adversary which of the scenarios he followed. In Appendix A we only assumed that the adversary can win the game for ElGamal encryptions related to a fixed generator g. In the result below we assume he can win the game for any generator, i.e. for any element not one. This is a reasonable assumption as the choice of the generator g should play no security role. Before coming to the main result, we need a lemma.

Lemma B.1 *Let G be a cyclic group generated by j and let y be an ElGamal public key with respect to j with private key x, i.e. $x = DL_j(h)$. Let (U, V) and (X, Y) be two pairs randomly chosen in $\mathrm{GF}(q)^* \times \mathrm{GF}(q)^*$. Then with overwhelming probability $(1 - 3/q)$ these pairs can be constructed as affine representatives based on two unique randomized versions R_1 and R_2 of an Elgamal encryption of a unique plaintext $S \neq 1$ under y.*

Proof: We are looking for $k, r, s, S \in \mathrm{GF}(q)$ such that

$$\begin{aligned}(U, V) &= (j^{r\cdot k}, S^k \cdot y^{r\cdot k})\\ (X, Y) &= (j^{s\cdot(1-k)}, S^{1-k} \cdot y^{s\cdot(1-k)})\end{aligned} \tag{12}$$

Then $S^k = V \cdot U^{-x}$ and $S^{1-k} = X \cdot Y^{-x}$. This will uniquely determine S. We assume that $S \neq 1$; which is the case with probability $1 - 1/q$. The number k will also be uniquely determined from $S^k = V \cdot U^{-x}$, we automatically have that $S^{1-k} = X \cdot Y^{-x}$. With probability $1 - 2/q$ we will have that $k = 0, 1$. So the probability that S is not one and that k is not zero or one is $1 - 3/q$. Next, r is determined by $DL_j(U)/k$ and s is determined by $DL_j(X)/(1-k)$. It is a simple verification that the k, r, s, S determined this way satisfy the two equalities in Formula (12). □

Proposition B.2 *If, for any generator j of group G, the adversary has a probability of winning the game, substantially better than guessing, he can break the Decision Diffie-Hellman problem in the group G.*

Proof: Let $j, h, A = j^k, B = h^l \in G$ be an arbitrary Decision Diffie-Hellman instance in the group G. Suppose that the adversary has an algorithm $\mathcal{A}_j()$ with which he can win the game with respect to a generator j substantially better than guessing. The adversary then chooses $r, s, a, b \in_R \mathrm{GF}(q)$ and forms the following two randomized ElGmamal encryptions of the plaintext $j^a \cdot h^b$ in the scheme with generator j and the public key h with private key $x = DL_j(h)$ unknown to all participants:

$$\begin{aligned}&(j^r, j^a \cdot h^b \cdot h^r)\\ &(j^s, j^a \cdot h^b \cdot h^s)\end{aligned} \tag{13}$$

Note that the adversary can simulate the learning phase with respect to these randomizations. Next, the adversary forms the pairs

$$\begin{aligned}&(A^r, A^a \cdot B^b \cdot B^r)\\ &((A^{-1} \cdot j)^s, (A^{-1} \cdot j)^a \cdot (B^{-1} \cdot h)^b \cdot (B^{-1} \cdot h)^s)\end{aligned} \tag{14}$$

Note that expressed in k, l the first pair in Formula (14) is equal to

$$\begin{aligned}(j^{r\cdot k}, j^{a\cdot k} \cdot h^{b\cdot l} \cdot h^{r\cdot l}) &= (j^{r\cdot k}, j^{a\cdot k} \cdot h^{b\cdot k} \cdot h^{b\cdot l - b\cdot k + r\cdot l - r\cdot k} \cdot h^{r\cdot k})\\ &= (j^{r\cdot k}, j^{a\cdot k} \cdot h^{b\cdot k} \cdot h^{(b+r)\cdot(l-k)} \cdot h^{r\cdot k}) = (U, V).\end{aligned}$$

The second expression in Formula (14) is equal to

$$\begin{aligned}(j^{s(1-k)}\ ,\ & j^{a(1-k)} \cdot h^{b\cdot(1-l)} \cdot h^{s\cdot(1-l)}) \\ &= (j^{s(1-k)}, j^{a(1-k)} \cdot j^{b(1-k)} \cdot h^{b\cdot(1-l)-b\cdot(1-k)+s\cdot(1-l)-s\cdot(1-k)} \cdot h^{s\cdot(1-k)}) \\ &= (j^{s(1-k)}, j^{a(1-k)} \cdot h^{b(1-k)} \cdot h^{-(b+s)\cdot(l-k)} \cdot h^{s\cdot(1-k)}) = (X, Y).\end{aligned}$$

We conclude that the pairs in Formula (14) are affine representatives of the randomizations in Formula (13) if and only if $k = l$. The idea is now that Formula (14) produces random affine representatives if $k \neq l$, allowing the adversary to use its algorithm $\mathcal{A}_j()$ to distinguish which case applies.

To further elaborate on this, note that by running through all r, s, T and a, b with the property that $a + x \cdot b = T$ Formula (13) will run through all ElGamal encryptions of the element $j^a \cdot h^b$ under public key y using generator g. In fact, we will run through all ElGamal encryptions of the element $j^a \cdot h^b$ under public key h^f using generator j^f for any non-zero f.

By replacing r, s by $f{\cdot}r, f{\cdot}s$ in Formula (14) we can transform this into the setting of using generator j^f and public key h^f. We can also replace A and B by anything of the form $A' = A^{t_1} \cdot j^{t_2} = j^{k\cdot t_1 + t_2}$ and $B' = B^{r_1} \cdot h^{r_2} = j^{k\cdot r_1 + r_2}$ with random t_1, t_2. If $k = l$ then the corresponding k', l' will also be equal and will run through all elements in GF(q). If $k \neq l$ then the corresponding k', l' are such that k' and $l - k \neq 0$ will run through all possibilities. By respecting the earlier chosen formula $a + x \cdot b = T$ we still have freedom to choose b. Now by appropriately choosing k' and f we can let U and X be any element in G. By subsequently choosing $l' - k'$ and b we can let V and Y be any element in G. It follows that if $k = l$ then Formula (14) corresponds to all random affine representatives based on Formula (13). Furthermore, if $k \neq l$ then Formula (14) corresponds to two arbitrary pairs of elements in G.

From Lemma B.1 it follows that, with overwhelmingly probability, the pairs in Formula (14) correspond to affine representatives based on a unique pair of ElGamal randomizations under a public key with respect to j^f. That is, the pairs in Formula (14) are either affine points based on Formula (14) or can be considered as affine representatives arising from another ElGamal encryption.

We conclude that if the adversary can win the game using his algorithm $\mathcal{A}_{j^f}()$ with successive randomizations of r, s, a, b, A, B, f. That is, he will successively play the game with randomized versions of the ElGamal encryption in Formula (13) in the learning phase and then will use his algorithm to decide if the pairs in Formula (14) are affine representatives based on those or not. Doing so the adversary can establish if $k = l$ or not with probability arbitrarily close to 1 , i.e. to solve the instance of the Decision Diffie Hellman problem we started with. $\square$