Master Thesis Computing Science

Polymorphic Pseudonymization in Educational Identity Federations

Hans Harmannij - hans@harmannij.nl

Supervised by:

prof. dr. E.R. Verheul (Radboud University)	drs. J. van Dijk (SURFnet)

22nd June 2016

Abstract

This work investigates the use of polymorphic pseudonyms in educational identity federations. Polymorphic pseudonyms can provide privacy enhancements in federated identity management (FIM) systems. An important property is that it limits linkability of users across different parties in a federation, moreso than conventional targeted pseudonyms. Furthermore, it limits the ability for intermediaries, such as a hub in a so called hub-and-spoke federation to aggregate information about users. This is especially the case when polymorphic pseudonyms are combined with polymorphic attributes.

A proof-of-concept implementation for polymorphic pseudonyms has been created, with accompanying plugins for SimpleSAMLphp, Shibboleth IdP and SP, and Microsoft AD FS. This shows that this technology can be used in standard software used in identity federations, without changes to the software itself.

We look at two variations of the use of polymorphic pseudonyms. These variations differ in the role of a hub in a hub-and-spoke federation, and of identity providers. Both variations have advantages and disadvantages. The variation with less responsibility for the hub cannot be implemented in all software products, without changes to those products.

We look at the use of polymorphic pseudonyms in two use cases, but much of the work can be applied to identity federations in general. The first use case is SURFconext, the hub-and-spoke identity federation provided by SURFnet to Dutch higher education and research institutes. The second use case is that of Kennisnet, which provides ICT infrastructure to Dutch primary, secondary and vocational education, for the system in which publishers and distributors of electronic learning materials and schools have to communicate about the pupils using these learning materials.

Contents

Chapter 1

Introduction

With federated identity management (FIM) users can authenticate to their organisation, which acts as an identity provider (IdP), and can subsequently access service providers (SPs) in that federation. In a so called *hub-and-spoke* federation IdPs are connected to a central hub, which is in turn connected to the SPs in the federation. In a *full mesh* federation IdPs and SPs are connected directly. FIM is explained in more detail in Chapter 3.

1.1 SURFnet Use Case

SURFnet provides FIM to organisations in Dutch higher education and research, through SURFconext, which is a hub-and-spoke federation.

When an IdP authenticates a user to SURFconext, a NameID is passed that identifies the user. SURFconext passes a NameID for the user to the SP the user wants to use. This NameID is different from the NameID from the IdP and unique for each SP. This protects the privacy of the users. Unique NameIDs prevent different SPs from linking the data they have about a user, and thus creating a more detailed profile than they need to have. SURFconext must maintain links between the NameIDs from the IdPs and the corresponding NameID for the SPs. Currently this is solved by storing all these links in a database.

Storing the links between NameIDs from IdPs and SPs in a database has downsides. If the database leaks, all NameIDs for different SPs can be linked, which we wanted to prevent. Furthermore, when the database is unavailable, authentication cannot take place. Finally, there is also the cost of maintaining the database.

Polymorphic pseudonyms are described in [54] and slightly differently in [46]. Polymorphic pseudonyms provide a way to create unique pseudonyms for different SPs. A polymorphic pseudonym is passed to the SP in encrypted form, and the SP can then decrypt it to get its unique pseudonym. Polymorphic pseudonyms may therefore be able to replace the database solution.

The hub in the hub-and-spoke federation handles a lot of privacy sensitive information which it passes on between IdPs and SPs. Polymorphic pseudonyms may be able to solve this, because the hub only sees encrypted pseudonyms which it cannot link together.

1.2 Kennisnet Use Case

Kennisnet has a similar use case. Schools order learning materials for their pupils from a distributor, including electronic learning materials from certain publishers. The pupil must get online access to these electronic learning materials at the publisher. The publisher must know whether the pupil should get access, depending on what the pupil ordered from the distributor. The publisher must be able to communicate test results back to the school.

Currently, personally identifying information such as the full name of the pupils is used to communicate about pupils [22]. The Dutch schools received a *Big Brother Award* for this [21], which is an award given out to privacy infringers. A privacy friendly solution is needed, where publishers and distributors do not get more information about the pupil than necessary, and information from different publishers and distributors cannot be linked, unless allowed by the school. Polymorphic pseudonyms can also provide a solution here.

1.3 Research Questions

In this thesis the following questions will be answered:

1. What pros and cons are there when using polymorphic pseudonyms in a hub-and-spoke federation?
2. There are variations available of the polymorphic pseudonym system. What are pros and cons of different variations?
3. Does the use of polymorphic pseudonyms fit in a standard SAML environment? How much extra processing is needed?
4. What is the impact on existing identity federations, such as Kennisnet and SURFconext, for IdPs, SPs as well as for the hub?
5. Which variation is the best fit for the SURFconext and Kennisnet use cases?

1.4 Research Methods

We give an overview of privacy, trust and usability requirements for authentication systems, and especially federated identity management. We can then evaluate whether polymorphic pseudonyms help us to meet these requirements, or prevent us from meeting them. This gives us a partial answer to questions 1 and 2. Another part of the answer to these questions is given by implementing polymorphic pseudonyms in existing software for FIM. This will tell us what the problems are when we want to use polymorphic pseudonyms in practice. This implementation, and therefore seeing what is necessary for this implementation, also gives an answer to question 3 and 4.

We give a description of the current situation and plans in the SURFconext and Kennisnet use cases, and what the considerations are that have to be made when changes are made. By comparing the pros and cons from question 2 to the current situation and the design considerations we can answer question 5.

1.5 Scope

Many parts of this thesis apply to the use of polymorphic pseudonyms in identity federations in general. The main focus is, however, SURFconext, and the hub-and-spoke model they use. We will also make some remarks on the Kennisnet use case.

Some of the advantages of polymorphic pseudonyms cannot be achieved in many set-ups with polymorphic pseudonyms alone. We will therefore also describe polymorphic attributes, which can help reaching all of the potential polymorphic pseudonyms have. The main focus of this thesis will however remain polymorphic pseudonyms.

1.6 Outline

We will start by looking at requirements for authentication systems in Chapter 2. This will then be further elaborated for federated identity management in Chapter 3. In this chapter we will also give a summary of the Security Assertion Markup Language (SAML) specification that is often used for FIM. In Chapter 4 we will look at the current situation in the Netherlands. We will describe the situation for the Kennisnet and SURFconext use cases, but also look at iDIN and Idensys, two other developments in the Netherlands. There has been earlier work on privacy in identity federations. We will look at some of this related work in Chapter 5. Chapter 6 describes how polymorphic pseudonyms work, and how they can be used in different models. The underlying cryptography is explained in Chapter 7. When we want to change the current situation, there are some considerations that have to be made. These are decribed in Chapter 8. The implementation work that was done is described in Chapter 9. An extension to polymorphic pseudonyms are polymorphic attributes. Those are described in Chapter 10. We have described requirements for authentication systems. In Chapter 11 we will look at how polymorphic pseudonyms and attributes perform against those requirements. In Chapter 12 we will see how some of the requirements can be met with conventional cryptographic techniques, but also where this falls short. We will come to our conclusion in Chapter 13.

Chapter 2

Requirements for Online Authentication

2.1 Trust

Authentication deals with verifying the truth of a claim which is made by some entity. In most cases the term authentication is used for identity authentication: if a user claims to have a certain identity, usually we also want authentication to ensure that the identity claimed by the user is indeed his identity. To accomplish this, it is necessary that there is trust. Without trust in an authentication system, we would still not be sure that the user we have identified is the actual user using the system. The European Commision has adopted the eIDAS regulation [49]. Article 8 of this regulation specifies three assurance levels: low, substantial and high. This is further implemented in a commission implementing regulation [12]. Authentication with only a password has assurance level low. Stricter requirements have to be met for assurance levels substantial and high. One of the requirements for level substantial is that at least two authentication factors are used. The high assurance level requires that an authentication means has further protection against duplication and tampering. Assurance levels are based on the quality and reliability of these elements:

- Enrolment. This includes verifying and proofing the identity of the user, and binding of electronic identification means of natural and legal persons.
- Electronic identification means management. This includes the design and characteristics, issuance, suspension, revocation and reactivation, and renewal and replacement of identification means.
- Authentication. This states requirements about the authentication mechanism.
- Management and organisation

2.2 Privacy

When information about users is processed, privacy starts to become an issue. This is also the case for authentication systems. A tension between privacy and

authentication is described by [40]: authentication often involves disclosure or confirmation of personal data. This does not mean they are polar opposites, but shows that privacy is an important issue in authentication.

Another way that authentication systems influence privacy is when a user authenticates to multiple services in such a way that these services can link these interactions. This is for example the case when an e-mail address is used during authentication. Services can easily link the information they have about the user with a certain e-mail address.

When authentication for multiple services is handled by one party, as is the case in FIM, this party may be able to know the pseudonym by which the user is known to these services, and know which services you use and when you use those. FIM is explained in more detail in Chapter 3.

2.3 Usability

Another important aspect of authentication systems is usability. If a system is not user friendly, users are less likely to use it, and more likely to circumvent it [1]. Some of the recommendations for identity management systems, which are related to usability, given by [16] are:

- *Make the "path of least resistance" the secure path. Integrate identity management into the browsers, operating systems, and applications people already use, and make it easy for them to configure and operate systems correctly.*
- *Reduce cognitive burden. Don't replace one burden with others, and evaluate how your system will be used in the larger context of other systems.*
- *Reduce the number of trust decisions users have to make. Don't try to achieve consent by overwhelming users with more warnings, dialogs, and indicators.*

In line with the first recommendation is also to allow users to use cards, devices or other authentication tokens they already have to authenticate to new services. FIM (Chapter 3) allows users to use credentials they already have for multiple services.

Users having to remember many passwords causes problems. There is a large cognitive burden associated with this, and users tend to use insecure password practices because of this [1].

2.4 Monitoring

Another requirement can be the possibility to perform monitoring. This can help solving problems in the system, but also aid in detecting fraudulent user behaviour. For law enforcement purposes, it may be necessary to trace actions back to the person who performed those actions. For the use cases which we are looking at, this is not a requirement, however. We will therefore not look further into this.

2.5 Authentication and Trusted Third Parties

In authentication systems, a trusted third party (TTP) can play a role. It can play a role in both management of authentication tokens, and in the use of the token.

2.5.1 No TTP

Simple username/password authentication, where a user registers at the service provider itself does not use a TTP. Another example of authentication without a TTP are the UAF [38] and U2F [51] specifications from the FIDO Alliance. In these specifications, the users have a FIDO enabled device. Before first use of a service, the device generates a public key pair. The public key is given to the SP, and ownership of the corresponding private key is proven by signing a challenge from the SP. In this way the public key of the user gets associated with his account at the SP. When authenticating to the service, a challenge-response protocol is again used to prove ownership of the private key. All interactions are between the SP and the user's device. The user does need to trust the issuer of the device, but neither when authentication tokens are issued to the device, nor when the user authenticates to a service, a TTP is used.

With this approach, there is no trusted source for attributes. Users have to convince each SP of the validity of the attributes they are giving, such as their name, age or social security number.

2.5.2 TTP only for Token Issuance

An example of an authentication system where a TTP is only used for token issuance is the Belgian eID card [53]. Cards are issued by the Belgian government, and they hold private keys that can be used for signatures. No TTP is needed for this. Each card holds a key for authenticating the citizen, and one for non-repudiation signatures. For these two keys the card holds a certificate that contains the name and national registration number (NRN) of the cardholder. The NRN is the unique nation-wide identification number that is assigned to Belgian citizens. The certificate is shared when the card is used, which means also the name and NRN are shared. A certificate chain with a trusted certificate at the root makes sure that the signatures that are generated by the card can be verified without the need for a TTP.

2.5.3 TTP for Token Issuance and Use

In a federated authentication system, a TTP is used during the registration phase, in which an authentication token will be issued, and also every time the user logs in using his authentication token. We will look further into federated identity management in Chapter 3. Examples are SURFconext, Kennisnet's Entree federation, Idensys and iDIN. These are described in more detail in Chapter 4.

Chapter 3

Federated Identity Management

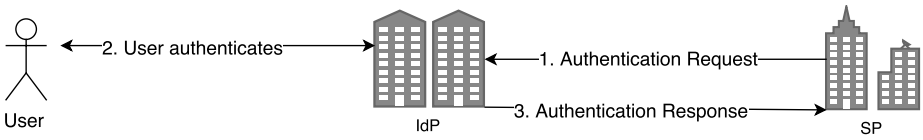


Figure 3.1: Simple setup of federated identity management (FIM)

Many online services require the user to log in. The user often needs to create an account at all the services he uses. It is however also possible to use federated identity management (FIM). Then the user has an account at an identity provider (IdP). When the user wants to use a service provider (SP) (also called relying party (RP)), he logs in with his account at the IdP, which then gives authentication statements to the SP, so it knows that the user is authenticated. This is depicted in Figure 3.1. The IdP may also pass attribute statements to the SP, to give information about the user. We are mainly interested in the interaction between IdPs and SPs. Therefore the interaction between IdP and user is not shown in the following figures. An identity federation is a collection of IdPs and SPs. Some federations are a *hub-and-spoke* federation, where both the IdPs and the SPs are connected to a central hub, and all authentications go through that hub (Fig. 3.2). In a *full mesh* federation (Fig. 3.3), IdPs are directly connected to the SPs [2].

One aspect of FIM is single sign on (SSO): the user logs in once and can then use multiple services without having to log in again each time. This is meant to improve usability, but there are security implications. When a user has logged in to use one service, he may also get logged in to another service without him noticing or wanting this to happen. This means that SSO conflicts with protection against duplication, as required for eIDAS assurance level 'high' (Section 2.1).

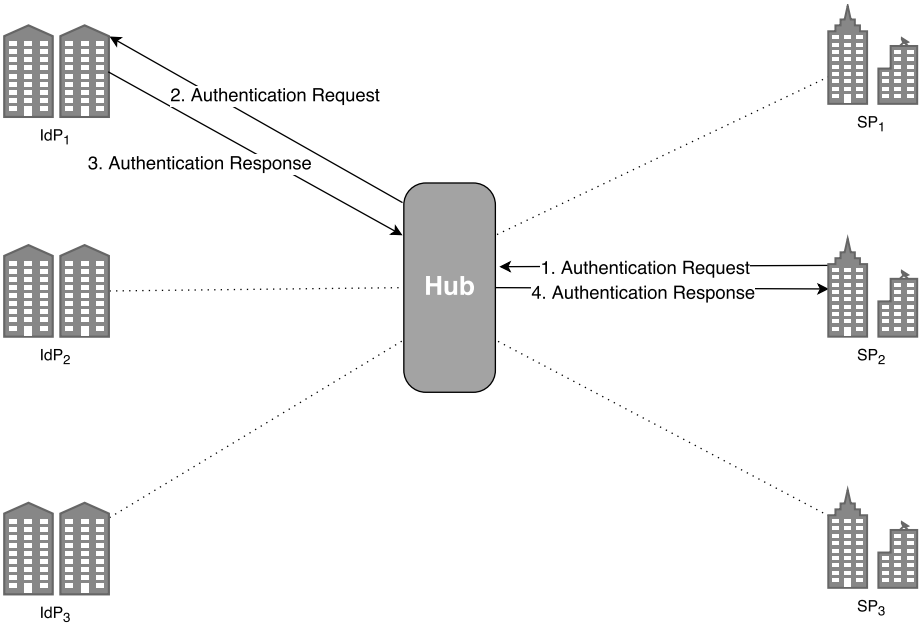


Figure 3.2: A hub-and-spoke federation, where a user wants to use SP_2 and authenticates to it using IdP_1

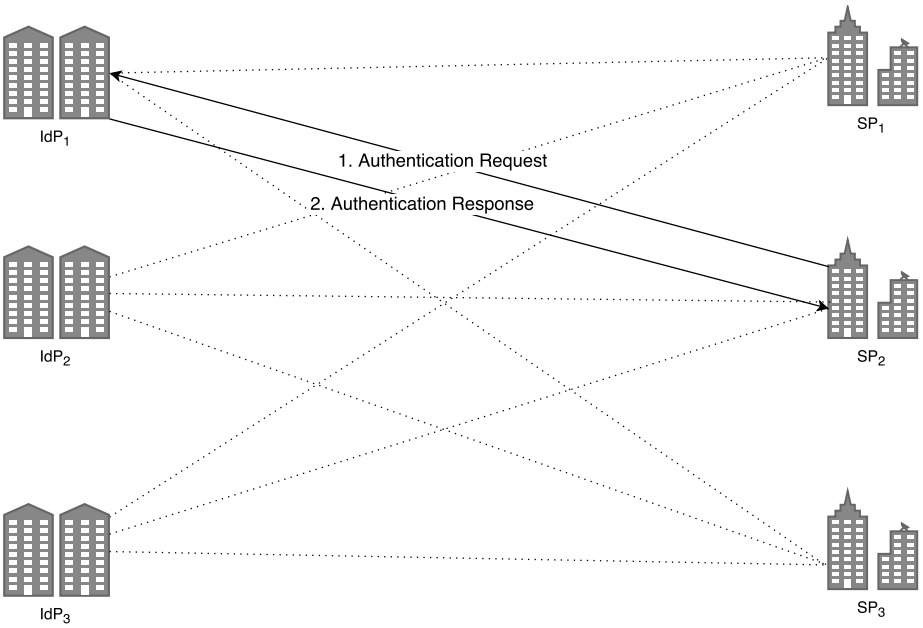


Figure 3.3: A full mesh federation, where a user wants to use SP_2 and authenticates to it using IdP_1

3.1 SAML

A widely used standard for FIM is the XML based Security Assertion Markup Language (SAML), standardised by OASIS [42]. The standard defines assertions, protocols, bindings and profiles. Assertions are statements about a user that the asserting party claims to be true. For example: an IdP claims that a user has successfully authenticated, or that he has certain attributes. Protocols describe how parties can make SAML requests and return appropriate responses. Assertions are transfered through these protocols. A binding describes how lower level protocols, like HTTP and SOAP, are used to transport SAML protocol messages. A profile describes how the SAML assertions, protocols and bindings can be used in a standard way for certain use cases.

3.1.1 Example

We will look at a web shop for students as an example of how SAML can be used for FIM. Bobby Tables is a student at the University of Harderwijk. He wants to buy a book on Unishop.com. Before he can order anything, Bobby has to log in. Unishop.com shows him a list of all universities from which students are allowed to log in, and Bobby chooses the University of Harderwijk. Unishop.com creates a protocol message to request authentication. This message is transported to the University of Harderwijk through the HTTP Redirect binding. This binding includes a redirect to the web site of the University of Harderwijk. At the university's website, Bobby is asked to log in. When he has done so, the University of Harderwijk creates a protocol message to respond to Unishop.com. This response contains an assertion stating that the user with identifier `a88c602021780133b9fee5f2fc8aba87cbf9982246` has successfully authenticated, and stating that he has these attributes:

name	Bobby Tables
email	b.tables@universityofharderwijk.com
address	Grotestraat 12, Harderwijk
affiliation	Student

Unishop.com can now see that Bobby is a student, and knows where to send the book. A confirmation of the sale can be sent to the specified e-mail address.

3.1.2 Web Browser SSO

What we described is an example of the Web Browser SSO profile[43]. This is the most important one for us. This profile describes how SAML messages can be sent via the browser of the user, to allow the user to authenticate to an IdP and consequently give him access to SPs.

SSO can be IdP-initiated, where a user first authenticates to his IdP, which then redirects him to an SP. For example: the University of Harderwijk shows a link to buy the book that is necessary for a course, and when a student clicks that link, he will be automatically authenticated to Unishop.com where he can buy the book. The earlier example is SP-initiated SSO, which is the more common variant. In this scenario, the user first tries to access an SP. The SP will let the user choose which IdP to use. The SP then sends an authentication request to the chosen IdP. The IdP will ask the user to authenticate, and upon

success send a response back to the SP, containing an authentication statement, and possibly some attribute statements.

3.1.3 Bindings

The bindings that can be used in Web Browser SSO are the HTTP Redirect binding, HTTP POST binding and HTTP Artifact binding. These bindings can be composed, such that different messages can be sent in one protocol exchange using two different bindings. Authentication requests can be sent using all these bindings. Authentication responses are often too long to be sent with the HTTP Redirect binding, and are therefore mostly sent with the HTTP POST or HTTP Artifact binding.

HTTP Redirect Binding

With the HTTP Redirect binding, SAML protocol messages can be sent within URL parameters. It can be used if the communicating parties need the user agent as intermediary, either because no direct channel is available, or because interaction with the user agent is necessary. An example of this is when the user needs to authenticate to an IdP.

The SAML message is encoded, and placed in a query variable called `SAMLRequest` or `SAMLResponse` of the URL of the targeted endpoint at the receiving party. The user is redirected to that URL, containing the SAML message in the query string.

The SAML message is URL encoded. By default, also DEFLATE [15] encoding is used to compress the message. If another encoding is used, this can be specified with the SAMLEncoding query string.

Theoretically, URL length is unlimited. In practice there are limits, so even though DEFLATE encoding is used, SAML messages sent with this binding cannot be too long. The HTTP POST or Artifact bindings can be used to send longer messages.

HTTP POST Binding

With the HTTP Post binding, SAML protocol messages can be sent as the base64 encoded content of an HTML form control. Like the HTTP Redirect binding, it can be used if the communicating parties need the user agent as intermediary, either because no direct channel is available, or because interaction with the user agent is necessary.

The SAML message is base64 encoded, and placed in a hidden form control with the name `SAMLRequest` or `SAMLResponse`, inside an XHTML document. The form uses POST as method, and the URL of the endpoint at the receiving party as action.

Typically there is some JavaScript that will automatically submit the form. If there is no script, or the script is not executed by the user agent, user interaction is necessary to submit the form. This way the receiving party will receive the message in a POST request from the user's browser.

HTTP Artifact Binding

With the HTTP Artifact binding, a reference to the actual message is sent, either with a SAMLArt query string variable, similar to the HTTP Redirect binding, or with a SAMLArt form element, similar to the HTTP POST binding. The artifact, which is the actual message, is sent via a direct channel, such as SOAP, between requester and responder. It can be used if the user agent is needed as intermediary, but the intermediary poses limitations to sending the entire message. This can be because of technical reasons, but also if the message should not be exposed to the user, and encryption is not practical.

3.1.4 Authentication Request

In the Web Browser SSO profile, the SP creates a SAML message, containing an `AuthnRequest` element. This element typically contains, among others, an `Issuer` element, which contains the EntityId of the SP.

The authentication request is typically sent to the SSO service of the IdP with the HTTP Redirect binding. This allows the IdP to interact with the user to let the user authenticate.

3.1.5 Response

The IdP sends a `Response` element. This element contains a `Status` element. Upon success, this element contains a `StatusCode` with value `urn:oasis:names:tc:SAML:2.0:status:Success`. When an error occurred, the `StatusCode` will contain a different value describing the error, and the `Status` may contain more information about the error.

In case of success, the `Response` contains one or more `Assertion`s. These can contain, among others, the `Subject` of the `Assertion` and zero or more `AuthnStatement`s and `AttributeStatement`s

Subject

The `Subject` element tells which user the `Assertion` is about. To this end it can contain a `NameID`. This is an identifier that identifies the user. The `NameID` has a format, some of which are defined by the SAML specification. It can however also be a different format. The formats defined in the SAML specification are:

- `urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified`: The format is unspecified, and the SP must decide for itself how to interpret the NameID.
- `urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress` : The NameID is an e-mail address.
- `urn:oasis:names:tc:SAML:1.1:nameid-format:X509SubjectName`: The NameID is an `X509SubjectName` as specified in [6]
- `urn:oasis:names:tc:SAML:1.1:nameid-format:WindowsDomainQualifiedName`: The NameID is a Windows domain qualified name
- `urn:oasis:names:tc:SAML:2.0:nameid-format:kerberos`: The NameID is in the form of a Kerboros principal name

- `urn:oasis:names:tc:SAML:2.0:nameid-format:entity`: The NameID is the EntityID of an entity that provides SAML services
- `urn:oasis:names:tc:SAML:2.0:nameid-format:persistent`: The NameID is a persistent identifier, specific for the SP. It is intended as a privacy protecting identifier, and must therefore follow some rules. The identifier must not be relatable to identifying information about the user. The identifier should not be shared with other providers unless they have established a shared identifier. It must not be logged without proper protection. The identifier should be unique for a user at a specific SP.
- `urn:oasis:names:tc:SAML:2.0:nameid-format:transient`: The NameID is a temporary identifier, which changes for different sessions of the same user.

Authentication Statement

An `AuthenticationStatement` tells that the `Subject` of the `Assertion` was authenticated in some way at a certain time.

Attribute Statement

An `AttributeStatement` tells that the `Subject` of the `Assertion` has some attributes associated with it. These attributes can come in the form of `Attribute` elements, of which the `AttributeStatement` contains one or more.

An `Attribute` has a name, and can have zero or more `AttributeValue`s. The `AttributeValue`s can contain any valid XML content. This ranges from simple types like strings and integers to complex XML types.

Sending the Response

The `AuthnRequest` is typically sent to the assertion consumer service of the SP with the HTTP POST binding. This way the user will be redirected back to the SP.

3.1.6 Hub-and-Spoke

In a hub-and-spoke federation, the hub acts as an IdP towards the SP. The SP will therefore send an `AuthnRequest` to the hub. The hub then acts as an SP towards the IdP of the user, and sends an `AuthnRequest` to the IdP. The IdP will respond to the hub with a `Response` as described above. The hub will then send a `Response` to the SP, containing the `Assertion`s from the `Response` it received from the IdP.

There are also other settings in which an intermediary is used. For example when a so called proxy is used to connect multiple SPs from one organization to a federation through a single connection. This works the same way as for the hub.

Scoping

To pass information about what lies behind the hub, `Scoping` can be used. It can be used in the `AuthnRequest` from the SP to the hub to tell the hub which

IdP or IdPs may be used. For this purpose an `IDPList` element can be used, containing one or more `IDPEntry` elements, which hold the EntityID's of the IdPs that may be used.

In the `AuthnRequest` the hub sends to the IdP, the `Issuer` element will contain the EntityID of the hub, and not the SP the request originated from. The `Scoping` element can therefore contain a `RequesterID` element. This `RequesterID` element contains the EntityID of the original SP. If there is more than one proxy, there can be multiple `RequesterID`s, each containing the EntityID of one of the entities that are behind the proxy that issued the `AuthnRequest`.

3.1.7 Security

SAML specificies mechanisms for both signatures and encryption [42, 44].

Without signatures a requester cannot rely on the validity of the assertions it receives. For example, for the HTTP Redirect and POST bindings, protocol message are sent via the user's browser. This allows the user to change these messages. With signatures it can be verified that messages come from the correct party. Also, if a hub or proxy leaves the signatures intact, the receiving party can verify that the signed data comes from the original provider of that data. XML Signatures [6] are used for this, and can be used on assertions, and on request and response messages.

XML Encryption [26] can also be used. This prevents, for example, the user to see the contents of encrypted parts, when the HTTP Redirect or POST binding is used. If an IdP encrypts data with the public key of the SP, intermediaries like hubs and proxies cannot see the contents of the encrypted elements. `Assertion` elements can be replaced by `EncryptedAssertion` elements, `NameID`s by `EncryptedID`s and `Attribute`s by `EncryptedAttribute`s. The `Encrypted*` elements contain the encrypted original elements.

In SURFconext only signatures, and no XML encryption is used. Requests and responses are sent via HTTPS, making sure outside observers cannot see the contents of these messages. The user is able to see his own attributes. In SURFconext, IdPs do not need to worry about the SP that is accessed, but only about the connection with SURFconext. This makes end-to-end encryption with XML encryption impossible, since the IdPs do not know which public key to use. So even in the case that encryption is used, this would be IdP-to-hub and hub-to-SP encryption, and the SURFconext hub still has access to the content of exchanged messages.

SURFconext also does not leave signatures intact, but signs the attributes it receives from the IdP with its own private key. This allows SPs to only care about SURFconext as IdP, and only verify whether signed data comes from SURFconext.

This means SURFconext is a man-in-the-middle, and is able to see and alter the contents of all messages.

3.1.8 Cookies

SAML requires state to be kept in the user environment, but does not specify how this should be kept. Cookies can be used for this purpose [33] by both IdPs and SPs. IdPs use cookies for SSO: a cookie is placed in order to remember the

user that logged in. This cookie typically contains a session ID. When a new authentication request is received, the user does not have to authenticate again, but an authentication response is sent immediately. SPs can also use a cookie to keep the session of the user for the duration that he remains logged in to the service.

3.2 Properties of Federated Identity Management

3.2.1 Trust

Trust is important in FIM. When an SP receives an authentication statement, authenticating a certain user, he must be able to trust that the user that is accessing the SP has indeed authenticated to the IdP, and that the user authenticated by the authentication statement is the same user as the one trying to access the SP. The SP must also be able to trust that the values in attribute statements are correct.

In a full mesh federation, this means that SPs must trust the IdPs they are connected to. In a hub-and-spoke federation, SPs must trust the hub, but also the IdPs the hub establishes a relationship with. If the IdPs sign their assertions, and the hub leaves the signatures untouched, it is possible to have a situation in which SPs only trust assertions signed by certain IdPs, and therefore don't have to trust all IdPs but only those IdPs. They still need to trust that the hub does not switch sessions, by switching assertions it receives for different users. If the hub uses its own signatures instead of the signatures from the IdP, and the hub tells the SP from which IdP the assertions originated, the SP must trust that the hub has checked the signatures from the IdP. In that case the SP can still decide to only trust assertions from certain IdPs.

Users must trust that their IdP does not initiate an authentication to an SP on their behalf without their explicit consent. They must also trust that the IdP does not share their information without their consent.

Man-in-the-browser

If the computer of the user is infected with malware, there can be the problem of a *man-in-the-browser*. If only a username and password are used, the man-in-the-browser can steal these. If external authentication tokens are used, there can still be problems. An attacker who wants to authenticate as the infected user to SP A, can wait for the user to visit another SP B, and use a man-in-the-browser to repace the authentication request from B with one from A. The user thinks he authenticates to B, but is in fact authenticating to A. The man-in-the-browser can steal the response and send it to the attacker so that he can use it to authenticate as the user to A.

When SSO is used, when a user has authenticated to SP B, at a later time the man-in-the-browser lets the user authenticate to A. Since SSO is used, and the user already authenticated to the IdP, the IdP will authenticate the user to A without user interaction. The man-in-the-browser can then access the account of the user at SP A.

To prevent this, the IdP can ask the user for confirmation that he really wants to authenticate to the SP he is authenticating to. To prevent the man-in-the-browser from changing or interacting with the confirmation process, this confirmation should happen using an external channel. This can for example be an SMS that says which SP the user is authenticating to, with a code that should be entered to confirm the authentication.

A man-in-the-browser can also tamper with other interactions between the user and the SP. For example, when the SP is a bank, and the user wants to transfer €10.00 to a friend. A man-in-the-browser can change this into a transfer of €10,000.00 to a bank account he can get the money from. The man-in-the-browser will make sure the web page still displays a transfer of €10.00 to the friend.

This can be prevented by not only confirming the authentication of a user via an external channel, but also confirming actions of the user through an external channel. In the bank example, the confirmation should show the amount and recipient of the transfer. Since this is shown through an external channel, the man-in-the-browser cannot tamper with it, and the user will see that he is in fact about to transfer €10,000.00 to the wrong account.

For eIDAS assurance level 'high', it is required that there is protection against tampering. This means that measures such as those we described are taken to prevent man-in-the-browser attacks.

3.2.2 Privacy

In identity federations, there are some privacy concerns that may be relevant. First of all, there is the possibility of *collusion*: if different service providers receive different attributes about a user, and they are able to link the attributes to the same user, the service providers can create a more extensive profile of the user by combining their knowledge. Also, the IdP often knows who the user is. In full mesh federations, and sometimes in hub-and-spoke federations, it can see which SPs the user uses. In the case of a hub-and-spoke federation, all authentication and attribute statements go through the hub. If these are not end-to-end encrypted, this makes the hub a privacy hot spot.

An SP may need to recognize a user as being the same when he uses the service multiple times. To achieve this, authentication statements can contain a persistent identifier, as described in Section 3.1.5. This is a pseudonym that is unrelatable to identifying information of the user, which is the same for a user, each time he logs onto an SP. To prevent linking of user data between SPs, this persistent identifier is different for different SPs. If the SP does not need to know whether the same user uses a service, transient identifiers can be used, which are different every time, but can be used to communicate with the IdP about the current session, for example when the user logs out.

The European Parliament has approved the General Data Protection Regulation (GDPR) [48]. Art. 5.1 of this regulation describes principles relating to the processing of personal data. One of these principles is 'data minimisation', which states that personal data shall be "adequate, relevant and limited to what is necessary in relation to the purposes for which they are processed" (Art. 5.1 (c)).

The GDPR will supersede the Data Protection Directive (DPD) [17], wich was used in [23]. Principles from data protection legislation and privacy guidelines

were applied to FIM, and architectural requirements were formulated that followed from that. Those requirements are, directly quoted from [23]:

AR1 ***Limited observability.*** *No entity shall be able to aggregate data about the usage of multiple services by users, which will keep it from being able to deduce personal interests or behavior.*

AR2 ***Limited linkability.*** *Relying parties (RPs) shall not be able to aggregate personal data used in different privacy domains. Only if it is necessary for a legitimate purpose shall two RPs processing data of a principal be able to link those data sets. Aside from unique identifiers, this concerns attributes that are identifying with high probability as well.*

An important measure is the use of pseudonyms. If the full pseudonymization of user attributes is not feasible, then at least those attributes that identify a user (almost) uniquely shall be pseudonymized. This applies, e.g., to the ubiquitous e-mail address.

AR3 ***Prevent the unauthorized aggregation of attributes by central intermediaries such as gateways, brokers, etc.*** *No actor shall be able to collect attributes beyond the specified purpose of a service and deduce personal information and behavior.*

AR4 ***Constrained linking.*** *Unidirectional links between instances of a principal in different privacy domains shall be possible either directly or mediated under control of the user or a third party.*

AR5 ***Consent handling.*** *The flow of releasing attributes should regard the processing of user consent, where explicit consent is appropriate.*

AR6 ***No supreme instance.*** *Actors managing trust roots must not have access to either attributes or transaction data.*

AR7 ***Minimize the release of attributes.*** *The identity provider, in its role as data controller of a principal's identity information, must be assured that only those attributes deemed necessary for the purpose of the service are released to the RP.*

AR8 ***Uniqueness of identification.*** *If the RP requires access control, e.g., to protect personal data, then the unambiguous identification of the principal is necessary, even if the user remains pseudonymous. Therefore, identifiers need to be immutable and non-reassignable in those cases.*

AR2 speaks of "attributes that are identifying with high probability". Not sharing any of such attributes is not feasible. For some attributes, such as e-mail addresses, it is possible to use proxy addresses as described in [24], which we look into in more detail in Section 5.1. However, it is extremely unlikely that there are no two SPs that process data that allow linking users with some probability. This makes **AR2** an impractical requirement that can never be

met. Therefore, for **AR2** we will only consider the linkability of pseudonyms, and not of all attributes.

AR6 entails that organizational roles should be seperated.

3.2.3 Usability

We have seen in Section 2.3 that the large number of passwords a user has to remember is a problem. FIM can improve the situation in this regard. Users only need one password to access multiple services. If a user wants to use a new service in a federation in which he already is registered with an IdP, he does not have to remember yet another password, but can use his existing account at the IdP.

There are however also some usability issues that may occur in FIM [4]:

- Users should be able to access SPs not only using the identity management system at home, but also from other locations. Academic publishers, for example, often verify whether someone should get access by checking whether the IP address of the user is of a subscribed organization. This system cannot be used from other locations, except when technologies like VPN are used.

- A user can have several identities, even within a single scope. Users should be able to manage their different roles. Furthermore, if a user wants to access a different service in a federation, SSO may cause him to be automaticcally logged in. This can also happen if he accesses the service in a different role, and would want to use a different identity than he used earlier. For example: a university employs students. This means a student can have a student account and an employee account. For his job related activities he may need to use the employee account, and for study related activities he may need to use his student account.

- Some information about the user may be stored at the IdP, other information at different SPs. This makes it difficult for users to manage their profile.

In the case of users managing their profile, FIM also provides opportunities. Instead of having to enter his details at every service he uses, they are shared by the IdP. Attributes may be shared that are necessary for the operation of a service, but more may be shared to allow more features and improve usability of the service. The profile of the user can be stored in one place: at the IdP, making the management of the profile easier. However, when SPs start creating their own profile for users or do not process updates from a user to their profile at the IdP we will get the privacy problems described above.

Some of the architectural requirements from Section 3.2.2 also have usability aspects. **AR4** can make it easier for users to use different services together. For **AR5** user interaction is necessary. To make this more user friendly, the party that asks for consent may display not only the names of attributes that will be shared with an SP, but also their contents. This helps the user in understanding what exactly they will be sharing. Usability can also be improved by allowing consent to be remembered. Then the user does not have to give his consent every time he accesses a service, but only on first use or periodically.

3.2.4 Conflicting goals

Some of the trust, usability and privacy goals are conflicting. This means a compromise has to be found between certain usability and privacy aspects.

AR7 asks for only releasing necessary attributes, but releasing more attributes can improve usability of the service.

Because of **AR7**, different sets of attributes are released to different parties. The specific set of attributes that will be released gives information about which service is used. To ask for consent to share a certain set of attributes, a party must know which set of attributes will be released. We should therefore assume that the party that asks for consent knows which service will be used. To achieve **AR1** this party should not know which user is giving consent. In a setup where the IdP knows who the user is, this means it cannot be the IdP that asks for consent. For **AR3** to be met, the party that asks consent must not know the content of the attributes that will be shared. This means it is not possible to show the content of the attributes by the party that asks for consent, if we also want **AR1**, **AR3** and **AR7**. The contents of attributes often identify the user, so even when we drop **AR3**, we still run into **AR1**.

However, the party asking for consent does not have to be the party that shows the content of the attributes. The consent page can include iframes for each attribute value it wants to show. These iframes load a URL from the IdP. This URL is specific for the attribute value that should be loaded. The user has authenticated to the IdP, and using the SSO mechanism, the IdP knows for which user it should return the attribute value. The response from the IdP should just be the value of the attribute, so it can be displayed as part of a consent page.

In case we want **AR1**, we do not want the specific set of attributes to give information to the IdP about which SP is used. This can be accomplished by including iframes for all attributes. The iframes for attributes that will not be released can be made invisible with CSS or a height of 0.

It should be noted that since the web page that asks for consent is under control of the party that asks for consent. This means he is able to include JavaScript that sends the attributes that are loaded from the IdP to the consent asking party.

To fully meet **AR1**, we do not want parties that know which services are used, to be able to link different sessions of the same user, even if those parties do not know who the user is. If we want to remember that a user has given consent, we need to be able to link different sessions. We have already seen that we should assume that the party that asks for consent knows which service is used. Therefore he should not be able to link different sessions, making it impossible to remember consent. If we do allow linking of sessions of the same user, visiting the same SP, but not sessions of the same user with different SPs, it can be possible to remember consent.

The IdP can be the most trusted party by the user. It may therefore not be desirable to move consent handling and decisions about which attributes are released to SPs to different parties. But if the IdP has these responsibilities, it will need to know which SPs are used by a user, and if the IdP knows who the user is, he will be violating **AR1**.

If we want to prevent the damage from a man-in-the-browser by letting the users confirm their authentication to a specific SP, as described in Section 3.2.1, the IdP must know which SP a user is authenticating to, and we cannot have **AR1**. In cases where an authenticated user mainly gets access to information, for which it is privacy sensitive that he wants to have this information, **AR1** is a useful requirement. When the SP actually lets the user perform actions, it becomes important that we can trust that it really is the user performing those actions. In those cases prevention of man-in-the-browser attacks is more important, and **AR1** is too strict. The IdP should know which SP the user is authenticating to.

Chapter 4

Current Situation

4.1 SURFconext

SURFconext requires IdPs to release the attributes `urn:mace:dir:attribute-def:uid` (in short `uid`) and `urn:mace:terena.org:attribute-def:schacHomeOrganization` (`schacHomeOrganization`) for its users. These attributes uniquely identify the user. For the services that require persistent nameID's, the first time the user accesses the SP, SURFnet randomly generates an identifier. The combination of `uid`, `schacHomeOrganization`, SP and identifier is stored in a database, so the identifier can be looked up when the user accesses the SP again.

SURFconext performs attribute filtering. This means that IdPs often release more attributes than necessary for the SP the user wants to access. SURFconext only sends those attributes to the SP that are actually needed. In some cases, SURFconext also looks at the value of the attributes to see whether they should be passed on.

In SURFconext it is possible to create teams of users of different IdPs. SPs can request team membership information by passing the persistent nameID to the teams API.

4.1.1 Privacy

We will compare SURFconext to the architectural requirements from [23], as described in Section 3.2.2. ✓ denotes an architectual requirement being met, ✗ denotes a requirement that is not met.

AR1 Limited observability. Both IdPs and the hub are able to aggregate data about the usage of multiple services by users ✗

AR2 Limited linkability. Persistent identifiers are used to limit linkability ✓

AR3 Prevent the unauthorized aggregation of attributes by central intermediaries such as gateways, brokers, etc. The hub is able to aggregate attributes ✗

AR4 Constrained linking. If SPs do need to cooperate, SURFconext can give them the same persistent identifier for a user, allowing them to communicate about the user ✓

AR5 Consent handling. Either SURFconext asks user consent before sharing attributes with an SP, or user consent is handled by the IdP ✓

AR6 No supreme instance. The hub is a supreme instance ✗

AR7 Minimize the release of attributes. SURFconext only shares those attributes with SPs that are necessary for the purpose of the service of the SPs ✓

AR8 Uniqueness of identification. Assuming IdPs use unique and constant user identifiers, the persistent identifiers assigned by SURFconext are immutable and non-reassignable ✓

4.2 Kennisnet

In Dutch education there is a collaboration between schools, distributors and publishers. Schools purchase learning materials for their pupils from distributors. These materials are created by publishers. If the school purchases electronic learning materials, pupils need to get access to this at the publisher. The distributor has to communicate to the publisher that the pupil has a licence for the product. The publisher has to communicate to the school about progress and results of the pupil. These communications are standardised by Edustandaard [18].

Kennisnet provides ICT infrastructure to Dutch primary, secondary and vocational education [32]. This includes an identity federation, called Entree [31], which is a hub-and-spoke federation. The virtual learning environment of the schools provide the IdP functionality. There are also privately developed federated systems, like Basispoort and Edu-iX. The different parties, like schools, distributors and publishers have several ways to communicate with each other. Sometimes the Entree federation is used, sometimes communication goes via the mentioned private systems, and sometimes communication goes directly.

The parties need some way to identify which pupil they are communicating about. This is called the matching problem. Currently, personal information about the pupil is used to perform the matching of a user at different parties to the same person. A privacy impact assessment (PIA) has been performed of the plans to improve the situation in Dutch education [22]. This PIA also lists some problems with the current situation:

- directly identifying data is spread through the education chain.
- distributors and publishers have a relatively independent position, which may allow them to perform data processing outside of the view of the schools, or which the schools are not fully aware about.
- distributors and publishers can use the data they receive in the context of their role in the education chain for their own business and use it to approach pupils or their parents for commercial purposes.
- there is strict regulation of the use of numbers that are assigned by law to persons in order to identify them.

4.2.1 Current plans

The mentioned PIA [22] was performed to assess the plans to improve on the described situation. These plans are also detailed in the PIA, and the following description of the plans is based on the description in the PIA. There are two solutions that together should solve the described problems. The first is a numbering facility with pseudonyms, and the second is a privacy covenant.

Numbering facility with pseudonyms

The numbering facility was in the first place devised to solve the matching problem. Instead of exchanging datasets to identify pupils, a so called chain pseudonym ('ketenpseudoniem') is used, which uniquely and unambiguously identifies pupils. A result of this is that less personal information is exchanged, and distributors and publishers obtain less personally identifying information. A consequence thereof is that distributors and publishers have fewer options to use the information they receive in the context of their role in the education chain for their own business and use it to approach pupils or their parents for commercial purposes.

Chain pseudonyms are created by the so called numbering facility. Schools request a chain pseudonym for their pupils once, and store that in their administration system. When a school requests a chain pseudonym, it sends a hash of the PGN (persoonsgebonden nummer), which for most pupils is their social security number. The numbering facility bases the pseudonym on three attributes: the hashed PGN, the education sector(primary, secondary or vocational education) and an identifier of the chain. The described system of schools, distributors and publishers is seen as one chain, but the design of the numbering facility allows differentiation within this system. The design also allows the numbering facility to be used for other uses. The pseudonym will then be different for the same pupil. Although the design allows differentiation of chains, currently everything is considered to be the same chain. The chain pseudonym is not based on the school of the pupil, so if a pupil moves to a different school in the same sector, the pseudonym will stay the same.

Privacy Covenant

The privacy covenant deals with the independent position of the distributors and publishers. The covenant is not meant to reduce the amount of personal data that is shared, but to give schools more control on how the data are used. The distributors and publishers are no longer responsible parties of the personal data, but processors, while the school is the responsible party.

4.2.2 Privacy

For the Kennisnet use case, we will compare the current plans to the architectural requirements from [23]. ✓ denotes an architectual requirement being met, ✗ denotes a requirement that is not met. ✗/✓ denotes a requirement that is not met, but for which it is not a problem.

AR1 Limited observability. Both IdPs and the gateways of the Entree and private federations are able to aggregate data about the usage of multiple

services by users. These SPs are distributors and publishers, and their usage is not very privacy sensitive. So there is not a real problem here ✗/✓

AR2 Limited linkability. The same pseudonym is used for a pupil across the entire sector (primary, secondary or vocational education) ✗

AR3 Prevent the unauthorized aggregation of attributes by central intermediaries such as gateways, brokers, etc. The gateways of the Entree and private federations are able to aggregate attributes ✗

AR4 Constrained linking. Since every party in a sector gets the same pseudonym for a pupil, linking is easy ✓

AR5 Consent handling. IdPs are required to handle user consent ✓

AR6 No supreme instance. The hub is a supreme instance ✗

AR7 Minimize the release of attributes. Kennisnet does not state clearly that attribute release is minimized ✗

AR8 Uniqueness of identification. Pseudonyms assigned by the numbering facility are immutable and non-reassignable ✓

Moreover, since the same pseudonym is used for a user across the entire sector, parties can combine data. The data minimisation principle as stipulated in Article 5 ofthe GDPR [48] is not followed.

4.3 Other developments in the Netherlands

We consider two other systems for identity federation that are being developed in the Netherlands: Idensys, a collaboration between Dutch government and private companies, and iDIN, which is developed by Dutch banks.

4.3.1 Idensys

Idensys pursues enabling the online identity and authorization of a person to be determined in an optimal way, such that people can trust online services from both government and business [34]. One standard is being developed for authentication and authorization for public and private organizations. Consumers, citizens and entrepreneurs can choose the token with which they authenticate to organizations, and organizations get more assurance about the identity of the user [47].

The interface specifications [25] of Idensys describes several roles. Their relation can be seen in Fig. 4.1. These roles are:

dienstverlener (DV) (service provider) A party that provides a service for which authentication is needed.

herkenningsmakelaar (HM) (broker) A single point of contact through which DVs use authentication services. It is the same as the hub in a hub-and-spoke federation. It acts as a router for all participating parties.

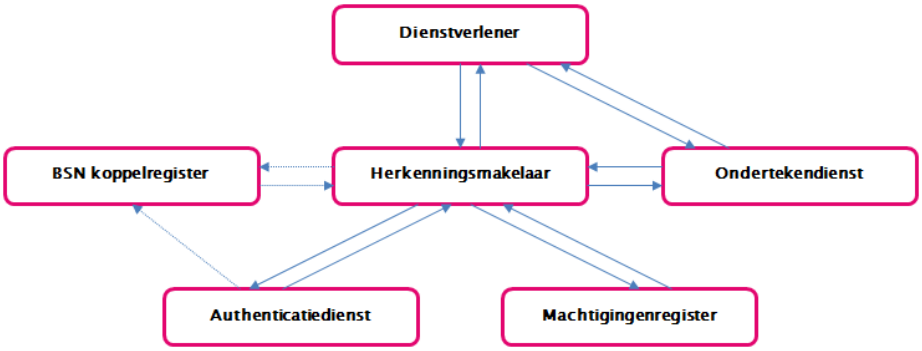


Figure 4.1: The roles and their relation in Idensys [25]

ondertekendienst (OD) (signature service) A party that facilitates the legal binding of volitions. It inextricably links a document to assertions about the identity of users, forming a digital signature.

koppelregister (KR) (linking register) A party that links an identifying attribute to a pseudonym used by a person. An example of this is the BSN koppelregister, which links BSNs to pseudonyms.

authenticatiedienst (AD) (identity provider) Authenticates natural persons based on the authentication token used by that person.

machtigingenregister (MR) (authorization information provider) Party that is responsible for registering, managing and checking authorizations, and make statements about these authorizations.

4.3.2 iDIN

Dutch banks have developed iDIN, an identity federation in which consumers can authenticate to merchants through their banks, and share attributes [13]. It is based on iDEAL, which allows consumers to make online payments to webshops using their own bank [14]. The bank of the consumer (issuer) acts as IdP. The merchant is the SP, but also has a bank (acquirer) which has a role in the iDIN protocols. A merchant may choose to outsource the handling of iDIN activities to a digital identity service provider (DISP). A schematic overview of these parties and their relations can be seen in Fig. 4.2.

- All iDIN transactions are initiated by a consumer on the website of a merchant.
- Consumers will be identified with a bank identification number (BIN). This is a persistent identifier that is unique for a consumer-merchant combination, and generated by the issuer.
- Information about a consumer will be end-to-end encrypted, so the acquirer cannot see the contents.
- Consumers have one identity per issuer, independent of the number of credentials.

- iDIN uses the iDx protocols, which uses SAML messages.

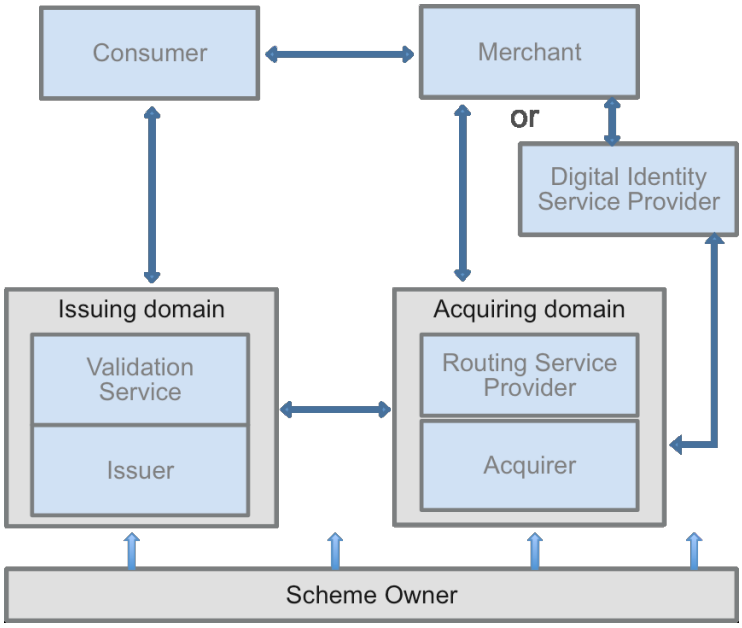


Figure 4.2: The four corner model of iDIN [13]

Chapter 5

Related work

5.1 Privacy Enhanced Federated Identity Managment

The privacy enhanced federated identity management (PE-FIM) model described by [24] is a way to improve privacy in identity federations. The high level architecture is shown in Fig. 5.1. It specifically solves **AR1**. A blind proxy, called the service broker, is used, which sets up a pseudonymous channel between IdP and SP. This channel is end-to-end encrypted, so the service broker cannot see the contents of exchanged messages. The IdP must be able to encrypt messages, while not knowing which SP will receive the messages and not being able to link different sessions with the same SP. Therefore for each session the SP generates a one-time key, which is signed by a certificate authority (CA). The certificate generated by the CA does not contain the name of the SP, so the IdP can use the key from this certificate to encrypt his messages, without knowing the SP for which he is encrypting them. The SP does know from which IdP a user comes, so he can use the public key from the IdP to encrypt his messages.

SPs with similar transaction volume, requested attributes and bindings are

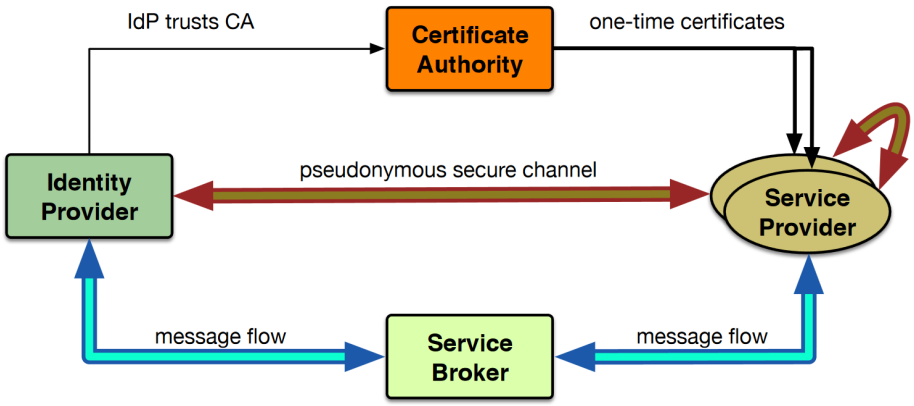


Figure 5.1: Schematic of the PE-FIM model [24]

grouped, and the IdP does learn the group the SP belongs to. Without grouping them, the IdP could still infer which SP is used by these properties.

In order to meet **AR1**, in PE-FIM consent is managed and stored by the service broker.

Targeted identifiers are used to avoid linkability between SPs. For identifying attributes like e-mail addresses, payment information and physical addresses, proxy services can be made available, in order to replace these idenitifying attributes with proxy addresses. In cases where SPs do need to communicate about a user, the service broker can provide identifiers for a user that are shared between these SPs.

We will compare this model to the architectural requirements from [23], as described in Section 3.2.2. ✓ denotes an architectual requirement being met, ✗ denotes a requirement that is not met.

AR1 Limited observability. The IdP does not learn which SPs are used by its users, by using a pseudonymous channel, as described. Other parties also do not learn anything about SP usage by certain users ✓

AR2 Limited linkability. Targeted identifiers and proxy addresses are used to avoid linkability ✓

AR3 Prevent the unauthorized aggregation of attributes by central intermediaries such as gateways, brokers, etc. Intermediaries cannot access attributes because they are end-to-end encrypted ✓

AR4 Constrained linking. PE-FIM describes shared identifiers in order to allow constrained linking, controlled by the service broker ✓

AR5 Consent handling. Consent is handled at the service broker. This does mean the consent UI cannot display identifying data ✓

AR6 No supreme instance. The CA should be organizationally seperated from other parties, such that it cannot become a supreme instance ✓

AR7 Minimize the release of attributes. Attribute release can be based on SP groups. It can be further limited by the service broker ✓

AR8 Uniqueness of identification. Targeted identifiers ensure uniqueness of identification ✓

We have seen in Section 3.2.4 that **AR1** conflicts with other properties we may want. In many cases it is desirable that the IdP shows to the user which SP he is authenticating to, to prevent attackers from letting the user authenticate to an SP he does not want to use. This is however not possible if we have **AR1**.

5.2 IRMA

A different approach to identity management is taken by the IRMA project [28]. IRMA, short for I Reveal My Attributes uses attribute based credentials (ABC) [3, 5], and is an implementation of the Idemix system [9]. The project originally focussed on implementing ABC efficiently on smart cards, but a phone application was later also developed [27].

ABC leave out the IdP. The user only discloses those attributes to an SP that it needs, without any extra identifying information. Instead of authenticating the identity of the user, only the attributes are authenticated, by the issuer of the attributes. This makes sure that a user does not need to identify himself, but the SP can still trust the attributes.

IRMA is an example of authentication with a TTP only for token issuance, as described in Section 2.5.2. A TTP gives out attributes to users, and the users can then use these attributes with SPs without a TTP. There are multiple parties that can issue attributes.

Release of identifying information can be reduced by using the right attributes. For example: an attribute stating that the user is at least 18 years old may be enough, and no date of birth is necessary. An attribute that states that the user has a subscription to a certain service may be enough and no membership number is necessary.

We will compare this model to the architectural requirements from [23], as described in Section 3.2.2. ✓ denotes an architectual requirement being met, ✗ denotes a requirement that is not met.

AR1 Limited observability. There is no IdP that can observe SP usage. Attributes are issued independent from their usage at SPs. Attribute issuers can therefore also not observe SP usage ✓

AR2 Limited linkability. When identifying attributes are released to an SP, these can be used to link users across SPs. Using the right attributes limits the use of identifying attributes ✓

AR3 Prevent the unauthorized aggregation of attributes by central intermediaries such as gateways, brokers, etc. There are no intermediaries ✓

AR4 Constrained linking. Giving services access to the same attribute can accomodate linking ✓

AR5 Consent handling. A PIN can be used to require user consent before attributes are released. This does not ensure that the user knows which attributes will be released. It is up to the SPs to show this information. Users can check afterwards whether indeed the right attributes were released by checking a log of their IRMA card ✓

AR6 No supreme instance. The scheme manager must be an independent organization ✓

AR7 Minimize the release of attributes. SPs should ask for only those attributes they need. The user can decide whether he wants to share the requested attributes ✓

AR8 Uniqueness of identification. It is possible to store identifiers as attributes. Uniqueness of these idenitifiers is the responsibility of the issuer ✓

Chapter 6

Polymorphic Pseudonyms

Polymorphic pseudonyms [54] provide a different approach to user identifiers. We will first introduce three forms of pseudonyms that will be used. We will then describe the basic model to show how these pseudonym forms are used.

polymorphic pseudonym (PP) The IdP can create this form of pseudonym for every user. It must be transformed into an EP in order to be used to access a service.

encrypted pseudonym (EP) A polymorphic pseudonym (PP) can be specialized for use with a specific SP. The resulting encrypted pseudonym (EP) is an encrypted, targeted ID that can only be used for that SP.

final pseudonym (FP) An SP can decrypt an EP to get the final pseudonym (FP). This pseudonym is the same for a user each time he uses a specific service, but is different for the same user at different SPs. It is similar to a persistent NameID in SAML, but it is only known to the SP.

6.1 Basic Model

The basic model of polymorphic pseudonyms works in a full mesh federation, so IdPs and SPs communicate directly, without intervention by a central hub. We need two additional parties:

pseudonym facility (PF) PPs are specialized into EPs by the pseudonym facility (PF).

key management authority (KMA) Each party needs certain keys. These parties include the PF, IdPs, and SPs. These keys are generated and distributed by the key management authority (KMA). It is not active during normal operation, only at the start, and when a new IdP or SP is connected to the system.

Fig. 6.1 gives a schematic overview of this model. The IdP generates a PP for its users. When a user (e.g. $\texttt{User}_{11}$) wants to use a service (e.g. $\texttt{SP}_2$), these steps are taken:

1. The user logs into his IdP ($\texttt{IdP}_1$).

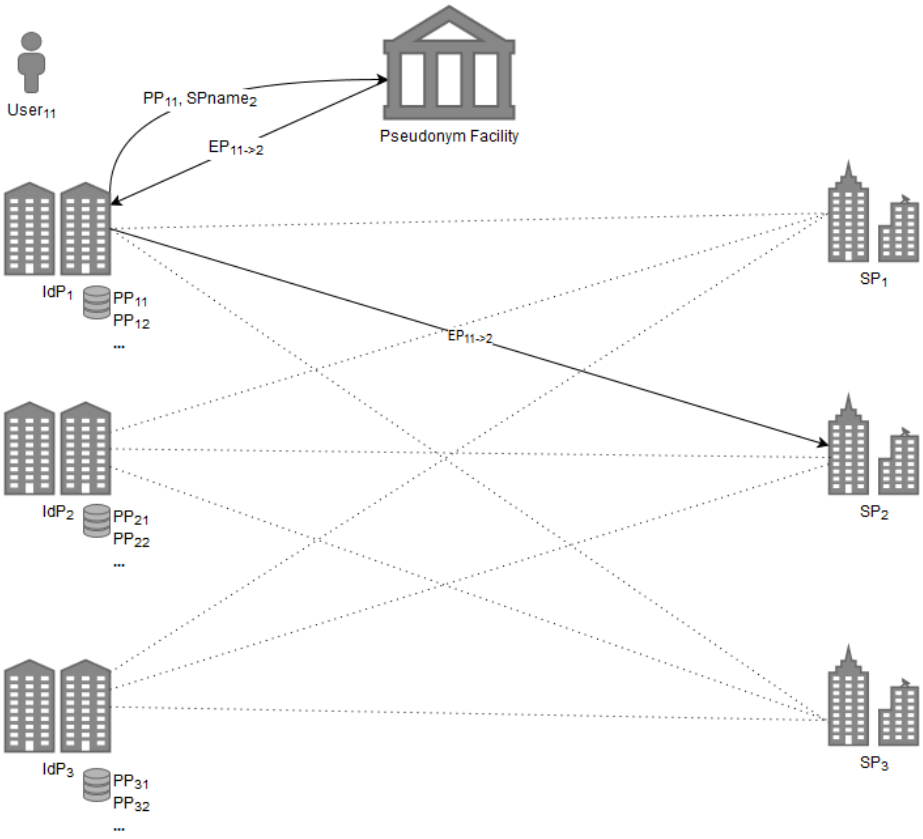


Figure 6.1: The basic polymorphic pseudonym model

2. The IdP generates the PP for the user ($\texttt{PP}_{11}$), or looks up a previously generated one.
3. The IdP contacts the PF, passing the PP for the user, and the name of the SP the user wants to use ($\texttt{EntityID}_2$).
4. The PF will transform the PP into the EP for the user, for the specified SP ($\texttt{EP}_{11\to2}$), and return this to the IdP.
5. The IdP places this EP in the authentication statement it sends to the SP.
6. The SP decrypts the EP, and gets the FP for the user at that SP.

6.2 Properties

The different types of pseudonyms have some interesting properties:

- PP and EP look different each time they are generated. This means that no information can be retrieved from only a PP or an EP. It is not possible to see from the PP or EP that different sessions are for the same user.
- PP and EP can be randomized. This means that a previously generated PP or EP can be changed, such that it looks different and is unlinkable to the first version of the PP or EP. No keys or extra information are needed to perform randomization. Both PP and EP can therefore be stored, and reused, without making the reuse linkable to earlier usages, as long as they are randomized before reuse.
- The FP is the same every time for the same user at the same service provider.
- The FP is different for different users.
- The FP is different for different service providers.
- Only the SP for which an EP was created can decrypt it into the correct FP.
- To create the PP, a globally unique user ID is needed. This can be something like `username@example.com`, for IdP `example.com`.
- If a user ID is used that does not depend on the IdP, like a social security number, the user can use different IdPs and still be identified as the same user by SPs.
- All three pseudonym forms do not reveal information about the user ID that is used. This means that if social security numbers are used, only the IdP is processing the social security number.

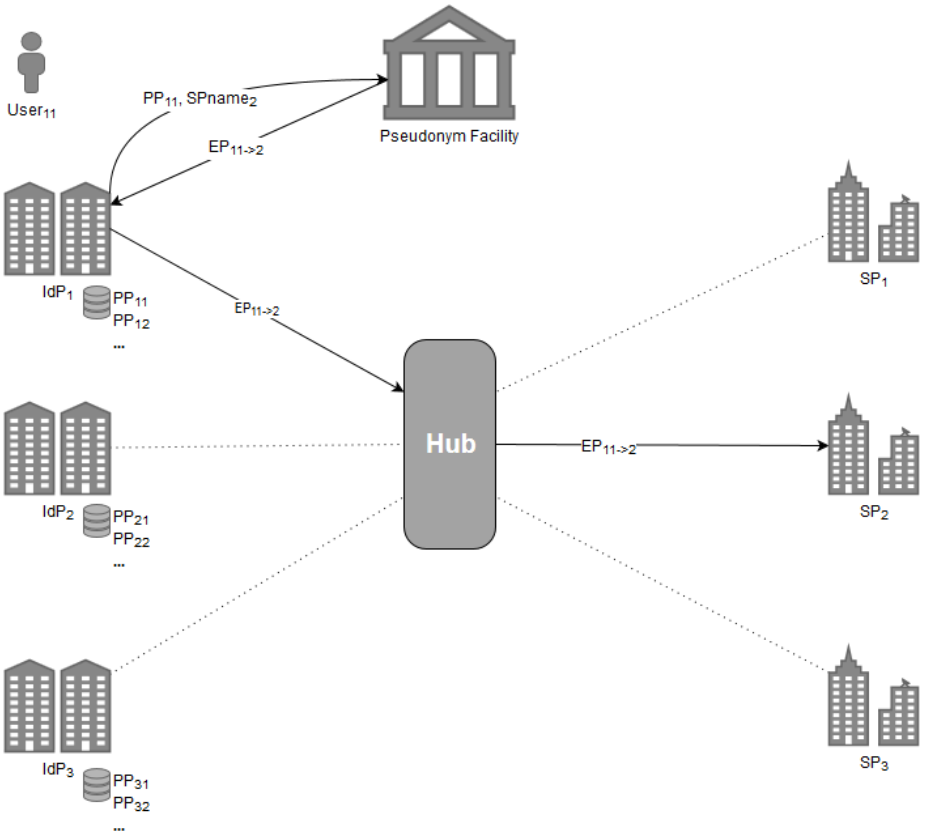


Figure 6.2: Polymorphic pseudonyms in a hub-and-spoke federation

6.3 Use Cases

6.3.1 Full mesh

We have already seen how polymorphic pseudonyms can be used in a full mesh federation (Fig. 6.1).

6.3.2 Hub-and-spoke

In a hub-and-spoke federation we can use polymorphic pseudonyms in almost the same way as in full mesh federations. The hub then just passes on what it receives, as shown in Fig. 6.2.

It is also possible to let the hub perform the task of the PF (Fig. 6.3). In this case, the IdP sends the PP to the hub, and the hub specializes it into an EP for the SP the user wants to use. The IdP does not need to worry about which SP is used, and does not even have to know about this. The hub only sees PPs and EPs. This does not tell him anything about the user, protecting the privacy of the user. An important requirement for this to work is that the hub may not be a service provider itself. If it is, the hub can specialize the PPs it receives for himself, decrypt the resulting EP into the FP which will always

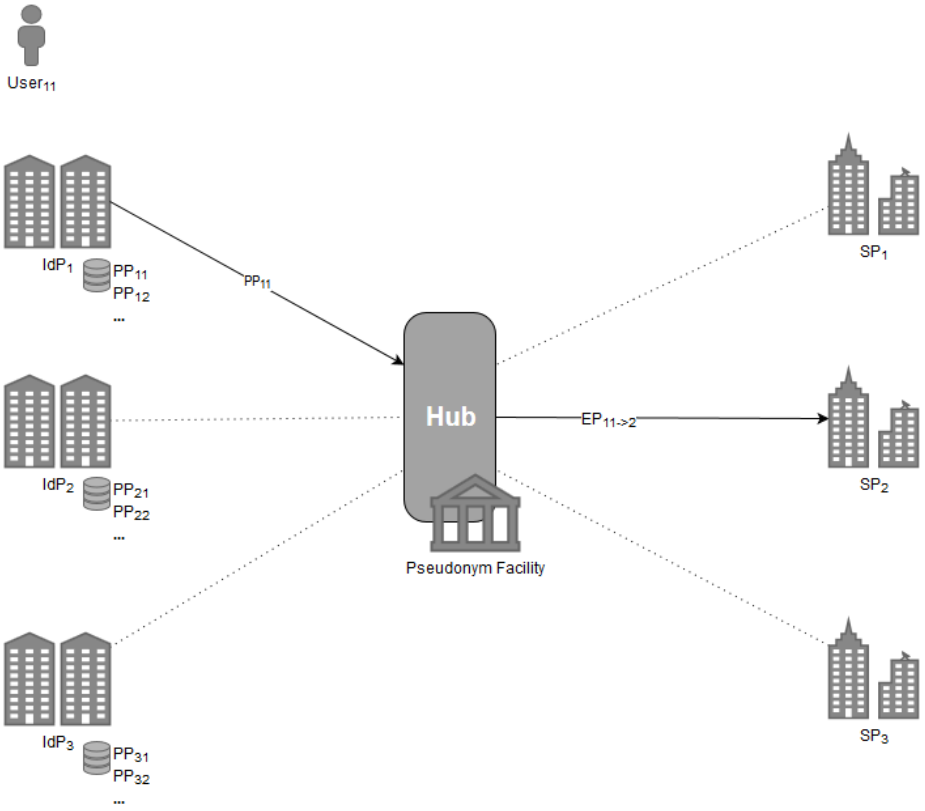


Figure 6.3: Hub-and-spoke federation with the pseudonym facility at the hub

be the same for the same user. The hub can then see which sessions are from the same user.

This way, polymorphic pseudonyms provide a different approach to achieve **AR1** than [24], which we have described in Section 5.1. With polymorphic pseudonyms we do not need a CA to give out one-time certificates for each session. We do need a KMA, but it is only used when new parties are connected to the federation. The hub does need to perform more work than in [24], since it must specialize the PPs.

Chapter 7

Cryptography of Polymorphic Pseudonyms

7.1 ElGamal

Polymorphic pseudonyms are based on the ElGamal public key cryptosystem [19]. ElGamal encryptions are performed over a multiplicative group G, of prime order q, with generator g. The Decision Diffie-Hellman (DDH) problem must be hard in G. This means that there is no efficient algorithm for distinguishing (g^a, g^b, g^{ab}) from (g^a, g^b, g^c) for random $a, b, c \in G$ [7].

The DDH assumption is stronger than the Computational Diffie-Hellman (CDH) assumption, which states that the Diffie-Helman function $DH_g(g^a, g^b) = g^{ab}$ is hard to compute. Indeed, if we have some (g^a, g^b, g^c), and we can compute $DH_g(g^a, g^b)$, than we can easily check whether $g^c = DH_g(g^a, g^b)$, and thus whether $(g^a, g^b, g^c) = (g^a, g^b, g^{ab})$.

The CDH assumption in turn is stronger than the Discrete Logarithm (DL) assumption, which states that it is hard to compute a for a given g^a. We can easily see this is the case, since if we were able to compute (a, b) for given (g^a, g^b), we can compute $g^{ab} = DH_g(g^a, g^b)$.

In this document 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. For practical implementations one can use a group of points G on an elliptic curve. There are standard curves that can be used, such as the Brainpool curves [37] and the NIST curves [41].

An ElGamal key pair consists of a random $x \in GF(q)$, and $y = g^x \in G$, where x is the private key, and y the public key. To encrypt a message m, with public key y, one takes a random $k \in GF(q)$, and calculates the tuple $(g^k, y^k \cdot m)$.

For polymorphic pseudonyms, we will not use these tuples, but triples which also include the public key that was used. An ElGamal encryption of message m, with public key y, and random k is then:

$$\mathcal{EG}(m, y, k) = (g^k, y^k \cdot m, y) \tag{7.1}$$

ElGamal is a partially homomorphic encryption scheme. This means that we

can perform certain operations on encrypted data, without having to decrypt it, or having to use the decryption key in any way. If we have an ElGamal encryption $\mathcal{EG}(m, y, k) = (A, B, C)$, we can perform these operations [54]:

$$(A^z, B^z, C) = \mathcal{EG}(m^z, y, k \cdot z) \tag{7.2}$$

$$(A^z, B, C^{(z^{-1})}) = \mathcal{EG}(m, y^{(z^{-1})}, k \cdot z) \tag{7.3}$$

$$(A \cdot g^z, B \cdot C^z, C) = \mathcal{EG}(m, y, k + z) \tag{7.4}$$

Equation 7.2 allows us to change the encrypted message. Equation 7.3 allows us to change the public key a message is encrypted with. Finally, equation 7.4 allows us to randomize an encrypted message: a different random value is used, so it looks different from the outside, but the decryption will stay the same.

7.2 Other cryptographic primitives

For polymorphic pseudonyms, we need a few more primitives. When encrypting data using ElGamal, the data should be represented as an element of G. For elliptic curve ElGamal, that means it should be represented as a point on the curve. For polymorphic pseudonyms we do not need to get the original data back, so a one way function mapping data to elements of G is sufficient. So a hash function $\mathcal{I} : \{0,1\}^* \to G$ is needed, that maps bit arrays into the multiplicative group G used for the ElGamal encryptions. Secondly, a key derivation function $\mathcal{M} : \{0,1\}^n \times \{0,1\}^* \to GF(q)$ is needed, which maps an n bit key and a bit array to $GF(q)$. This function is used to derive private keys and secret exponents from the name of an SP. The key that is used by this function makes sure that the derived values will be unkown to other parties.

7.3 System Setup

The involved parties first have to agree on the choices for the primitives that are used in the system. They need to agree on a multiplicative group G, a generator g of this group, a secure hash function $\mathcal{I}$ and a key derivation function $\mathcal{M}$.

Then, several keys have to be generated, at the KMA. A public key pair is generated, with private key $x_K \in_R GF(q)$ and public key $y_K = g^{x_K}$. Here $x \in_R S$ denotes an element x that is taken randomly from set S. The KMA furthermore chooses a key derivation function (KDF) key D_K at random. We call this key D_K the *ElGamal master key.*

All registered parties, so all SPs and IdPs need to be associated securely with a unique name $\mathcal{N}$. The KMA generates a private key for these parties, derived from the system private key x_K. The key D_K ensures that the derived private keys remain secret to other parties.

$$x_{\mathcal{N}} = \frac{x_K}{\mathcal{M}(D_K, \mathcal{N})} \tag{7.5}$$

This means that for the public key the following will hold:

$$y_{\mathcal{N}} = y_K^{(\mathcal{M}(D_K, \mathcal{N})^{-1})} \tag{7.6}$$

Each registered party chooses a random *pseudonymization closing key* $c_{\mathcal{N}} \in GF(q)$. This key is generated by the party itself, and is therefore out of the control of the KMA. The closing key ensures that the pseudonym facility cannot generate FPs for known user IDs, and compare them to FPs it learned in some way. Furthermore, it ensures that the pseudonym facility cannot transform FPs in FPs for different SPs.

The pseudonym facility (PF) chooses a KDF key D_P at random. This key is called the *pseudonymization master key*. Parties that do not have D_P cannot generate FPs or EPs by themselves. Parties can therefore not compare FPs they know to FPs they generated themselves for known user IDs.

The system public key y_K is reliably provided to the IdPs. The private key $x_{\mathcal{N}}$ for all parties is securely distributed to the party it belongs to. The parties already have the closing key $c_{\mathcal{N}}$. The ElGamal master key D_K is securely distributed to the PF. The PF already has pseudonymization master key D_P.

7.4 Pseudonym Generation

To generate a polymorphic pseudonym (PP) for a user, a unique user ID u is needed. The PP for a user is then an ElGamal encryption of the hashed user ID with the system public key y_K:

$$PP = \mathcal{EG}(\mathcal{I}(u), y_K, k) = (g^k, y_K^k \cdot \mathcal{I}(u), y_K) \tag{7.7}$$

for some $k \in_R GF(q)$

7.5 Specialization

The PF can specialize the PP to form an EP. This is done in three steps. In the first step the content of the encrypted message is changed, using equation 7.2. Let the PP be: $PP = \mathcal{EG}(\mathcal{I}(u), y_K, k) = (A_1, A_2, A_3)$. We then calculate:

$$(B_1, B_2, B_3) = (A_1^{\mathcal{M}(D_P,\mathcal{N})}, A_2^{\mathcal{M}(D_P,\mathcal{N})}, A_3) \tag{7.8}$$

Then we have, with $k' = k \cdot \mathcal{M}(D_P, \mathcal{N})$:

$$(B_1, B_2, B_3) = \mathcal{EG}(\mathcal{I}(u)^{\mathcal{M}(D_P,\mathcal{N})}, y_K, k') \tag{7.9}$$

In the second step, we change the public key of the encryption to that of the party the EP is for (equation 7.3):

$$(C_1, C_2, C_3) = (B_1^{\mathcal{M}(D_K,\mathcal{N})}, B_2, B_3^{(\mathcal{M}(D_P,\mathcal{N})^{-1})}) \tag{7.10}$$

Because of how the private keys for the parties are derived, we now have, by equation 7.6, with $k'' = k' \cdot \mathcal{M}(D_K, \mathcal{N})$:

$$\begin{aligned}(C_1, C_2, C_3) &= \mathcal{EG}(\mathcal{I}(u)^{\mathcal{M}(D_P,\mathcal{N})}, y_K^{(\mathcal{M}(D_K,\mathcal{N})^{-1})}, k'') \\ &= \mathcal{EG}(\mathcal{I}(u)^{\mathcal{M}(D_P,\mathcal{N})}, y_{\mathcal{N}}, k'')\end{aligned} \tag{7.11}$$

The final step is a randomization, using equation 7.4, with $l \in_R GF(q)$:

$$(D_1, D_2, D_3) = (C_1 \cdot g^l, C_2 \cdot C_3^l, C_3) \tag{7.12}$$

We now have

$$(D_1, D_2, D_3) = \mathcal{EG}(\mathcal{I}(u)^{\mathcal{M}(D_P,\mathcal{N})}, y_{\mathcal{N}}, k'' + l) \tag{7.13}$$

This is the EP.

7.6 Decryption

The receiving party first performs a normal ElGamal decryption. If the received EP is $(D_1, D_2, D_3) = \mathcal{EG}(\mathcal{I}(u)^{\mathcal{M}(D_P,\mathcal{N})}, y_{\mathcal{N}}, k)$, for some k, the party calculates:

$$E = D_2 / D_1^{x_{\mathcal{N}}} \tag{7.14}$$

We now have:

$$E = \mathcal{I}(u)^{\mathcal{M}(D_P,\mathcal{N})} \tag{7.15}$$

The party uses his pseudonimization closing key $c_{\mathcal{N}}$:

$$F = E^{c_{\mathcal{N}}} \tag{7.16}$$

To get the final pseudonym this is hashed with a secure hash function $\mathcal{H}$:

$$FP = \mathcal{H}(F) \tag{7.17}$$

We now have:

$$FP = \mathcal{H}(\mathcal{I}(u)^{\mathcal{M}(D_P,\mathcal{N})\cdot c_{\mathcal{N}}}) \tag{7.18}$$

If we only had $\mathcal{I}(u)^{\mathcal{M}(D_P,\mathcal{N})}$ as the FP, the pseudonym facility (PF) would be able to calculate the following:

$$FP^{-\,\mathcal{M}(D_P,\mathcal{N})} = \mathcal{I}(u) \tag{7.19}$$

This can be used to relate FPs to known user IDs. The pseudonym facility could also perform this step:

$$\mathcal{I}(u)^{\mathcal{M}(D_P,\mathcal{N}')} \tag{7.20}$$

This means the pseudonym facility would be able to transform FPs from one SP into FPs from another SP.

This manipulation by the pseudonym facility is prevented by using a hash function. But if only a hash function would be used, the pseudonym facility would still be able to calculate FPs for known user IDs, and thus learn how a user is identified at an SP. The closing key prevents this.

7.7 Randomization

Based on equation 7.4, we can randomize PPs and EPs:

$$(B_1, B_2, B_3) = (A_1 \cdot g^l, A_2 \cdot A_3^l, A_3) \tag{7.21}$$

where $l \in_R GF(q)$.

We have already seen this in the final step of specialization in equation 7.12. This operation can, however, be applied in the same way to PPs and previously specialized and randomized EPs.

7.8 Key Compromise

There are five different types of keys that must remain secret. For different keys, the consequences are different when they do become compromised.

7.8.1 KMA

Keys: x_K, D_K

PPs are ElGamal encryptions with the public key belonging to private key x_K. When x_K is compromised, an attacker can therefore decrypt PPs. This will give him $\mathcal{I}(u)$. This allows him to link sessions of the same user, or to check whether they belong to a user with known user ID u.

Private keys $x_\mathcal{N}$ for SPs are derived from x_K. Replacing x_K therefore entails also replacing all private keys for SPs. FPs do not change, since they do not depend on the public key pairs that were used, as can be seen in Equation 7.18.

When D_K is compromised, an attacker can perform the step from Equation 7.10. This means that he can change the encryption of PPs into one with the public key of a chosen SP. If that is an SP he controls himself, this means he can effectively decrypt PPs, giving him $\mathcal{I}(u)$, just as for a compromise of x_K.

Private keys $x_\mathcal{N}$ for SPs are also derived from D_K. Replacing D_K therefore entails also replacing all private keys for SPs. FPs do not change, since they do not depend on D_K, as can be seen in Equation 7.18. When D_K is replaced, it should also be transported to the PF.

A compromise of the KMA may leak both x_K and D_K. An attacker having both keys can derive private keys for any SP, using Equation 7.5. He can use this to decrypt EPs. This allows him to link different randomizations of an EP for the same user. It does not give hime the FP, since this also depends on the closing key $c_\mathcal{N}$.

7.8.2 PF

Keys: D_P, D_K

When an SP decrypts an EP it receives, it has $\mathcal{I}(u)^{\mathcal{M}(D_P,\mathcal{N})}$ (Equation 7.15). When D_P is compromised, and is known to an SP, it can get $\mathcal{I}(u)$. This allows him to link a pseudonym to a user with a known u. If u is a social security number all possible values for u can be checked.

When D_P is replaced, no other keys need to be replaced. However, since the FP depends on it (Equation 7.18), all FPs will change, and SPs cannot link the new pseudonyms of users with the old ones.

We have seen in the previous section what the consequences are for leaking D_K. When both D_P and D_K are compromised, an attacker can specialize PPs for any SP.

7.8.3 SP

Keys: $c_\mathcal{N}$, $x_\mathcal{N}$

When the closing key from an SP is known to the pseudonym facility, the pseudonym facility can create FPs for any user with known u for the SP, and see whether leaked FPs belong to users with known u.

When an SP wants to renew $c_{\mathcal{N}}$, this will change the FPs. The SP can use the deprecated $c_{\mathcal{N}}$ next to the new one for a period of time, to relate old FPs with new FPs.

When $x_{\mathcal{N}}$ is compromised, an attacker can decrypt EPs that were targeted at the SP. This gives the attacker $\mathcal{I}(u)^{\mathcal{M}(D_P,\mathcal{N})}$.

$x_{\mathcal{N}}$ is derived from x_K, D_K and $\mathcal{N}$. To be able to replace $x_{\mathcal{N}}$ we can append a version number to $\mathcal{N}$. The PF must know which version to use. When a new $x_{\mathcal{N}}$ is needed, the KMA can generate a new key with the new version appended to $\mathcal{N}$. The pseudonym facility must start to use the new version number from then on. The FPs will not change.

7.8.4 IdP

The IdP only uses public key y_K, so there is no risk of key compromise here.

Chapter 8

Design Considerations

When deciding whether polymorphic pseudonyms are a good solution, and how exactly they are to be used, there are some considerations that come into play. These considerations are different for SURFconext and Kennisnet.

8.1 Dropping the database

This consideration is only applicable to SURFconext. Right now, SURFconext has a database that maps users from the connected organizations to pseudonyms for the SPs. They are interested in a solution where they do not need such a database.

8.2 Configuration of connected parties

For polymorphic pseudonyms to work, some changes have to be made to the IdPs and SPs. The software that is used by these parties should allow these changes. Furthermore, it should not take too much effort to make the required changes. In the Kennisnet use case, plans are already made to introduce a numbering facility, which requires changes for the connected parties. Since changes already need to be made, it is less of a problem to also make changes related to polymorphic pseudonyms. For SURFnet it is more difficult to get connected parties on board, and therefore the configuration should be as easy as possible.

8.3 Software maintenance

The products that are currently used by the IdPs and SPs are mostly not developed and maintained by SURFnet and Kennisnet. If polymorphic pseudonyms are to be used, it is not to be expected that these products will start to support this, and therefore software may need to be developed by SURFnet and Kennisnet. This also brings the extra burden of maintaining this software. This has to be taken into consideration when decisions about the use of polymorphic pseudonyms are made.

8.4 Independent facilities

For polymorphic pseudonyms a pseudonym facility (PF) is needed, as wel as a key management authority (KMA). These should at least be logically seperated from the operations performed by SURFconext and Kennisnet. Organizational separation provides more guarantees about the privacy and security of the system. It has to be decided whether this is feasible. For the Kennisnet use case a numbering facility is already being set up as an independent facility. The functionality of the numbering facility can be adapted to the use of polymorphic pseudonyms.

8.5 Consent

In the current situation, both in the Entree federation and in SURFconext, the hub can ask the user for consent to share certain attributes with an SP. Consent can be remembered, such that the user does not have to give consent every time he visits the same SP. This functionality should still be possible when polymorphic pseudonyms are used.

Chapter 9

Implementation

An implementation of polymorphic pseudonyms was created. There are several roles which had to be implemented: PF, KMA, hub, IdP and SP. In a federation, there are several IdPs and SPs, that are not all using the same software. We want to know how difficult it is for IdPs and SPs to start using polymorphic pseudonyms. Therefore plugins were created for several often used systems:

- SimpleSAMLphp, which works both as IdP and SP, written in PHP.
- Shibboleth SP, which is SP software written in C++.
- Shibboleth IdP, which is the IdP counterpart of Shibboleth SP, written in Java.
- Microsoft AD FS, a closed source product, which can act as IdP. Plugins are written using .NET.

Since we need to be able to use polymorphic pseudonyms in different languages, different libraries were created for those different languages. A Java library was created using the standard Bouncy Castle crypto library [36]. A .NET library was written in C#, which uses the .NET version of BouncyCastle. This library only implements IdP functionality, since that is all we need for the AD FS plugin. A C library is written based on the standard OpenSSL crypto library [45]. This can be used from C++. Also an extension for PHP was written based on this C library. All source code is available online[1].

9.1 Primitive Choices

For this implementation we used Elliptic Curve ElGamal. The used curve is brainpoolP320r1 [37]. The hash function is SHA-256 [52]. These hashes are embedded in a point on the curve by taking the standard numerical encoding of the hash, and using that as the x-value for the point. An according y is calculated, if it exists. In that case we have a point on the curve. If not, we append a byte to the hash and try again. We keep incrementing this byte until a point on the curve is found. As key derivation function we use the HMAC [35] based on SHA-256.

[1] `https://github.com/polymorphic-pseudonyms`

9.2 SimpleSAMLphp

In SimpleSAMLphp it is possible to create *authentication processing filters* [50]. For the IdP side, the documentation describes this as follows:

> *"Authentication processing filters postprocess authentication information received from authentication sources. It is possible to use this for additional authentication checks, requesting the user's consent before delivering attributes about the user, modifying the user's attributes, and other things which should be performed before returning the user to the service provider he came from." [50]*

These filters can also be used at the SP side. We can use authentication processing filter to perform the necessary pre- and post-processing for polymorphic pseudonyms. Four filters were created:

PseudonymEncrypt To be used in an IdP. This filter creates a polymorphic pseudonym from an input attribute, which is specified in the plugin configuration.

PseudonymSpecializeRemote To be used in an IdP or hub. This filter contacts a PF to perform the specialization. If a *RequesterID* is present (Section 3.1.6), which is the case if the actual SP that should receive the EP is behind a hub, that ID is used to specialize the PP.

PseudonymSpecializeLocal To be used in the hub that also acts as PF. This filter specializes a PP into an EP for the SP SimpleSAMLphp received an authentication request from. It uses the RequesterID if present, just like `PseudonymSpecializeRemote`.

PseudonymDecrypt To be used in an SP. This filter decrypts an EP that is in an attribute specfied by the plugin configuration, and stores the FP in a specified attribute.

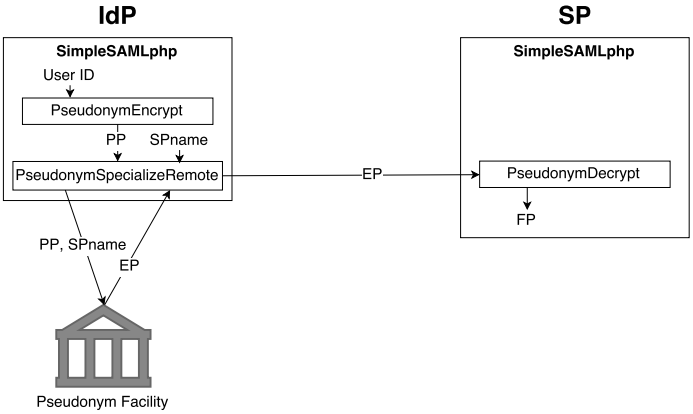
These filters were used in different set-ups. Figure 9.1a shows how the filters can be used in a full mesh federation. Both generation of the PP and requesting an EP from the PF are performed by the IdP. We can also do this when there is a hub, as shown in Figure 9.1b. Now the RequesterID is used, since the IdP sees the hub as being the SP.

If we want the pseudonym specialization to be performed at the hub, we have two options. We can have the PF perform the actual specialization, as seen in Figure 9.1c, or the hub performs the specialization itself, which is depicted in Figure 9.1d.

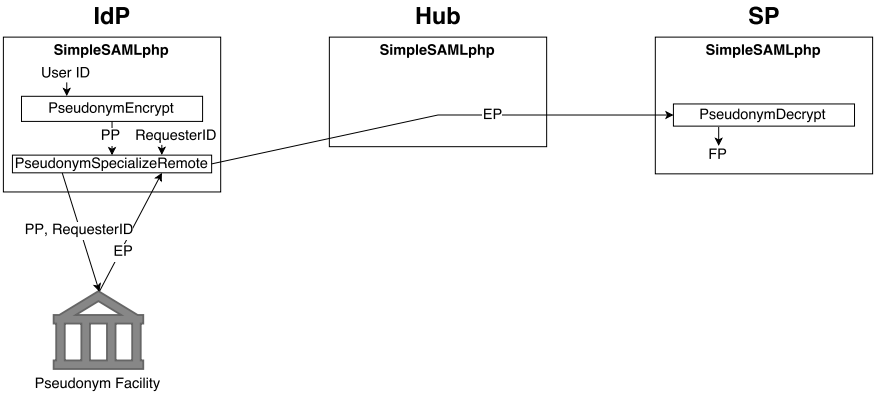
In these described set-ups, the IdP, the hub and the SP were all instances of SimpleSAMLphp. They were also used in combination with the products described below. As long as the same types of pseudonym are used at the same places, this works without problem.

9.3 Shibboleth SP

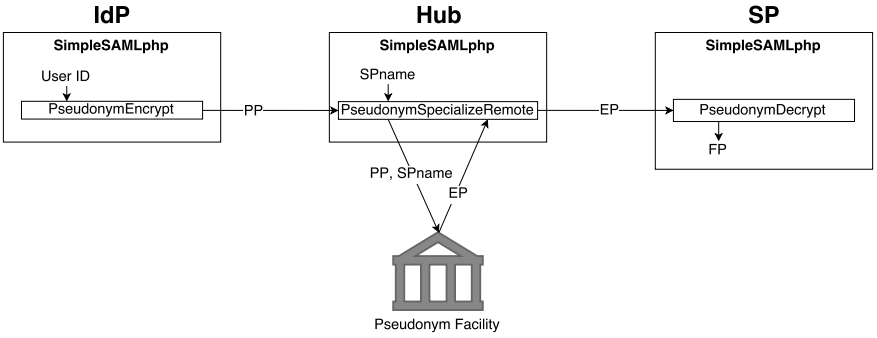
Shibboleth SP provides plugin functionality as is demonstrated in a skeleton plugin project [10]. In this plugin we can extend the AttributeResolver class, to



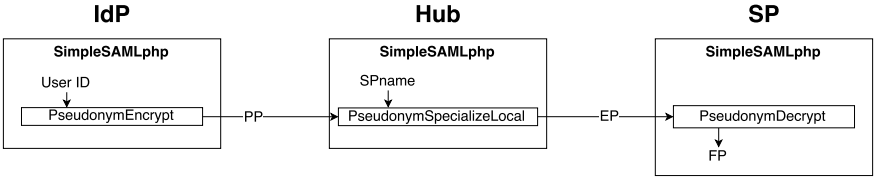
(a) A full mesh federation



(b) The hub passes on the EP it receives



(c) Pseudonym specialization at the hub, performed by the PF



(d) Pseudonym specialization is performed by the hub

Figure 9.1: Set-ups for Authentication Processing Filters in SimpleSAMLphp

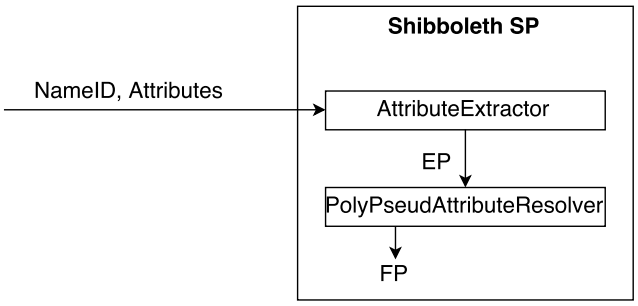


Figure 9.2: Polymorphic pseudonyms in Shibboleth SP

create our own attribute resolver. An attribute resolver can be used to obtain attributes from different locations than what is in the received SAML response, or to perform transformations on received attributes [11]. We want to do the latter: the EP we receive should be transformed into an FP.

A plugin was created that does exactly this. The attribute resolver can be configured with a source attribute containing the EP, a destination attribute in which the FP should be placed, and the private key and closing key of the SP. The C library described above is used to perform the decryption of the EP.

Figure 9.2 shows how this is used. In Shibboleth SP we have attribute extractors to place the NameID or attributes from the SAML response into attributes that can be further used by the SP. These extractors are part of Shibboleth SP, and not of our plugin. An attribute extractor is configured to place the EP, which can be the NameID or the value of a normal attribute, in an internal attribute we also call `EP`. The plugin provides a `PolyPseudAttributeResolver`, which is configured to use the `EP` internal attribute as input, and places the FP in an other internal attribute.

9.4 Shibboleth IdP

Little documentation is available about creating a plugin for Shibboleth IdP. An example is given by [20] of how we can extend certain classes from Shibboleth IdP in a plugin. This way, we can extend the `net.shibboleth.idp.attribute.resolver.AbstractDataConnector` class, to create our own data connector [55]. A data connector can produce `IdPAttribute` objects, which the IdP administrator can configure to be sent to the SP.

Shibboleth IdP has some built in data connectors. One of those is the `RelationalDatabaseConnector`, which can load data from a relational database, using a specified SQL query.

We can make use of this built in data connector, if we use a different approach than for the SimpleSAMLphp plugin. This is depicted in Figure 9.2. We now assume that the IdP knows beforehand which SPs can be used by the users, and a database is prepopulated with PPs and EPs for all users for those SPs. The EP loader fills the database. For each user and for each SP it checks whether there is already a PP in the database. If not, it is generated and stored. Then an EP is requested for the user and SP. This is then stored in the database.

In Shibboleth IdP, the EP for the user, for the SP he wants to use, is loaded

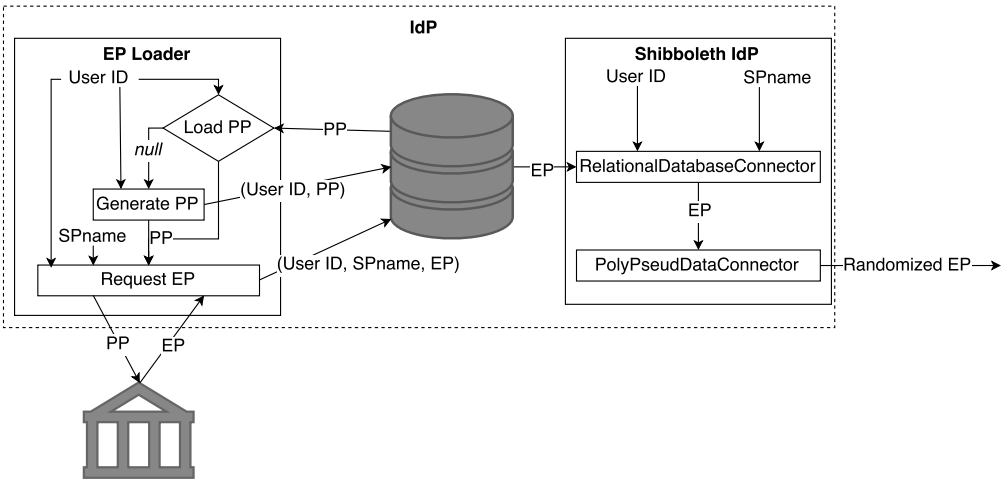


Figure 9.3: Polymorphic pseudonyms in Shibboleth IdP

from the database using a `RelationalDatabaseConnector`. The `PolyPseud-DataConnector`, which is part of our plugin, randomizes the EP.

In the model where PPs are specialized at the hub, it is also possible to only store PPs in the database, and use these as input for our plugin.

It is possible to move more functionality into the plugin, so no prepopulated database is needed.

In Shibboleth IdP it is unfortunately not possible to get access to a RequesterID of an authentication request (Section 3.1.6). This means that if a hub or proxy is present, we cannot send EPs for the SP behind the proxy, since in our software we have no way of knowing which SP that is. This means we must either have a full mesh federation (similar to Figure 9.1a), or specialization should be performed at the hub (similar to Figure 9.1c or 9.1d).

9.5 Microsoft AD FS

An often used system is Microsoft Active Directory Federation Services (AD FS), both by SPs and IdPs. This is a closed source product, so it is not evident that the necessary pre- and post-processing can be added. However, it is very well possible. AD FS passes attributes through a claims engine. This engine uses Attribute Stores. A common attribute store is Active Directory, in which the users and their attributes are stored. It is possible to write a custom attribute store, by implementing the `IAttributeStore` interface in a .NET class [39]. In this custom attribute store we can perform our operations. The system administrator can define claim rules. A claim rule can be used to tell the claims engine to use our custom attribute store. Our set-up can be seen in Figure 9.4.

An attribute store was created which can create a PP for a user, and request an EP for a PP and a specific SP from a PF. PPs and EPs are stored in a database, so they are only generated or requested when no PP can be found in the database for the user, or no EP can be found for the combination of user and SP.

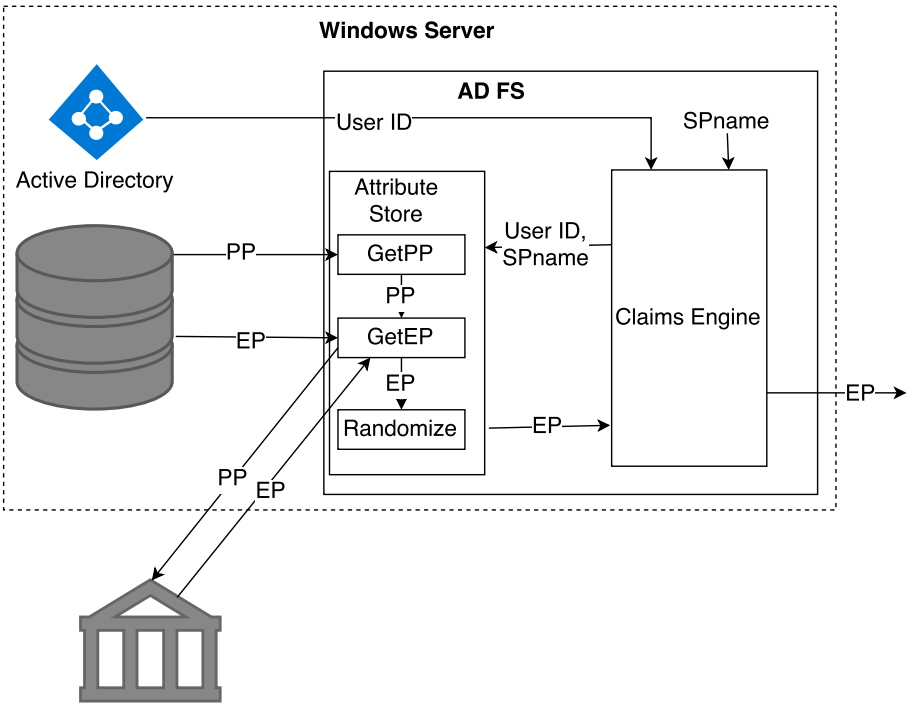


Figure 9.4: Polymorphic pseudonyms in AD FS

The current version of AD FS does not support the scoping element in authentication requests. It responds with an error when the scoping element is present, so when a request is send to an AD FS IdP, the scoping element cannot even be present. The upcoming version has some support for scoping, but it is not yet clear whether it will be possible to get access to the RequesterID inside a custom attribute store. As with Shibboleth IdP, this means that we must have a full mesh federation (Figure 9.1a), or perform the specialization at the hub (Figure 9.1c or 9.1d). The upcoming version may allow specialization at the IdP in a hub-and-spoke federation (Figure 9.1b).

Chapter 10

Polymorphic Attributes

We have been looking at pseudonyms as a way to improve privacy, but all that is negated if there are also attributes being sent that are personally identifiable or contain otherwise personal information. In a similar way to polymorphic pseudonyms, we can also create polymorphic attributes. This way some of the privacy problems of sending attributes can be solved.

The contents of an attribute can be put in a polymorphic attribute. Without a key the original contents cannot be inferred from this polymorphic attribute. The polymorphic attribute can be specialized into an encrypted attribute for a specific SP. Only that SP can decrypt the encrypted attribute with its private key. He will get the original attribute value back. This is different from polymorphic pseudonyms where not the original user ID, but a pseudonym for the user at that specific SP is retreived.

10.1 Opacity at the Hub

We have seen two ways to use polymorphic pseudonyms in a hub-and-spoke federation. In the first variation the hub receives encrypted pseudonyms, which it can just pass on to the SP. In the same way it can also receive encrypted attributes which it just passes on. The hub cannot infer the content of the attributes.

The second way is that the hub receives polymorphic pseudonyms which it specializes for the SP the user wants to access. This can also be done with polymorphic attributes. The hub receives polymorphic attributes. It cannot infer the content of the attributes. It can perform filtering based on attribute names, so it only sends those attributes to the SP that are needed by the SP. It specializes the polymorphic attributes it wants to pass on into encrypted attributes, which can be decrypted by the receiving SP, and only by that SP.

10.2 Attribute Provider

With polymorphic attributes, it is possible to create an attribute provider that can provide attributes without being able to see the content of those attributes. Some party with information about a user can create a polymorphic attribute

containing that information. This polymorphic attribute is stored at the attribute provider. When an SP makes a request for the attribute, the attribute provider must decide whether the SP is allowed to receive the attribute. If that is the case, he specializes the polymorphic attribute into an encrypted attribute for that SP, and sends this encrypted attribute to the SP.

10.3 Cryptography of Polymorphic Attributes

The cryptography of polymorphic attributes is very similar to that of polymorphic pseudonyms, as described in Chapter 7. We need a new primitive: instead of the secure hashing function $\mathcal{I}$, we need a function $\mathcal{E} : \{0,1\}^* \rightarrow G$ that embeds bit arrays into the multiplicative group G which has an inverse $\mathcal{E}^{-1} : G \rightarrow \{0,1\}^*$, so we can get the original value back. Other than that, creating polymorphic attributes is the same as creating polymorphic pseudonyms. A polymorphic attribute PA for value v is an ElGamal encryption with the system public key y_K, using random k:

$$PA = \mathcal{EG}(\mathcal{E}(v), y_K, k) = (g^k, y_K^k \cdot \mathcal{E}(u), y_K) \tag{10.1}$$

We have seen that polymorphic pseudonym specialization consists of three steps:

1. The content of the encrypted message is changed
2. We change the public key of the encryption to that of the party the EP is for
3. The EP is randomized

For attributes, we do not want the final value to be different for each SP. It should be the same value everywhere: the original content of the attribute. Therefore we skip the first step for polymorphic attributes.

Let the polymorphic attribute be: $PA = \mathcal{EG}(\mathcal{I}(u), y_K, k) = (A_1, A_2, A_3)$. We calculate (by equation 7.3):

$$(B_1, B_2, B_3) = (A_1^{\mathcal{M}(D_K,\mathcal{N})}, A_2, A_3^{(\mathcal{M}(D_P,\mathcal{N})^{-1})}) \tag{10.2}$$

We now have, by equation 7.6: $(B_1, B_2, B_3) = \mathcal{EG}(\mathcal{E}(v), y_K^{(\mathcal{M}(D_K,\mathcal{N})^{-1})}, k) = \mathcal{EG}(\mathcal{E}(v), y_\mathcal{N}, k')$, with $k' = k \cdot \mathcal{M}(D_P, \mathcal{N})$

Then we perform randomization, using equation 7.4:

$$(C_1, C_2, C_3) = (B_1 \cdot g^l, B_2 \cdot B_3^l, B_3) \tag{10.3}$$

We now have $(C_1, C_2, C_3) = \mathcal{EG}(\mathcal{E}(v), y_\mathcal{N}, k'+l)$. This is the encrypted attribute.

Decryption is done by performing a standard ElGamal decryption:

$$D = C_2 / C_1^{x_\mathcal{N}} \tag{10.4}$$

We now have $D = \mathcal{E}(v)$.

The final step is to get the original value back from it's embedded version:

$$E = \mathcal{E}^{-1}(D) \tag{10.5}$$

We now have $E = \mathcal{E}^{-1}(\mathcal{E}(v)) = v$

Similar to polymorphic pseudonyms, with polymorphic attributes the randomization step from equation 10.3 can also be performed on polymorphic attributes, and on previously specialized and randomized encrypted attributes.

10.3.1 Multiple Keypairs

The public key pair (x_K, y_K) used for polymorphic attributes should not be the same as the one used for polymorphic pseudonyms. If they are the same, someone may be able to let the PF specialize a PP as if it were a polymorphic attribute. When the recipient decrypts the resulting encrypted attribute, he would get $\mathcal{I}(u)$ instead of a pseudonym for the user, which deanonimizes the user.

It is also possible to not only use different key pairs for pseudonyms and attributes, but also for different attribute groups. Examples of attribute groups are contact information, physical address, and payment details. Another way to group attributes is in levels of privacy sensitivity. By using different key pairs for different attribute groups, we can generate key pairs for SPs only for certain attribute groups, controlling what types of information SPs can get access to.

10.3.2 Hybrid encryption

For longer values it may not be possible to embed them into the multiplicative group that is used in a reversible way. It is common to use hybrid encryption, in which the value itself is encrypted symmetrically, and only the key for this symmetric encryption is encrypted asymetrically [30, Section 10.3]. We would like to be able to do that as well with polymorphic attributes.

The first way to do this, is that we generate a random symmetric key, which we use to encrypt the attribute value. This symmetric key is then encrypted with polymorphic encryption, as we have described above. In this case we can only randomize the encrypted key. The symmetrically encrypted attribute value will stay the same. This is enough to achieve opacity at the hub. If the attributes are encrypted with a new random symmetric key for each session, the sessions can not be linked to each other. We can link what is sent from the IdP to the hub, to what is sent from the hub to the SP for one session. This does however not reveal much information about the users.

The attribute provider we described only stores encrypted values, and can therefore not perform encryption with a different symmetric key each time an attribute is requested. Different requests for attributes of the same user can then be linked to each other. If we want to randomize the encrypted values, we generate a new random symmetric key, and use that to encrypt the already encrypted value again. If we are randomizing an already specialized encrypted attribute, we can ElGamal encrypt the new symmetric key with the public key of the receiving party. If a polymorphic attribute is randomized, we use polymorphic encryption as described above. This encrypted symmetric key will have to be sent to the receiving party, as well as all the previously used keys. So each randomization adds an extra encrypted key that has to be sent.

10.4 Signatures

In order for the SP to trust the attributes it receives, SAML specifies the use of signatures, as we have seen in Section 3.1.7. When we sign polymorphic attributes, or assertions or messages that include polymorphic attributes, signatures will no longer be correct after attributes are specialized or randomized. This

means that one still has to think about how to use signatures when polymorphic attributes are used.

Chapter 11

Security and privacy of polymorphic pseudonyms

11.1 Cryptographic properties

Some cryptographic properties of polymorphic pseudonyms are given by [54] for the education sector use case. When generalized to identity federations, we get the proposition below.

Proposition 11.1. *When the group G has the required cryptographic properties, the KMA, PF and IdPs do not deviate from the described protocols, and parties always use fresh copies of PPs and EPs, then we have the following cryptographic properties:*

CP1 *SPs are not able to link pseudonyms to the user ID that was used to generate the pseudonym.*

CP2 *Cooperating SPs are not able to link their user's pseudonyms.*

CP3 *IdPs are not able to calculate pseudonyms of users at other parties, link those pseudonyms to user ID's, or link pseudonyms at different parties.*

CP4 *The PF does not get information about the activities of users, and is not able to link user pseudonyms with user ID's, or link pseudonyms at different parties.*

CP5 *The KMA does not get information about the activities of users, and is not able to link user pseudonyms with user ID's, or link pseudonyms at different parties.*

CP6 *Eavesdroppers are not able to link user information exchanges based on polymorphic or encrypted pseudonyms.*

11.2 Trust

The use of polymorphic pseudonyms and encryption does not change anything about the issues in Section 3.2.1. The party that performs specialization of PPs,

either by itself or by requesting EPs from the PF should be a trusted party. It needs to be trusted to not specialize them for a different receiver.

11.3 Usability

Polymorphic pseudonyms and attributes work at a technical level which is not visible to the user. The use specifically of polymorphic pseudonyms and attributes to reach certain privacy objectives does therefore not influence usability. The privacy objectives that can be reached, however, make certain usability features harder to implement, especially around consent.

As we have seen in Section 3.2.3, when we encrypt data in such a way that the hub cannot access it, and we want the hub to ask for consent, it is not possible to let the hub show the content of the attributes that are to be released. Using iframes or JavaScript, however, it is possible to let the user agent load attribute values from the IdP. The hub decides which attributes are actually shown on the consent form.

In Section 8.5 we stated that we want to be able to remember consent. We have also seen in Section 3.2.3 that not being able to link sessions makes it impossible to remember consent. The randomization of PPs and EPs makes sure that sessions are unlinkable, and therefore we cannot easily remember consent.

11.3.1 Consent Remembering Service

We also stated that if we do allow linking sessions of the same user and the same SP, that it can be possible to remember consent. It is possible to do this with a consent remembering service, and a special version of the EP. In Section 7.5 we described three steps to specialize a PP:

1. Changing the contents
2. Changing the encryption into one with a different public key
3. Randomizing

For the special EP we change the contents in the first step using the name of the targeted SP. A different pseudonymization master key D_P should be used, to make sure the consent remembering service will receive a different FP from the one received by the SP. In the second step we change the encryption using the name of the consent remembering service, so it will be an encryption with its public key.

Since the special EP is an encryption with the public key of the consent remembering service, it can decrypt it to get a special FP. This FP will be different for each SP the user wants to access, so the consent remembering service cannot link sessions for different SPs. It will be the same each time for the same SP, so the service can link those sessions. Because a different D_P is used, the special FP will be different from the FP at the SP.

The hub can now do the following when deciding whether to ask consent or not:

1. Request a special EP from the pseudonym facility for the PP it received and the SP the user wants to access

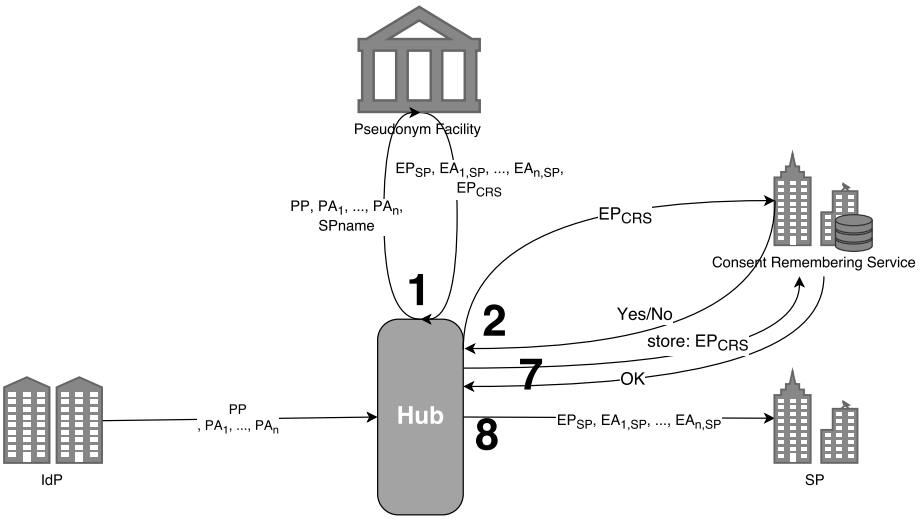


Figure 11.1: Working of the consent remembering service

2. Ask the consent remembering service whether consent was remembered for that special EP
   (a) The consent remembering service decrypts the special EP
   (b) The consent remembering service checks in a database whether consent should be remembered for the special FP
3. If not: continue with the following steps. Otherwise: go to step 8
4. Ask the user for consent
5. If not given: stop. Otherwise continue with these steps
6. Ask if consent should be remembered (this can be combined with step 4)
7. If not: go to step 8. Otherwise: tell the consent remembering service to store consent for the special EP
   (a) The consent remembering service decrypts the special EP
   (b) The consent remembering service stores in a database that consent should be remembered for the special FP
8. release attributes

This is displayed in Figure 11.1.

11.4 Privacy

In Section 3.2.2 we introduced architectural requirements from [23] for privacy by design. In Chapter 4 we saw that not all of these requirements were met in the SURFconext and Kennisnet use cases. We will now see how polymorphic

pseudonyms and polymorphic attributes can help meeting the architectural requirements. ✓ denotes an architectual requirement that can be met using polymorphic pseudonyms, ✗ denotes a requirement that is not met.

AR1 Limited observability. We have seen that this requirement is at odds with prevention of man-in-the-browser attacks (Section 3.2.1). If desired however, it can be accomplished by using polymorphic pseudonyms with a hub. The hub must not pass a RequesterID or other information about which SP is used to the IdP, otherwise, if the IdP knows who the user is, he can learn which services are used by a user. The IdP sends the PP to the hub, without knowing for which SP it is going to be used. The hub cannot know from this PP which user is accessing a certain service, but he can specialize it and pass the EP to the SP.

With polymorphic pseudonyms it is also possible to have a solution in which the IdP does not know who the user is, which we will see in Section 11.4.1 ✓

AR2 Limited linkability. FPs are different and unlinkable for different SPs. No party except the SP knows the FP for a user. So the IdP, hub or SPs with which constrained linking is allowed all do not know this FP. Limited linkability can therefore be implemented with polymorphic pseudonyms ✓

AR3 Prevent the unauthorized aggregation of attributes by central intermediaries such as gateways, brokers, etc. With end-to-end encryption, attribute contents can be shielded from intermediaries. We can use polymorphic attributes to do this, which will have to be specialized by a hub. That way we can use end-to-end encryption while retaining **AR1** ✓

AR4 Constrained linking. By passing the EP of a user at one SP to another SP, the second SP can communicate to the first one about the user. This way constrained linking can be achieved ✓

AR5 Consent handling. If user consent is handled at the IdP nothing changes. When consent is handled at the hub, and we use polymorphic encryption, the hub cannot show the contents of the attributes that will be shared, only their names. With a consent remembering service, we can still remember consent ✓

AR6 No supreme instance. The KMA must be a separate entity, without access to attributes and transaction data ✓

AR7 Minimize the release of attributes. If polymorphic attributes are used, the IdP can send all attributes it wants to a hub, without knowing or caring about the SP they will be sent to, for example to accomplish **AR1**. The hub can filter attributes based on their name, so the release of attributes to the SP is still minimized. At the same time, no attributes are released to the hub, since they are encrypted ✓

AR8 Uniqueness of identification. If the user ID that is used as input for the generation of the PP is the same each time, the FP will also be the same for an SP. Furthermore, if the group G that is used for the cryptographic

operations is large enough, the FP can be assumed to be unique for a user ✓

11.4.1 Polymorphic Pseudonym Card Application

We have seen that there are security problems when we want the IdP to comply with **AR1**. It cannot ask for consent, and use this to prevent man-in-the-browser attacks. We do not have those problems if the IdP does not know who the user is. Then the IdP can do all those things without violating **AR1**. A polymorphic pseudonym card application (PPCA), as described by [46], allows an IdP to store the user's PP on a smartcard. The card can randomize the PP. The IdP can first issue a PP to the card. The user can at a later time use this with the same IdP, but since the PP is randomized, the IdP cannot link this usage to earlier uses or to the issuance of the PP. This allows the user to authenticate while the IdP does not know who the user is. Using techniques similar to those used by the German eID card [8], the IdP can check whether the card is valid, e.g. it is not revoked.

Chapter 12

Using conventional cryptography

Some of the objectives of polymorphic pseudonyms can also be reached with conventional cryptographic techniques. We will take a look at these objectives and how they can be solved with conventional methods. We will also see where they fall short.

12.1 Prevent the unauthorized aggregation of attributes by central intermediaries such as gateways, brokers, etc. (AR3)

End-to-end encryption can make sure intermediaries cannot access attribute values. The IdP encrypts the user data with the public key of the SP. A hub or other intermediary can only pass on the data it receives.

12.2 Limited linkability (AR2)

Targeted identifiers are a well known technique to limit linkability. They can be randomly generated and stored in a database, or we can use a HMAC of the user ID and the unique name of the service provider, with a key only known to the party creating the pseudonym. This can be the IdP or a seperate PF. We need a key instead of just a hash, otherwise any party with access to pseudonyms would be able to generate pseudonyms for user IDs, and see whether they are the same as the pseudonyms it knows. With this approach, the party that creates the pseudonyms knows the pseudonym of a user at an SP.

We also may not want any intermediary to be able to link different sessions of the same user, even if he cannot see the user or the contents of attributes. If encryption is used, we can include a random nonce or random padding like RSA-OAEP[29]. This makes sure that the encryptions, and therefore the sessions, cannot be linked. It is not possible to randomize already encrypted data this way.

12.3 Uniqueness of identification (AR8)

The HMAC solution already ensures non-reassignability of identifiers, since if a secure HMAC is used, it will be different for different input, and therefore also for different users. We do not have immutability, since we will have different pseudonyms when one user uses different IdPs. This is caused by the different keys the IdPs use for the HMAC. We can set up a central facility that generates all pseudonyms. That pseudonym facility will know for which user-SP combinations pseudonyms are requested, which means it can see which services a user uses. Furthermore, the PF will have access to all pseudonyms for all users, allowing it to deanonimize all users. It does not have access to the sessions between users and SPs, so it still does not know what a user does at the SPs it visits.

12.4 Constrained linking (AR4)

If party A wants to communicate something about a user to party B, with polymorphic pseudonyms we can send the encrypted pseudonym of the user for B to A, so A can include this encrypted pseudonym in his communication with B. We can do something similar with existing techniques. The IdP sends the pseudonym of the user at B with the public key of B, and sends that to A. A can then again include this encrypted pseudonym in his communication with B, so B knows about which user they are communicating.

12.5 Putting it together

We will see how this works together in an example. We have a hub-and-spoke federation. In this federation we have SpaceDrive, a cloud storage service. Additional storage space at SpaceDrive can be purchased via Unishop.com. Bobby Tables, student at the University of Harderwijk, authenticates with the University of Harderwijk as his IdP to Unishop.com, and purchases extra storage space for SpaceDrive. Unishop.com now has to communicate to SpaceDrive that Bobby has purchased additional storage space. There is, however, no need for Unishop.com or SpaceDrive to know that the user is Bobby, so pseudonyms are used. There is also no need for Unishop.com to know Bobby's pseudonym at SpaceDrive, or vice versa. Combining the methods described above, this works as follows:

1. The University of Harderwijk sends a request to the PF for
   `(b.tables@universityofharderwijk.com, Unidrive.com)`
   and one for
   `(b.tables@universityofharderwijk.com, SpaceDrive.com)`.

2. The PF calculates

$$P_{unishop.com} = HMAC(D_P, \texttt{b.tables@universityofharderwijk.com} \mid \texttt{Unishop.com})$$

and

$$P_{spacedrive} = HMAC(D_P, \texttt{b.tables@universityofharderwijk.com} \\ |\texttt{SpaceDrive.com})$$

and sends these back to the University of Harderwijk.

3. The university stores the pseudonyms in a database.
4. Bobby Tables wants to log in to Unishop.com to purchase additional storage. Unishop.com sends an authentication request to the hub, which in turn sends an authentication request to the University of Harderwijk.
5. Bobby Tables authenticates to his university.
6. The University of Harderwijk looks up the attributes of Bobby Tables that are needed for Unishop.com, and his pseudonyms $P_{unishop.com}$ and $P_{spacedrive}$.
7. The attributes are encrypted with the public key of Unishop.com: $\{attr_1\}_{K_{unishop.com}}, \ldots, \{attr_n\}_{K_{unishop.com}}$
   $P_{unishop.com}$ is also encrypted with the public key of Unishop.com, and $P_{spacedrive}$ is encrypted with the public key of SpaceDrive: $\{P_{unishop.com}\}_{K_{unishop.com}}, \{P_{spacedrive}\}_{K_{spacedrive}}$
8. These encrypted attributes and both encrypted pseudonyms are sent in the response to the hub, which passes them on to Unishop.com.
9. Bobby makes the purchase at Unishop.com. Unishop.com sends a message to SpaceDrive that additional storage was purchased, and includes $\{P_{spacedrive}\}_{K_{spacedrive}}$ in this message.
10. SpaceDrive decrypts $\{P_{spacedrive}\}_{K_{spacedrive}}$ and knows that the user with pseudonym $P_{spacedrive}$ has purchased additional storage.

The general set-up for this process is depicted in Figure 12.1. Unishop.com is SP_1, and SpaceDrive is SP_2. We use $\{m\}_{K_A}$ as the notation for an encryption of message m, with the public key of A, concatenation of x and y is denoted as $x|y$, and $HMAC(K, m)$ denotes the HMAC of message m, with key K. We assume encryptions are using random nonces or random padding. We only show the system for one user, and one IdP. Each IdP can do the same for all its users.

- The hub does not see the contents of attributes, nor pseudonyms of users, since end-to-end encryption is used.
- If a user uses a different IdP, the PF will still generate the same pseudonym for the same user, as long as the same user ID is used. Bobby Tables can keep using his additional storage if he logs in via a different IdP.
- Using a cryptographically secure HMAC function ensures unlinkability of pseudonyms to different pseudonyms for the same user, or to the user ID. The key D_P makes sure that only the PF can generate pseudonyms, and no other party can compare pseudonyms to self-generated pseudonyms for user ID's he knows.

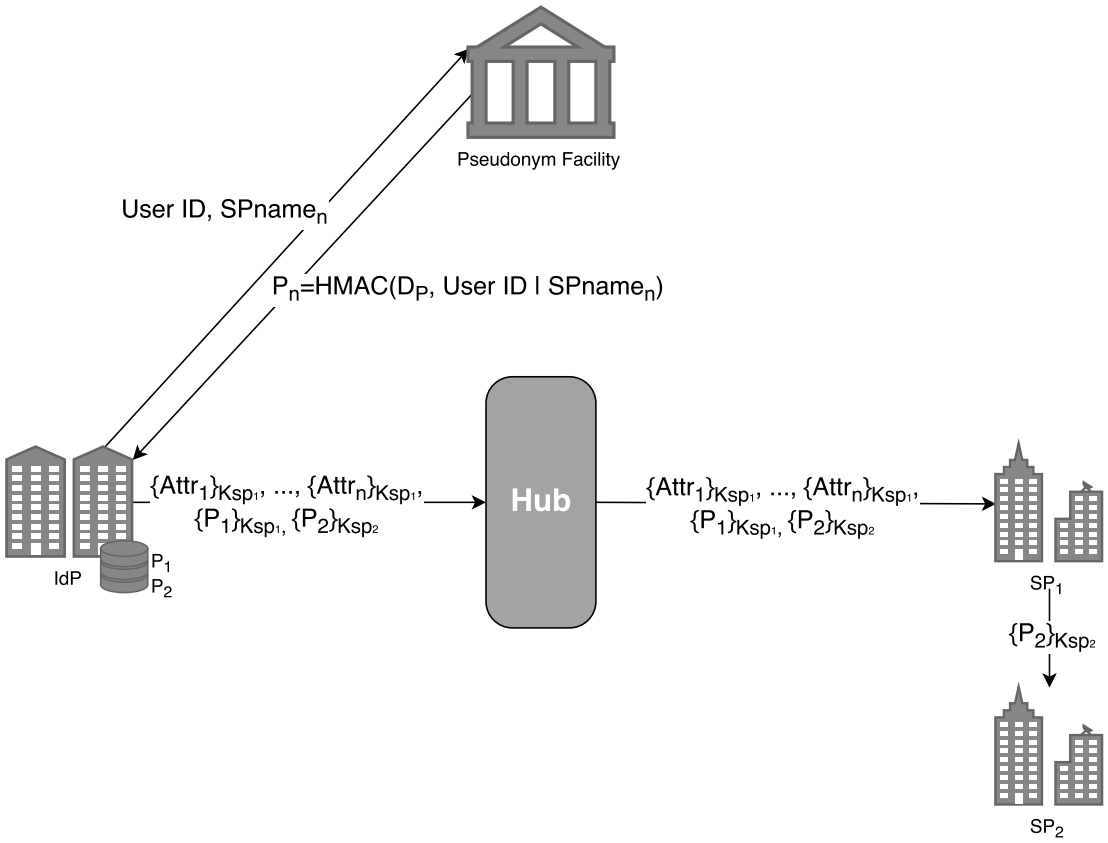


Figure 12.1: A set-up with conventional cryptographic techniques

- The IdP can allow linkability of a user between SPs by including an encrypted pseudonym, in the way we described above.
- Encryption of the pseudonym that is used for linkability ensures that cooperating SPs do not learn the pseudonym of a user at other SPs.

12.6 Comparison with polymorphic pseudonyms

The advantage of using the solutions described in this chapter is that it can be used with standard SAML without much trouble. SAML already supports encryption, as described in Section 3.1.7, and by using conventional encryption we can use this existing part of the specification. Polymorphic pseudonyms and attributes require a more specific use of encryption which is not part of the SAML standard. SAML does give us enough freedom, however, to send our encrypted pseudonyms and attributes. This will just not be in a standardized way.

There are some limitations to the described solutions First of all **AR1** is still violated, although this can be solved with a more complex solution such as [24], which we described in Section 5.1. Not only for privacy reasons, but also because of limitations in the software and to reduce complexity for IdP administrators it can be valuable for the IdP to not have to worry about the SP that is used. Even in the solution from [24] the IdP has to know the one-time public key of the SP, so the IdP does need to know something about the SP. We have seen in Chapter 9 that not all software gives plugins access to this information in a hub-and-spoke scenario. With polymorphic pseudonyms the IdP does not have to know anything about the SP that is used. This makes it possible to create a plugin for existing software, and it makes it easy to configure the connection with the federation.

ElGamal encrypted data, and therefore polymorphic pseudonyms and attributes, can be randomized. With other solutions, randomization can only be done before encryption.

The PF knows which SPs are used by a user, and it knows the pseudonym the user gets. With polymorphic pseudonyms the PF does not know for which user it is specializing a PP, and it does not know the FP of the user at the SP.

Another property of polymorphic pseudonyms is that the FP for a user at an SP is only known to that SP. FP, IdP, hub and collaborating SPs all do not know this FP. In the solution described here, if no restrictions are put in place on the users for which an IdP can request pseudonyms, any IdP can learn the pseudonyms for any user. So linkability is more limited when polymorphic pseudonyms are used, than when conventional cryptography is used.

Chapter 13

Conclusion

13.1 Future Work

We have analysed the security of polymorphic pseudonyms in Chapter 11. However, since cryptography is very hard to get right, further review of the system is necessary to make sure no mistakes were made. This is also the case for the implementations.

An analysis was made on the use of polymorphic pseudonyms and attributes in SURFconext and Kennisnet. Both SURFnet and Kennisnet must do further evaluation about what the consequences are of introduction of these techniques, and what needs to be changed to the current setup. Based on that, and on the findings in this thesis, a decision has to be made whether and how to use these techniques in these use cases.

13.2 Conclusion

What pros and cons are there when using polymorphic pseudonyms in a hub-and-spoke federation? The use of polymorphic pseudonyms can greatly improve the privacy of the users, especially when combined with polymorphic attributes. We can use it to ensure that the hub is not able to gather information about the users and their usage of services. The hub can no longer see the contents of attributes, and also cannot see which user a session belongs to. The hub can therefore not link usage of specific services to users.

A problem in hub-and-spoke federations is that in multiple software products for IdPs it is not possible to know the actual SP when there is a hub between the IdP and the SP. This is a problem in general, since the IdP cannot show to the user which SP he is authenticating to, but also limits us in the possibilities to use polymorphic pseudonyms. We cannot let the IdP request EPs, since it needs to know the SP for which it needs an EP. However, the hub can request EPs.

Polymorphic pseudonyms, in combination with polymorphic attributes, however, also give new solutions to problems that are caused by the fact that the IdP does not know the actual SP. If we want end-to-end encryption, with conventional encryption the IdP needs to know the SP for which it should encrypt attributes and pseudonyms. The IdP can however send polymorphic pseud-

onyms and attributes to the hub without knowing the SP. The hub can then specialize these into encrypted pseudonyms and attributes for the SP.

There are variations available of the polymorphic pseudonym system. What are pros and cons of different variations? The location of pseudonym specialization can vary. It can be performed by a PF, upon request of the IdP, or at the hub. In this last variation we can still have a separate PF from which the hub requests EPs, or we can make the PF part of the hub and let it specialize PPs by itself. If we want to allow multiple hubs, it is advantageous to have the PF as a seperate entity, such that not all hubs need to have the keys that are necessary for the specialization. Furthermore, for organizational and network security reasons it can be desirable to have the PF as a seperate entity. For the further discussion there are no differences between these possibilities, and we will therefore not treat them as separate variations.

A problem with specialization at the hub is that if the hub operator also operates any SPs, it can specialize the PPs and polymorphic attributes it receives not only for the intended recipient, but also for an SP operated by itself. This means he can see the contents of attributes, and can link sessions of a user to the FP of the user at its own SP. When EPs are requested by the IdP we do not have this problem.

A problem with requesting EPs by the IdP is that the IdP must know which SP is used. This is desirable anyway, in order to let the IdP ask for consent to release attributes to that specific SP, and prevent damage from man-in-the-browser attacks, but for privacy reasons one may want the IdP to remain unknowing of this information. There is also the problem that most IdP software does not properly handle this information, and has no access to the name of the actual SP, and only sees the hub as being the SP. When PPs and polymorphic attributes are specialized at the hub, the IdP does not have to know the actual SP, and we do not have these problems.

When PPs are specialized at the hub, it is still possible to give the IdP the name of the SP. If the software supports it, it is therefore still possible to let the IdP use this information, for example to ask consent.

When EPs are requested by the IdP, we do not need a hub, so this can also work in a full mesh federation. In this case the IdP does know which SP is used, and we do not have the problems related to the IdP not knowing this.

Using PPCA we can have properties of both set-ups. The IdP is then allowed to know the SP that is used, and is therefore able to ask the user for consent to authenticate to that specific SP. This also means no hub is necessary. However, the IdP does not know which user is authenticating, and therefore cannot link users to the usage of SPs.

Does the use of polymorphic pseudonyms fit in a standard SAML environment? How much extra processing is needed? SAML allows us to use the 'unspecified' format for the NameID. This does mean that no information is given on how to interpret it. We can also define a custom format. Existing software, however, will still not know what this means and does not know how to interpret the NameID. Another option is to use the 'transient' NameID format, which means the NameID will be different and unlinkable every time, and include the PP or EP in an attribute.

The IdP has to generate PPs for its users. They only have to be generated once and can be stored and reused. Depending on the used variation, the IdP also needs to request EPs for each user, for each SP he is using. These can also be stored and reused. When stored PPs or EPs are reused, they need to be randomized. SPs have to decrypt EPs they receive.

For standard SAML software, plugins were created that perform the required processing. In most IdP software, however, it is not possible to let an IdP in a hub-and-spoke federation request EPs. For SPs there were no such problems.

What is the impact on existing identity federations, such as Kennisnet and SURFconext, for IdPs, SPs as well as for the hub? On IdPs and SPs a plugin must be installed and configured. Once a plugin is finished for the software used by the IdPs or SPs, no extra implementation work has to be done by the IdPs and SPs. The plugin will have to be maintained, but this can be the responsibility of one party, and not of all connected parties. Some extra computation power and storage is necessary to generate, randomize, decrypt and store pseudonyms.

If EPs are requested by the IdP, the hub only has to pass on the EPs it receives, and no extra processing is necessary. In the other case, the hub has to perform the specialization. In both cases, if polymorphic attributes are also used, the hub and its operators are not able to see the contents of attributes. This will make it more difficult to solve problems.

Which variation is the best fit for the SURFconext and Kennisnet use cases? The model where PPs and polymorphic attributes are sent to the hub, where they are specialized before sending them to SPs seems to be the best fit for SURFconext. Currently, SURFconext does not require the IdPs to worry about which SP is used. The software that is used by many IdPs does not support the IdP to know which SP is used in a hub-and-spoke federation. This means requesting EPs by the IdP is not possible. IdPs that do support using the name of the actual SP can still be given this information.

In the Kennisnet use case we do not have a single hub-and-spoke federation, but communication about pupils can go through different channels. Therefore, the model where specialization is done at the hub is not feasible. EPs and encrypted attributes should be requested by the IdP. This does pose some problems with used software, since the software needs to know the targeted SP, and not all software supports this for hub-and-spoke federation. A hybrid solution is possible, where the IdP requests EPs for those SPs with which it communicates directly, and PPs are sent to the hubs from the different federations. The hubs can then request EPs from the PF.

Acronyms

ABC attribute based credential.

AD authenticatiedienst.

AD FS Microsoft Active Directory Federation Services.

BIN bank identification number.

CA certificate authority.

CDH Computational Diffie-Hellman.

DDH Decision Diffie-Hellman.

DISP digital identity service provider.

DL Discrete Logarithm.

DPD Data Protection Directive.

DV dienstverlener.

EP encrypted pseudonym.

FIM federated identity management.

FP final pseudonym.

GDPR General Data Protection Regulation.

HM herkenningsmakelaar.

IdP identity provider.

IRMA I Reveal My Attributes.

KDF key derivation function.

KMA key management authority.

KR koppelregister.

MR machtigingenregister.

NRN national registration number.

OD ondertekendienst.

PE-FIM privacy enhanced federated identity management.

PF pseudonym facility.

PGN persoonsgebonden nummer.

PIA privacy impact assessment.

PP polymorphic pseudonym.

PPCA polymorphic pseudonym card application.

RP relying party.

SAML Security Assertion Markup Language.

SP service provider.

SSO single sign on.

TTP trusted third party.

VPN Virtual Private Network.

References

[1] Anne Adams and Martina Angela Sasse. 'Users Are Not the Enemy'. In: *Commun. ACM* 42.12 (Dec. 1999), pp. 40–46. ISSN: 0001-0782. DOI: 10.1145/322796.322806. URL: http://doi.acm.org.ru.idm.oclc.org/10.1145/322796.322806.

[2] Cloud Security Alliance. *Security guidance for critical areas of focus in cloud computing v3.0*. 2011. URL: https://cloudsecurityalliance.org/download/security-guidance-for-critical-areas-of-focus-in-cloud-computing-v3/.

[3] Gergely Alpár and Jaap-Henk Hoepman. 'A Secure Channel for Attribute-based Credentials: [Short Paper]'. In: *Proceedings of the 2013 ACM Workshop on Digital Identity Management*. DIM '13. Berlin, Germany: ACM, 2013, pp. 13–18. ISBN: 978-1-4503-2493-9. DOI: 10.1145/2517881.2517884. URL: http://doi.acm.org/10.1145/2517881.2517884.

[4] Gergely Alpár, Jaap-Henk Hoepman and Johanneke Siljee. 'The Identity Crisis. Security, Privacy and Usability Issues in Identity Management'. In: *CoRR* abs/1101.0427 (2011). URL: http://arxiv.org/abs/1101.0427.

[5] Gergely Alpár and Bart Jacobs. 'Credential design in attribute-based identity management'. In: *Bridging distances in technology and regulation, 3rd TILTing Perspectives Conference*. 2013, pp. 189–204.

[6] Mark Bartel et al. *XML Signature Syntax and Processing (Second Edition)*. W3C Recommendation. June 2008. URL: http://www.w3.org/TR/2008/REC-xmldsig-core-20080610/.

[7] Dan Boneh. 'Algorithmic Number Theory: Third International Symposiun, ANTS-III Portland, Oregon, USA, June 21–25, 1998 Proceedings'. In: ed. by Joe P. Buhler. Berlin, Heidelberg: Springer Berlin Heidelberg, 1998. Chap. The Decision Diffie-Hellman problem, pp. 48–63. ISBN: 978-3-540-69113-6. DOI: 10.1007/BFb0054851. URL: http://dx.doi.org/10.1007/BFb0054851.

[8] Bundesamt für Sicherheit in der Informationstechnik. *Advanced Security Mechanisms for Machine Readable Travel Documents and eIDAS Token – Part 3 – Common Specifications Version 2.20*. Technical Guideline TR-03110-3. Feb. 2015. URL: https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/Publications/TechGuidelines/TR03110/BSI_TR-03110_Part-3-V2_2.pdf.

[9] Jan Camenisch and Els Van Herreweghen. 'Design and Implementation of the Idemix Anonymous Credential System'. In: *Proceedings of the 9th ACM Conference on Computer and Communications Security*. CCS '02. Washington, DC, USA: ACM, 2002, pp. 21–30. ISBN: 1-58113-612-9. DOI: 10.1145/586110.586114. URL: http://doi.acm.org.ru.idm.oclc.org/10.1145/586110.586114.

[10] Scott Cantor. *cpp-sp-ext*. Git repository, revision 48a907140e04788a3e68f4efcdff34ce6ae0b334. Nov. 2011. URL: https://git.shibboleth.net/view/?p=cpp-sp-ext.git.

[11] Scott Cantor et al. *NativeSPAttributeResolver*. Version 24. Apr. 2014. URL: https://wiki.shibboleth.net/confluence/display/SHIB2/NativeSPAttributeResolver.

[12] *COMMISSION IMPLEMENTING REGULATION (EU) 2015/1502. on setting out minimum technical specifications and procedures for assurance levels for electronic identification means pursuant to Article 8(3) of Regulation (EU) No 910/2014 of the European Parliament and of the Council on electronic identification and trust services for electronic transactions in the internal market*. OJ L 235, 9.9.2015, p. 7—20. Sept. 2015. URL: http://data.europa.eu/eli/reg_impl/2015/1502/oj.

[13] Currence. *iDIN Merchant Implementation Guidelines*. Version 1.041. Jan. 2016. URL: http://www.betaalvereniging.nl/giraal-en-online-betalen/idin/documentatie-aanvragen/.

[14] Currence. *Welkom bij iDEAL*. Retreived 26th April 2016. URL: https://www.ideal.nl/.

[15] P. Deutsch. *DEFLATE Compressed Data Format Specification version 1.3*. RFC 2104. May 1996. URL: http://www.rfc-editor.org/info/rfc2104.

[16] R. Dhamija and L. Dusseault. 'The Seven Flaws of Identity Management: Usability and Security Challenges'. In: *IEEE Security Privacy* 6.2 (Mar. 2008), pp. 24–29. ISSN: 1540-7993. DOI: 10.1109/MSP.2008.49.

[17] *DIRECTIVE 95/46/EC OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL. on the protection of individuals with regard to the processing of personal data and on the free movement of such data*. OJ L 281, 23.11.1995, p. 31 – 50. Oct. 1995. URL: http://data.europa.eu/eli/dir/1995/46/oj.

[18] Edustandaard. *ECK Distributie en toegang*. Version 2.0. Feb. 2016. URL: https://www.edustandaard.nl/standaarden/afspraken/afspraak/eck-dt/2.0/.

[19] Taher ElGamal. 'A public key cryptosystem and a signature scheme based on discrete logarithms'. In: *Advances in cryptology*. Springer. 1985, pp. 10–18.

[20] Jim Fox. *webservice dataconnector for shib idp 3.2.1*. GitHub repository, revision ea4a75b662ee5b853eea1a405241ff0dc34ba722. Feb. 2016. URL: https://github.com/UWIT-IAM/uw-idp-rws-connector.

[21] Bits of Freedom. *De winnaars: Opstelten en de Nederlandse Scholen*. Dec. 2014. URL: https://bba2014.bof.nl/de-winnaars-opstelten-en-de-nederlandse-scholen/.

[22] PBLEQ HEC. *Privacy Impact Assessment Nummervoorziening in de Leermiddelenketen, version 1.0*. May 2015. URL: https://www.rijksoverheid.nl/documenten/rapporten/2015/05/27/privacy-impact-assessment-nummervoorziening-in-de-leermiddelenketen.

[23] R. Hörbe and W. Hötzendorfer. 'Privacy by Design in Federated Identity Management'. In: *Security and Privacy Workshops (SPW), 2015 IEEE*. May 2015, pp. 167–174. DOI: 10.1109/SPW.2015.24.

[24] Rainer Hörbe. 'A Model for Privacy-enhanced Federated Identity Management'. In: *CoRR* abs/1401.4726 (2014). URL: http://arxiv.org/abs/1401.4726.

[25] Idensys. *Interface specifications*. Mar. 2016. URL: https://afsprakenstelsel.etoegang.nl/display/as/Interface+specifications.

[26] Takeshi Imamura, Blair Dillaway and Ed Simon. *XML Encryption Syntax and Processing*. W3C Recommendation. Dec. 2002. URL: http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/.

[27] *IRMA app*. URL: https://www.irmacard.org/irmaphone/.

[28] *IRMA – I Reveal My Attributes*. URL: http://irmacard.org/.

[29] J. Jonsson and B. Kaliski. *Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1*. RFC 3447. Feb. 2003. URL: http://www.rfc-editor.org/info/rfc3447.

[30] Jonathan Katz and Yehuda Lindell. *Introduction to modern cryptography*. CRC Press, 2007.

[31] Kennisnet. *Entree Federatie*. URL: https://www.kennisnet.nl/entree-federatie/.

[32] Kennisnet. *Wie wij zijn*. URL: https://www.kennisnet.nl/wie-wij-zijn/.

[33] Charles Knouse. *SAML Implementation Guidelines*. Working Draft 01. Aug. 2004. URL: https://www.oasis-open.org/committees/download.php/8958/sstc-saml-implementation-guidelines-draft-01.pdf.

[34] Carlo Koch and Gerrit Jan van 't Eind, eds. *Masterplan eID*. Sept. 2014. URL: https://www.idensys.nl/fileadmin/bestanden/idensys/documenten/basisdocumentatie/Masterplan_eID_vs1_00_def.pdf.

[35] H. Krawczyk, M. Bellare and R. Canetti. *HMAC: Keyed-Hashing for Message Authentication*. RFC 2104. Feb. 1997. URL: http://www.rfc-editor.org/info/rfc2104.

[36] Legion of the Bouncy Castle Inc. *Bouncy Castle Crypto APIs*. 2015. URL: http://www.bouncycastle.org/.

[37] Manfred Lochter and Johannes Merkle. *Elliptic curve cryptography (ECC) brainpool standard curves and curve generation*. RFC 1654. Mar. 2010. URL: http://www.rfc-editor.org/info/rfc5639.

[38] Salah Machani et al., eds. *FIDO UAF Architectural Overview*. Dec. 2014. URL: https://fidoalliance.org/specs/fido-uaf-v1.0-ps-20141208/fido-uaf-overview-v1.0-ps-20141208.html.

[39] Microsoft. *IAttributeStore Interface*. URL: https://msdn.microsoft.com/en-us/library/microsoft.identityserver.claimspolicy.engine.attributestore.iattributestore.aspx.

[40] L. I. Millett and S. H. Holden. 'Authentication and its privacy effects'. In: *IEEE Internet Computing* 7.6 (Nov. 2003), pp. 54–58. ISSN: 1089-7801. DOI: 10.1109/MIC.2003.1250584.

[41] National Institute of Standards and Technology. *Digital Signature Standard (DSS)*. FIPS PUB 186–4. July 2013. DOI: 10.6028/NIST.FIPS.186-4. URL: http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-4.pdf.

[42] OASIS. *Assertions and protocols for the OASIS security assertion markup language (SAML) v2.0*. Mar. 2005. URL: http://docs.oasis-open.org/security/saml/v2.0/saml-core-2.0-os.pdf.

[43] OASIS. *Profiles for the OASIS security assertion markup language (SAML) v2.0*. Mar. 2005. URL: http://docs.oasis-open.org/security/saml/v2.0/saml-profiles-2.0-os.pdf.

[44] OASIS. *Security and Privacy Considerations for the OASIS Security Assertion Markup Language (SAML) V2.0*. Mar. 2005. URL: http://docs.oasis-open.org/security/saml/v2.0/saml-profiles-2.0-os.pdf.

[45] *OpenSSL Cryptography and SSL/TLS Toolkit*. URL: https://openssl.org/.

[46] programma eID. *Polymorphic Pseudonymization*. July 2014. URL: https://www.idensys.nl/fileadmin/bestanden/idensys/documenten/basisdocumentatie/documentatieset/PP_Scheme_091.pdf.

[47] Programmabureau eID. *Jaarplan 2015 Programma eID*. Mar. 2015. URL: https://www.idensys.nl/fileadmin/bestanden/idensys/documenten/basisdocumentatie/Jaarplan_eID-programma__2015.pdf.

[48] *REGULATION (EU) 2016/... OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL. on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation)*. ST 5419 2016 REV 1 - 2012/011. Apr. 2016. URL: http://eur-lex.europa.eu/legal-content/EN/NOT/?uri=consil:ST_5419_2016_REV_1.

[49] *REGULATION (EU) No 910/2014 OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL. on electronic identification and trust services for electronic transactions in the internal market and repealing Directive 1999/93/EC*. OJ L 257, 28.8.2014, p. 73–114. July 2014. URL: http://eur-lex.europa.eu/eli/reg/2014/910/oj.

[50] SimpleSAMLphp. *Authentication Processing Filters in SimpleSAMLphp*. URL: https://simplesamlphp.org/docs/stable/simplesamlphp-authproc.

[51] Sampath Srinivas et al., eds. *Universal 2nd Factor (U2F) Overview*. May 2014. URL: https://fidoalliance.org/specs/fido-undefined-undefined-ps-20150514/fido-u2f-overview-v1.0-undefined-ps-20150514.html.

[52] National Institute of Standards and Technology. *Secure Hash Standard (SHS)*. FIPS PUB 180-4. Aug. 2015. URL: http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf.

[53] Pieter Verhaeghe et al. 'Emerging Challenges for Security, Privacy and Trust: 24th IFIP TC 11 International Information Security Conference, SEC 2009, Pafos, Cyprus, May 18–20, 2009. Proceedings'. In: ed. by Dimitris Gritzalis and Javier Lopez. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009. Chap. Security and Privacy Improvements for the Belgian eID Technology, pp. 237–247. ISBN: 978-3-642-01244-0. DOI: 10.1007/978-3-642-01244-0_21. URL: http://dx.doi.org/10.1007/978-3-642-01244-0_21.

[54] Eric R. Verheul. *Privacy protection in electronic education based on polymorphic pseudonymization*. Cryptology ePrint Archive, Report 2015/1228. http://eprint.iacr.org/2015/1228. 2015.

[55] Rod Widdowson, Scott Cantor and Alex Stuart. *DataConnectorConfiguration*. Version 25. Jan. 2016. URL: https://wiki.shibboleth.net/confluence/display/IDP30/DataConnectorConfiguration.