MASTER

Extending DigiD to the private sector (DigiD-2)

Moniava, G.

Award date:
2008

TECHNISCHE UNIVERSITEIT EINDHOVEN
Department of Mathematics and Computer Science

MASTER'S THESIS

Extending DigiD to the Private Sector (DigiD-2)

By
Giorgi Moniava

Supervisors:
Eric Verheul (RU, PwC)
L.A.M. Schoenmakers (TU/e)

Eindhoven, August 2008

(This page is intentionally left blank.)

Abstract

DigiD is a single sign on system used by Dutch governmental organizations for authentication of Dutch citizens online. Currently, the scope of organizations that can use DigiD is only limited to Dutch governmental organizations. In this thesis, we recommend that the scope of use of DigiD is extended to the private sector, since we think that such an initiative can result in potential benefits for all parties involved in the system; we will refer to the extended version of DigiD, as DigiD-2. As a result, we also indicate that the extension of DigiD to the private sector is not a trivial task, due to the issues that arise in such an extension. Consequently, several ways of solving these issues are suggested throughout the thesis. Further, several new features are suggested to be implemented in the new version of DigiD. Also, we study the current kernel of DigiD, which is a SSO system called A-Select, and argue why we consider it beneficial, both from functionality and security perspectives, to replace the current kernel of DigiD, with a browser based federated identity management system, called Security Assertion Markup Language 2.0, and use it as a new kernel for DigiD-2. Finally, we give a description of DigiD-2, in the context of SAML 2.0.

Acknowledgements

This thesis is a result of my 6 months internship at the PricewaterhouseCoopers advisory, and it concludes my two years of Master's studies at the Technische Universiteit Eindhoven (TU/e), with specialization in Information Security Technology. There are several people who I would like to thank for their assistantship during my internship. First of all, I would like to thank my supervisors, Eric Verheul, and Berry Schoenmakers, for their committed guidances, many insightful discussions and different critical remarks. Eric is further appreciated for allowing me to work on this topic, and for letting me to conduct this research as part of an internship at the PricewaterhouseCoopers advisory. Further, I would like to thank my personal coach at PricewaterhouseCoopers, Cas de Bie, for his useful administrative and personal advices. My colleagues at PricewaterhouseCoopers are also appreciated, for making my stay during the internship interesting; Jeroen Prins and Pieter Siekermann are further appreciated for providing me with useful comments related to my thesis. I would also like to thank Bart Jacobs, for being a member of my assessment committee, and Wim Kegel for his answers on my questions related to DigiD. Last but not least, I would like to thank TU/e scholarship foundation, for providing me with a full two year scholarship which had allowed me to continue my Master's studies at the Technische Universiteit Eindhoven.

(This page is intentionally left blank.)

Contents

Chapter 1

Introduction

The goal of this chapter is to give the reader information on the research that was performed for this thesis, and highlight motivations for the overall research. More concretely, in Section 1.1, background information on concepts like Identity Management (IDM), Single Sign On (SSO) systems, and Federated Identity Management (FIM) are given. In Section 1.2, we describe an SSO system: DigiD[1], used by Dutch governmental organizations for authenticating their citizens online. Further, in Section 1.3, we discuss why it can be reasonable to extend DigiD to the private sector, and describe several technical problems that potentionally arise in such an extension. In Section 1.4, we mention that due to the new context of DigiD, it is reasonable to select a new kernel for DigiD, namely, a browser based FIM system, called Security Assertion Markup Language 2.0. Finally, in Section 1.5, we discuss the contributions and organization of the thesis.

[1] It should be indicated that DigiD is currently based on another SSO system called A-Select [38].

1.1 Background Information

1.1.1 Identity Management (IDM)

A digital identity of an individual can be represented by a set of different identity attributes [23]. Such attributes typically contain various statements regarding personal, qualification, or other information of the individual in question. Also, each individual may have different digital identities in different electronic contexts. Further, the lifecycle of a digital identity typically consists of the following processes: creation, storage, maintenance, and deletion of digital identities. The term IDM, is related to managing the lifecycle of users' digital identities, and enforcement of access control rights on such identities, with respect to available resources. Consequently, various tools and IDM systems are used nowadays, in order to implement different IDM related processes in an efficient, effective and interoperable manner. Some examples of such tools are: Computer Associates' Identity Manager [57], and IBM Tivoli Identity Manager [58].

1.1.2 Single Sign On (SSO)

A Single Sign On system is a popular means of user authentication, and can be used as part of the overall identity management process, within a single organization, or across multiple organizations. The major benefit of an SSO system, is that a user may authenticate once with an authentication server (i.e., identity provider IdP), and during the validity of the authentication session, access various applications (i.e., service providers SP) without the need to re-authenticate. Such functionality is typically achieved in the following manner. An entity called an IdP, will authenticate a user, and generate an authentication assertion, which contains information related to the authentication status of the authenticated user. Such an assertion is then delivered to the SP, who processes it and grants access to the holder of the assertion, without the need to re-authenticate. Further, if an SSO system is deployed in a closed environment, we will refer to such an SSO system, as an enterprise SSO system [56]. Some examples of enterprise SSOs are: Computer Associates' eTrust Single Sign-On [59] and Evidian's AccessMaster [60]. If an SSO system is deployed in an open environment (e.g., internet), we will refer to such a system, as a regular SSO system [56]; or simply an SSO system. Some examples of SSO systems are: Kerberos [55], Microsoft Passport [62], A-Select [39], DigiD [43].

1.1.3 Federated Identity Management (FIM)

The term FIM is related to managing users' identities across several organizations. That is, by using a FIM system, different organizations (e.g., IdPs), may exchange various identity related information about their users, with other organizations (e.g., SPs). The exchange of information, similarly to an SSO system, is performed using assertions. However, the main differences of a FIM system, as compared to an SSO system, are twofold: (1) using a FIM system, the exchange of user related information is not limited only to authentication information, and it typically consists of personal, qualification, or other relevant information, and (2) as it has been highlighted, an SSO can be deployed within a single organization as well (e.g., an enterprise SSO system), however, a FIM system, is typically used between various organizations. Clearly, using a FIM system, SSO functionality can be easily achieved. To sum up, the main benefit of a FIM system is that it allows different organizations to benefit from the identity related information that a partner organization has about different users.

Further, several FIM systems exist nowadays, and they can be mainly classified in two types [24]: (1) Browser Based FIM systems: systems, where generated assertions have a short life time, and can be exchanged between identity and service providers, using typical browsers, and (2) Private Credential Systems: systems, where generated assertions have a long life time, are cryptographically protected, and are stored on the local machines of corresponding users; the users then use their local software, in order to communicate such assertions to the service providers. Examples of browser based FIM systems are: Security Assertion Markup Language [7], Liberty Alliance ID-FF [9], and WS-Federation [12]. Some examples of private credential systems are: Brand's Credentials [21], Chen's Credential System [17], and Idemix [30]. Finally, it should be indicated that there may exist two types of deployment of FIM systems [23] in general. (1) A Centralized FIM system: that is, a FIM system where only a single identity provider, and various service providers are involved in transactions, or a (2) Distributed FIM system: that is, a FIM system where several identity providers, and various service providers are involved in transactions.

1.2 DigiD

DigiD is an SSO[2] system, used by Dutch governmental organizations, for authenticating their citizens in a single sign on manner. That is, once a user gets authenticated by DigiD, she can access the web-site of the specified governmental organization without the need to re-authenticate. DigiD is currently based on an open source SSO system, called A-Select [39]. A typical operation of DigiD is the following. After authenticating a user, DigiD, using an authentication assertion, informs the corresponding governmental organization about the result of the authentication and the BSN[3] number of the authenticated user. The governmental organization then uses the received BSN number, as a key in the citizen's database (GBA), in order to know who was authenticated and engage in an online transaction with her; note that sometimes users can also themselves supply their identity related information to organizations. Some examples of Dutch governmental organizations that currently use DigiD are: The Social Insurance Institute, the Centre for Work and Income, the Tax Authorities, and Donorregister.

1.3 Extending DigiD to the Private Sector: DigiD-2

1.3.1 Benefits

Currently, the scope of the organizations that can use DigiD is only limited to Dutch governmental organizations. As a result, it becomes reasonable to start thinking of extending DigiD to the private sector; note that we will refer to the new version of DigiD, as DigiD-2. Such an extension could result in the following benefits. (1) Increased usability of the system for the users: due to the SSO functionality, (2) reduced costs for organizations: due to outsourcing the authentication procedures to DigiD-2, and last (but not least) (3) better fraud detection and prevention activities: private sector could notify Dutch government, about the identifiers of the users suspected in fraud, which the latter could use for conducting necessary legal activities with them. As a supporting argument for the latter statement, note that The Dutch Association of Banks suggested[4] the usage of BSN numbers as their clients identifiers for better fraud and terrorism investigations.

[2] As it will be elaborated in Section 4.2.3 DigiD currently, does not strictly satisfy all requirements of an SSO system.

[3] A BSN number is a Dutch social security number; previously it was also called a SOFI number.

[4] Clearly, such a request could be satisfied without directly providing banks with BSN numbers, but rather with pseudonyms. Dutch government would then still be able to link the supplied pseudonyms with corresponding BSN numbers.

Additionally, if DigiD-2 were extended to a centralized FIM system, DigiD-2 could supply the private organizations with personal information of her users by embedding them into corresponding assertions.

1.3.2 Issues

Before the benefits of DigiD-2 can be evaluated, it needs to be indicated, that currently DigiD can not be directly extended to the private sector, for at least following two reasons. (1) As it was highlighted in Section 1.2, Dutch governmental organizations use the received BSN numbers, as a key in the citizen's database (GBA), in order to know who was authenticated, and engage in an online transaction with her if necessary. However, due to Dutch regulations, currently, Dutch private organizations are not allowed to use the BSN numbers[5] of Dutch citizens. In order to avoid this issue, DigiD-2 could supply the requesting organizations with identifiers of her users that are different from regular BSN numbers (e.g., pseudonyms). (2) Even if DigiD-2 starts to use new identifiers of her users instead of their BSN numbers, organizations still need a means to (once) relate the identity of the authenticated user with the newly supplied identifier (e.g., pseudonym). Equivalently, this would mean that they need to somehow, relate the received identifier with the local account of the corresponding user.

1.4 SAML 2.0: A New Kernel for DigiD-2

Security Assertion Markup Language 2.0 is an XML based open standard from OASIS [7], used to achieve browser based federated identity management scenarios. It is a combination of SAML 1.1, and other browser based FIM standards, like Liberty Alliance ID-FF 1.2, and Shibboleth. In SAML 2.0, one organization, for example a service provider may ask another organization, for example an identity provider, to authenticate a user and return the result of the authentication together with user identity related information to her. Such information is exchanged using assertions, which are XML messages, and contain various types of identity related information about a user. For instance, a SAML 2.0 assertion may contain the following information: "A user with pseudonym: X, was authenticated at time: Y, with authentication mechanism: Z, and his membership status is: Q". Clearly, a service provider can use such an assertion, in order to grant access to the owner of such an assertion, without the need to re-authenticate. Also, by using SAML 2.0 assertions, one could also achieve features like exchange of users' attribute related information, instead of only exchanging authentication related information. Finally, using SAML 2.0 DigiD could be easily extended to a centralized FIM system, as suggested in Section 1.3.1.

1.5 Contributions and Organization of the Thesis

First of all, we suggest extending DigiD to the private sector. That is, to also allow Dutch private organizations to use DigiD as an authentication framework. Further, we suggest that DigiD supports transportation of users' identity related information to the requesting organizations, and also, we suggest usage of pseudonyms instead of BSN numbers as alternate user identifiers (in order to avoid BSN number related issues). Also, we suggest that DigiD-2 becomes a browser based centralized FIM system. Finally, the thesis has following concrete contributions.

[5] Nor are they allowed to use the citizen's database, for which the BSN is the key.

- **Chapter 2: Open Standards for Browser Based FIM.** In this chapter, we give a technical description of the existing browser based FIM systems, like SAML 2.0, WS-Federation, and Liberty Alliance. Further, we compare these systems between each other, and provide discussions about their differences. Finally, we conclude with some comments related to the architectures of these systems, and indicate that in some situations it may be more reasonable to use SAML 2.0 as an underlying FIM architecture as compared to the other browser based FIM systems.

- **Chapter 3: Private Credential Systems and Pseudonyms for DigiD-2**. In this chapter, we first survey some of the major existing private credential systems, and compare them between each other. Further, we compare the architectures of browser based FIM systems, with those of private credential systems, and discuss why we consider that a browser based FIM system best fits the architecture of DigiD-2 as compared to a private credential system. Also, we investigate several mathematical techniques for generating pseudonyms in DigiD-2, and identify some requirements that these pseudonyms should satisfy. Finally, some discussion on the pseudonym generation techniques is provided together with explanation on why we consider that generated pseudonyms should be based on BSN numbers.

- **Chapter 4: DigiD/A-Select and Other SSO Systems.** In this chapter, we give a background information related to the architecture of A-Select and current DigiD, and provide two single sign on use cases, one with general and another one with concrete parameters. Further, we discuss the functional and security limitations of current DigiD. Also, we briefly describe other existing single sign on systems, like Kerberos [55] and Microsoft Passport [62], and discuss differences of such systems with respect to each other.

- **Chapter 5: Extending DigiD to the Private Sector (DigiD-2)[6].** In this chapter, we firstly argue why it can be beneficial to extend DigiD to the private sector. Next, we describe DigiD-2 and explain how organizations can link supplied pseudonyms with local accounts of their corresponding users. Also, we identify functional and non functional requirements that DigiD-2 should satisfy. Further, together with the arguments mentioned before, we argue why it is reasonable to particularly use SAML 2.0 as a new kernel for DigiD-2. Also, we give a specification of a typical DigiD-2 SSO use case, by using SAML 2.0 Name Web SSO Profile and Artifact Binding, and show techniques how DigiD-2 could allow users to choose, which attributes they would like to release to requesting service providers.

- **Chapter 6 and Appendix A.** Finally, we provide concluding remarks of the whole thesis in Chapter 6 and describe several open issues that remain and provide directions for future research. Also, in Appendix A, we give several mathematical preliminaries, that can be useful, in order to understand surveyed private credential systems, and suggested pseudonym generation techniques. Mathematical areas covered in the Appendix include Number Theory, Group Theory, Cryptography, and Elliptic Curve Cryptography.

[6] Aside: We also specify an SSO use case, which allows users, in a browser based FIM model, to conduct necessary internet transactions without the fear of being traced by an identity provider. To the author's best knowledge, such a technique, has not been mentioned anywhere in the existing literature.

Chapter 2

Open Standards for Browser Based Federated Identity Management

This chapter describes several existing federated identity management standards, which are used to implement browser based FIM systems. Main organizations involved in the development of such standards are: OASIS[7] and Liberty Alliance[8]. The main federated identity management standards that have been established or are in the process of establishment by these organizations are listed below: (1) Security Assertion Markup Language (SAML) versions 1.0, 1.1, and 2.0, (2) Liberty ID-FF (Identity Federation Framework), and (3) WS-Federation. Hence, this chapter is organized in the following way. In Section 2.1 we will describe the architecture of SAML 2.0, followed by Section 2.2 that briefly surveys the main differences between SAML 2.0, SAML 1.1, and Liberty ID-FF. Section 2.3, will describe specifications introduced by Microsoft and IBM for achieving security of web services, with a focus towards federated identity management system, called WS-Federation. Finally, in Section 2.4, the chapter will be concluded by discussion and comments related to the comparison of architectures of above listed systems.

[7] OASIS is a non-profit consortium established in 1993 that drives the development, convergence and adoption of open standards. http://www.oasis-open.org/

[8] Liberty Alliance is a non-profit organization, formed in 2001 and focused on establishment of open standards, for identity management. http://www.projectliberty.org/

2.1 Security Assertion Markup Language (SAML 2.0)

2.1.1 Introduction

The Security Assertion Markup Language (SAML) defines an XML based framework that enables different business entities, to communicate user authentication, entitlement, and attribute information between each other. It consists of the following four major components. (1) Assertions: XML packages containing authentication, entitlement, or attribute information about users, (2) Protocols: used to define request/response XML messages that SAML entities can use for achieving particular objectives, (3) Bindings: used to define how communication protocols (e.g., HTTP, SOAP) can be used to transport SAML request/response messages between different SAML entities, and (4) Profiles: used to combine above listed components, in order to support particular use cases.

The following entities are mainly involved in a typical SAML interaction: (1) Asserting party: an entity that makes SAML assertions about users, (2) Relying party: an entity that receives and processes assertions made by the asserting party, and (3) User: an entity about whom assertions are made. In most of the cases, the user will be a human entity, but in other cases, it can also be a company or a computer. Note that throughout this chapter, we will refer to the asserting party as an identity provider (IdP: which is used to for authenticating principals), and to the relying party as a service provider (SP: which is used for offering various services to its users). Further, there have been several business use cases that have motivated the development of SAML 2.0; these business use cases will be (on a high level) described in the next section.

2.1.2 Major SAML 2.0 Use Cases

2.1.2.1 Single Sign On (SSO)

In this use case, there are two organizations: Organization A, acting as an identity provider (IdP), and Organization B, acting as a service provider (SP). Also, there is a user named Joe Doe, who is registered at both organizations. It is assumed that Organization A and Organization B, prior to using SAML, have agreed on a common name identifier for Joe, which is represented as: JoeDoeAB. The goal of this use case is to allow Joe, to authenticate at the identity provider, and during the validity of the authentication session, use services at the service provider without the need to re-authenticate.

1. User Joe visits the Service Provider's web-site, in order to use some services. From SP's web-site, he is redirected to the identity provider (IdP) for authentication.
2. After successfully authenticating himself to the IdP, an assertion is generated by the IdP stating that: "The user: JoeDoeAB was successfully authenticated." This assertion is delivered to the service provider.
3. The SP, after receiving this assertion, in order to know who was authenticated, links the identifier JoeDoeAB to its local account, which corresponds to user Joe Doe. Finally, the SP allows Joe to access requested service without the need to re-authenticate.

Note that the high level scenario described above refers to a use case where the user had first accessed the SP, and then was redirected to the IdP for authentication. In SAML, such a use case is called an SP initiated web SSO scenario. SAML 2.0 also supports IdP initiated web SSO scenarios, where the user is first authenticated to an IdP, and then, using for example a link located at the IdP's web-site, is directed and authenticated to the SP in an unsolicited manner.

2.1.2.2 Identity Federation Using Persistent Pseudonyms

As in the previous use case, there are again two organizations: Organization A, acting as an identity provider (IdP), and Organization B, acting as a service provider (SP). Also, there is a user named Joe Doe, which is registered at both organizations. However, *Joe* has different local accounts at each organization. At Organization A he is registered as user: JoeA, and at Organization B, as user: JoeB. The goal of this use case is to allow Joe, to dynamically federate (link) his accounts between these organizations using persistent pseudonyms, and at the same time be able to use the single sign on feature described in the previous section.

1. User Joe visits the service provider's web-site, in order to access some services. From SP's web-site, he is redirected to the identity provider (IdP) for authentication.
2. After successfully authenticating himself with the IdP, Joe is asked whether he wants to federate his identity with the SP. After Joe's consent, the IdP generates a new identifier for Joe: azhqu67 and generates a SAML assertion, which indicates to the SP that name identifiers should be federated, and stating that: "The user: azhqu67 was successfully authenticated." This assertion is delivered to the SP.
3. After the SP receives such an assertion, needs to find out to which local account, azhqu67 corresponds to. But, since, this is the first time that the SP has seen this identifier she can not directly relate this identifier with any of her local accounts.
4. The SP, asks Joe, to log into the system using his local (JoeB) account. After successful authentication, his account is linked with the new identifier: azhqu67. Clearly, in the future, Joe will not need to re-authenticate, and the SP will be able to identify him from the supplied identifier: azhqu67, thus maintaining the single sing on experience.

It should be noted that SAML 2.0 supports new type of name identifiers with privacy preserving characteristics, named pseudonyms. These pseudonyms can be of two types: (1) persistent pseudonyms: pseudonyms that are not changed throughout their life cycle (pseudonym described in this use case was persistent), and (2) transient pseudonyms: pseudonyms that are newly generated at each new request. Clearly, SAML 2.0 does not dictate the means how these pseudonyms should be generated, rather it just offers means of transporting them.

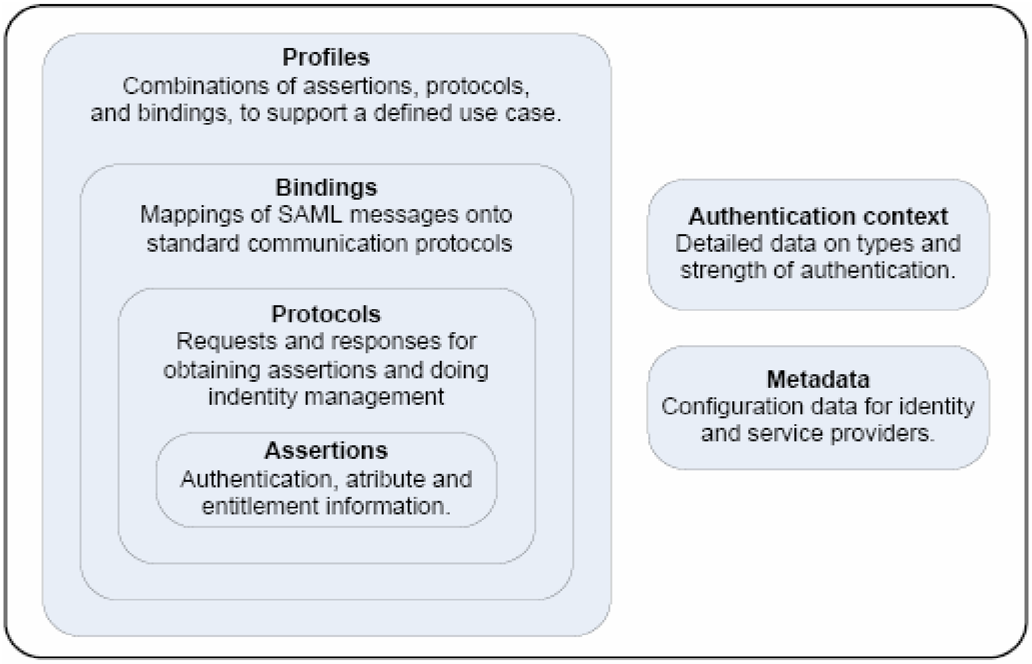
2.1.3 Benefits of Using SAML 2.0

Clearly, the use cases described above, can be achieved using other mechanisms than SAML, but, there are several technical benefits why SAML should be considered advantageous over other mechanisms.

- **Limitations of Browser Cookies**. Before SAML, most of the SSO products have relied on using cookies for tracking user authentication state so that re-authentication is not necessary. But, since browser cookies are not transferred between DNS domains, identity and service providers were limited to being located in the same DNS domain.
- **SSO Interoperability.** Various SSO products were implemented in different proprietary ways. So, organizations willing to support a cross domain single sign on had to use the same product in order to achieve interoperability.
- **Web Services**. There are various standards in web services based applications that have no standard way to convey security related attributes between different domains. SAML assertions can be used by other web applications as a packet of security related information. For example, as it will be discussed in the final section, WS-Security uses SAML assertions as its security tokens.

2.1.4 SAML 2.0 Architecture

The main components of SAML 2.0 are: assertions, protocols, bindings and profiles. Additionally, SAML supports following two components. (1) Authentication context: used to describe the type of authentication a particular user used while authenticating, and (2) Metadata: defines ways to express and share configuration information between SAML parties. The figure below illustrates the relationship between these components.


Profiles
Combinations of assertions, protocols, and bindings, to support a defined use case.
Bindings
Mappings of SAML messages onto standard communication protocols
Protocols
Requests and responses for obtaining assertions and doing indentity management
Assertions
Authentication, atribute and entitlement information.
Authentication context
Detailed data on types and strength of authentication.
Metadata
Configuration data for identity and service providers.

[Figure 1]. Basic SAML 2.0 concepts and their relationships; based on a figure from [7].

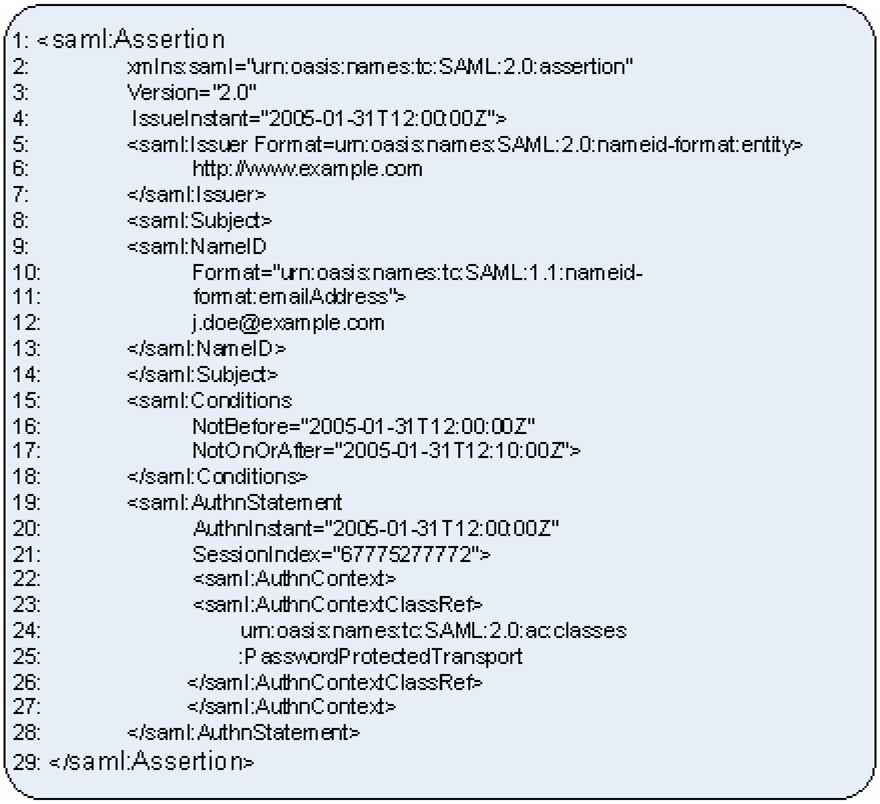
2.1.4.1 SAML 2.0 Assertions

A SAML assertion ([Figure 2]) is an XML message that typically contains user identity related information. The outer structure of the assertion is generic, and provides information that is common to all of the statements within it. This (generic) information can for example be: information about the issuer of the assertion, name of the subject, conditions used to validate assertions, and a signature used for the integrity and authenticity properties of the assertion. SAML 2.0 defines three types of assertions

1. **Authentication Assertions**. These types of assertions describe the details of the authentication, like the method used for authentication, and time when the authentication had occurred. For example, an assertion can indicate that Joe was authenticated using two-factor authentication, at time 13.30, GMT +01:00, date 01/01/99.
2. **Attribute Assertions**. These types of assertions describe different attributes that an asserted subject is associated with. The attributes can be any type of text describing some property of a user. For example, an attribute assertion could say that a user Joe is above 21 years old, or that user Joe is a Gold member of some company.
3. **Authorization Decision**. These types of assertions describe whether a user is entitled to perform some action. For example, whether Joe is permitted to buy a specific item, at some internet shop.

Following figure shows an example of a single SAML 2.0 authentication assertion (lines 1-29), which states that a given user was authenticated by the identity provider using a password based authentication mechanism. More details on the given assertion are described below.

1. **Lines (1-4).** Describe the declaration of the SAML assertion, by indicating the SAML assertion namespace (line 1), the version of the assertion (line 2), and the date when the assertion was issued (line 3).
2. **Lines (5-7).** Describe the issuer of the assertion (e.g., identity provider), which in our case is represented as http://www.issuer.com.
3. **Lines (8-14).** Describe the subject of the assertion, which in our case is represented using an email address (line 9), that has the following value: j.doe@example.com (line 10).
4. **Lines (15-18).** Describe the time period during which the given assertion is valid.
5. **Lines (19-28).** Describe the authentication statement, which states that a given subject was authenticated at date: 2005-01-31T12:00:00Z, using the password authentication mechanism (line 21).



```
<saml:Assertion
        xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
        Version="2.0"
         IssueInstant="2005-01-31T12:00:00Z">
        <saml:Issuer Format=urn:oasis:names:SAML:2.0:nameid-format:entity>
            http://www.example.com
        </saml:Issuer>
        <saml:Subject>
        <saml:NameID
            Format="urn:oasis:names:tc:SAML:1.1:nameid-
            format:emailAddress">
            j.doe@example.com
        </saml:NameID>
        </saml:Subject>
        <saml:Conditions
            NotBefore="2005-01-31T12:00:00Z"
            NotOnOrAfter="2005-01-31T12:10:00Z">
        </saml:Conditions>
        <saml:AuthnStatement
            AuthnInstant="2005-01-31T12:00:00Z"
            SessionIndex="67775277772">
            <saml:AuthnContext>
            <saml:AuthnContextClassRef>
               urn:oasis:names:tc:SAML:2.0:ac:classes
               :PasswordProtectedTransport
           </saml:AuthnContextClassRef>
           </saml:AuthnContext>
        </saml:AuthnStatement>
</saml:Assertion>
```

[Figure 2]. An example of a single SAML 2.0 authentication assertion.

2.1.4.2 SAML 2.0 Protocols

SAML 2.0 supports a number of protocols, defined in terms of request/response XML messages, that allow service and identity providers to achieve certain objectives. The initiator of the request in the protocol is referred to as a SAML requester, while the responder party is referred to as a SAML responder. SAML requesters inform SAML responders about the protocol they plan to initiate, by including corresponding XML request elements in the transported XML messages; similarly, SAML responders return corresponding response messages, as XML response elements. Note that in the protocol descriptions below, only the XML elements used to identify corresponding protocol messages are described, while other details like element attributes and other message contents are omitted.

- **Authentication Request Protocol**. Requests that an identity provider authenticates a principal and returns the corresponding assertion. The request is issued using the <AuthRequest> element. The successful <Response> will contain an assertion about the authenticated principal.
- **Single Logout Protocol.** Used to request near-simultaneous logout of all active sessions associated with a principal. The logout can be initiated by the user, service provider, or identity provider. The request is issued using the <LogoutRequest> element. And the requester will receive a <LogoutResponse> element as an indication of a successful logout.

- **Artifact Resolution Protocol.** Provides a mechanism by which SAML protocol messages may be passed by reference, using a small, fixed-length value called an artifact. These artifacts can be later dereferenced in order to retrieve actual messages. The artefact receiver uses the <ArtifactResolve> element, to ask the protocol requester to dereference the artifact. A successful <Response> message will contain a corresponding protocol message.
- **Name Identifier Management Protocol.** Used by the identity providers to inform the service providers, about the change, or termination of a name identifier for a particular principal. The request is issued using the <ManageNameIDRequest> element. A successful <Response> element will inform about the success of the performed operation.
- **Name Identifier Mapping Protocol.** When a service provider shares a name identifier about a principal with an identity provider, and wants to request such an identifier in a particular format, or for a particular federation namespace, it can use the name identifier mapping protocol. The request is issued using the <NameIDMappingRequest> element. A successful <Response> element will indicate the success of the performed operation, and contain the requested identifier.
- **Assertion Query and Request Protocol**. Used to request assertions from the asserting party. The assertions can be requested based on various criteria:
    - If the requester knows the unique identifier of a particular assertion, then it can use the element <AssertionIDRequest>. A successful <Response> message will contain the requested assertion.
    - If the requester wants to know what authentication assertions are already available for a particular user, he can use the <AuthQuery> element. A successful <Response> message will contain such assertions.
    - If the requester wants to specify authentication context requirements of authentication statements returned in response to a request or a query, he can use the <RequestedAuthContext> element.
    - If the requester wants to request a particular, or all attributes of a user, he can use the <AttributeQuery> element. Successful <Response> will contain the requested attributes of a user.

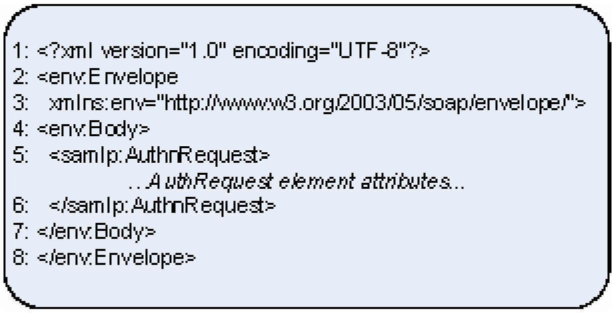
2.1.4.3 SAML 2.0 Bindings

SAML 2.0 bindings define how SAML entities can exchange various SAML 2.0 protocol messages between each other using standard communication protocols. Message exchanges between these entities can be performed either directly, or via an intermediary. For scenarios where SAML entities want to exchange such messages between each other directly, a SOAP binding can be used. While for scenarios where SAML 2.0 entities, like SP and IdP, want to exchange request/response messages between each other via an intermediary, such as a user, bindings like HTTP redirect with GET, POST or Artifact can be used. Note that in each of the remaining sections, before the actual bindings will be described, some background information relevant to them will be given.

2.1.4.3.1 SOAP Binding

SOAP (Simple Object Access Protocol) is a (request/response) protocol, used for exchanging XML based information between different computers using standard communication protocols like HTTP. SOAP messages are basically XML documents, that are embedded in a SOAP envelope which consists of a SOAP header (infrastructure data), and SOAP body (application data). For more information about SOAP, an interested reader is referred to [14].

The SAML SOAP binding describes how SAML 2.0 protocol (request/response) messages can be embedded inside SOAP messages. The basic operation of this binding consists of the following two steps: (1) A SAML requester sends a SAML request element ([Figure 3) to the SAML responder, embedded in the body of a SOAP message, (2) the SAML responder, returns either a SAML response element embedded in the body of a SOAP message, or generates a SOAP fault. For example, the next figure shows a scenario, where a SP sends a <AuthRequest> element (used in the authentication request protocol) to the IdP, embedded in the body of a standard SOAP message; note that since the figure is used to highlight how a SAML protocol message can be embedded into a SOAP message, the actual attributes of such a message are omitted, but an interested reader is referred to [1] for the full picture. Clearly, any other SAML protocol message can be embedded into other SOAP messages in a similar way.



```
<?xml version="1.0" encoding="UTF-8"?>
<env:Envelope
  xmlns:env="http://www.w3.org/2003/05/soap/envelope/">
<env:Body>
  <samlp:AuthnRequest>
            ...AuthRequest element attributes...
  </samlp:AuthnRequest>
</env:Body>
</env:Envelope>
```

[Figure 3]. A sample <AuthRequestElement> embedded inside a SOAP message.

2.1.4.3.2 HTTP Redirect POST and GET Bindings

HTTP is a basic request/response based protocol, used to exchange various types of resources between a client (e.g., web browser) and a server (e.g., web server) over the internet. Typically, a client establishes a TCP connection with the server, and issues a request that consists of following three components: (1) HTTP request or response Line: The request line defines a method that specifies whether a client wants to submit or retrieve a resource from the server, whereas the response line, reports the status of a submitted request; clearly depending on the type (e.g., request or response) of the transaction, a corresponding line is used. The most frequently used method for requesting a resource from a web service is the GET method, which takes the URL of a requested resource as a parameter. On the other hand, the POST method is used to submit data from a client to the server, or vice versa. For example, a web server may return a requested (e.g., using GET) HTML page to the client, using the POST command.

(2) HTTP Header Line: The header line contains several headers used to describe various information about corresponding requests or responses. For example, a web browser may use the Accept Header to inform the web service about specific media types which are acceptable for her. The Location Header can be used by a web server, for redirecting a user to a different web site, by embedding the URL of the new web site inside it. Clearly other types of headers are supported as well, for a full list of available HTTP headers, an interested reader is referred to [15] (3) HTTP Body. The body of an HTTP transaction contains the actual data that is exchanged between communicating parties. For instance, if the browser wants to submit some data contained in the HTML form to the web server, then this data is embedded in the body of the corresponding HTTP transaction, and is sent to the web service by using the POST method.

The main goal of both SAML Redirect GET and POST bindings, unlike SOAP binding, is to enable information exchange between IdPs' and SPs' (or vice versa), using an intermediary (e.g., the user). Below, the basic operation of both of these bindings is described; note that since both bindings are functionally the same, the author decided to merge them into a single representation.

1. The user makes an HTTP request to the SAML requester, in order to access some service. Based on the request, the SAML requester decides to initiate a SAML protocol with the SAML responder, using the user as an intermediary.
2. The SAML requester, generates a SAML request, and sends it to the user, encoded as a (1) URL in the HTTP location header: in case of a GET binding, or as a (2) HTML document containing a form: in case of a POST binding.
3. The user, after receiving the request, forwards it to the SAML responder, by issuing a (1) HTTP GET request: in case of GET binding or (2) HTTP POST request: in case of a POST binding, on it. Note that this step can also be automated, by performing an automatic redirection in the first case, or auto-submitting the form using JavaScript in the second case.
4. The SAML responder, after receiving the request, generates a SAML response, and sends it to the user, encoded in the same way as in the second step of the protocol. The user, similarly to the third step of the protocol, delivers this response back to the SAML requester.
5. The SAML requester, upon receiving the SAML response, delivers the requested service to the user, by returning an arbitrary response to her.

2.1.4.3.3 Other Bindings

- **HTTP Artifact Binding [1].** In this binding, similarly to the previous two bindings, SAML messages are exchanged between SAML entities like, IdP and SP, via an intermediary such as the user. However, instead of transmitting the actual protocol messages, SAML entities exchange references (i.e., artifacts) to these messages, using HTTP redirect or POST bindings. The transmitted references are then used by SAML entities in order to retrieve the actual messages, using a direct SAML binding, like SOAP for example. The motivation for using such references is that an assertion encoded as an URL, can be rather long, which can be a problem for browsers that set different limitations on URL lengths.

- **PAOS and URI.** Two other bindings are also supported in SAML 2.0, but are not reviewed in this section: (1) PAOS binding: Defines a multi-stage SOAP/HTTP message exchange that permits an HTTP client to be a SOAP responder, and (2) URI binding: Defines a means for retrieving an existing SAML assertion by resolving a URI (uniform resource identifier). An interested reader is referred to [1] for more details on this binding.

2.1.4.4 SAML 2.0 Profiles

SAML profiles define how assertions, protocols and bindings can be used in order to support particular use cases. Additionally, certain types of profiles specify a set of rules describing how to embed SAML assertions into and extract from a protocol or binding. Examples of such profiles are: (1) Artifact resolution profile, (2) Assertion query request profile, (3) Name identifier mapping profile. These profiles, define how to use corresponding protocols (described before) and assertions together with SAML binding like SOAP.

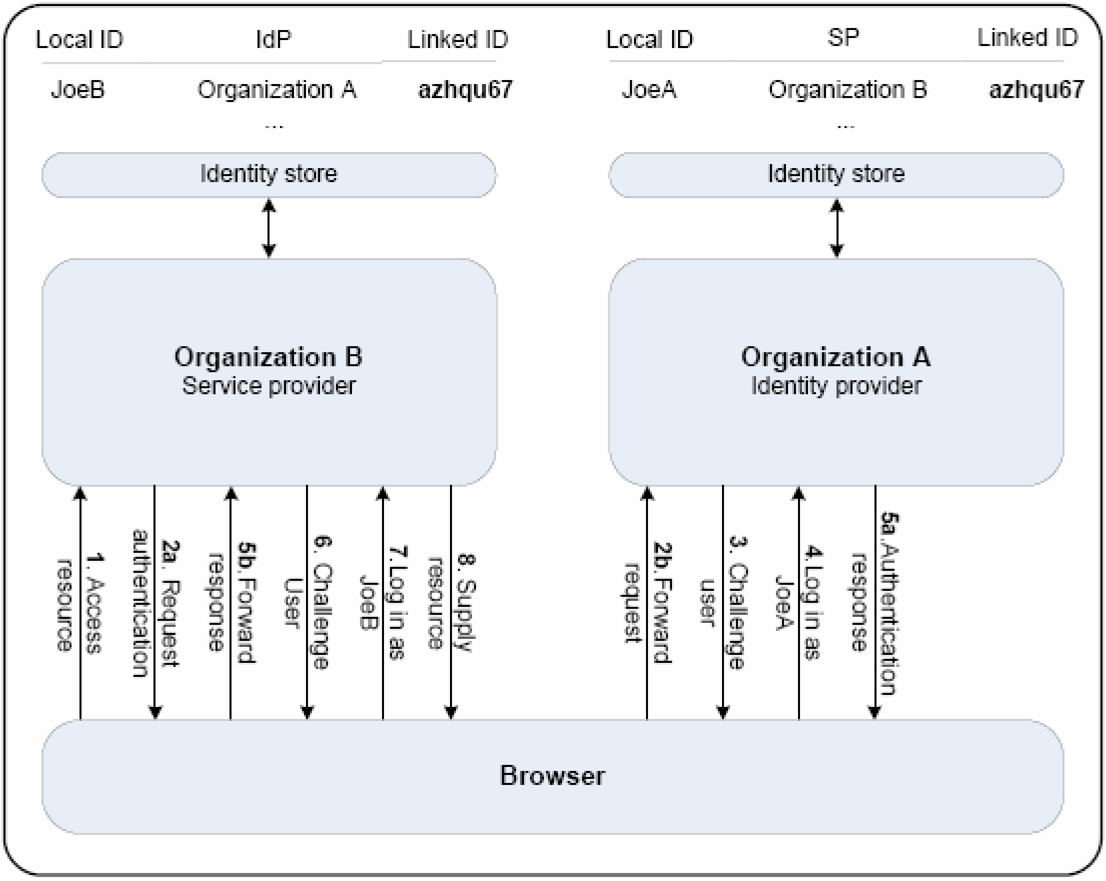
On the other hand, the profiles that are used to support particular use cases, may require the use of specific SAML functionality (e.g., attributes, conditions, or bindings), or define the processing rules to be followed by profile actors. Examples of such profiles are:

- **Web SSO Profile.** This profile describes how SAML assertions, protocols, and bindings can be combined, in order to support single sign on and federation scenarios. Depending on the bindings used, this profile can be split into different but equivalent profiles. A particular example of such a profile can be viewed in Section 2.1.4.4.1.
- **Identity Discovery Profile**: used to describe how service providers can use common domain cookies to enable discovery of identity providers to whom a particular user corresponds to.
- **Single Logout Profile:** Together with the logout protocol, this profile details how logout can be initiated by different parties, and based on that, puts several constraints on the contents of exchanged messages.
- **Name Identifier Management Profile.** Together with the name identifier management protocol, this profile describes how above mentioned protocol can be used with different bindings, and based on that, puts some constraints on the contents of the exchanged messages.
- **Attribute Profile.** This profile defines how SAML assertions can embed already existing user attribute profiles, like X.500/LDAP or XACML attribute profiles.
- **Enhanced Client Proxy Profile.** This profile describes how users can use proxies, as intermediaries in SAML interactions, like web single sign on for example.

2.1.4.4.1 Identity Federation Using Pseudonyms and HTTP Redirect Binding Profile

Below is a SAML 2.0 Web SSO profile that refers to the federation use case described in the beginning of this chapter (Section 2.1.2.2). This profile is defined in terms of HTTP redirect binding (either GET or POST). Clearly, other bindings can be applied to this profile, based on which, steps involving message exchanges between SAML entities will be changed accordingly.

1. The user Joe accesses the SP's web-site, in order to use some services. The SP, in order to grant the user access to these services, requests that she is authenticated at the identity provider.
2. In order to request such authentication, the SP initiates the authentication request protocol, by sending a message to the identity provider, containing the <AuthRequest> element. Since this profile uses HTTP redirect binding, the authentication request message is first sent to the user, encoded in HTTP location header. The user then forwards this message to the IdP by issuing a HTTP GET command on it.
3. The IdP, after receiving the authentication request sent by the SP, decides to authenticate the user by challenging her for credentials.
4. The user provides the IdP with valid credentials, and manages to login with her local account: JoeA.
5. After authenticating the user, the IdP asks Joe if he wants to federate his identity with Organization B. After Joe's consent, the IdP creates a persistent pseudonym for Joe: azhqu67, and builds a SAML assertion indicating that user with pseudonym: azhqu76 was authenticated. Then, using the name identifier management protocol, the IdP, generates a authentication response message, containing above mentioned authentication assertion indicating that the SP should federate account related to the azhqu67 pseudonym with the IdP. This request, similarly to the second step of this profile, is sent to the service provider, via the user, using the HTTP redirect binding.
6. The SP, after receiving the request message sent by the IdP, uses the supplied name identifier, in order to determine whether a previous federation with such user has been established. If a previous federation has been established (because the name identifier maps to a local account) then go to step 8. If no federation exists, then the service provider needs to determine to which local account received pseudonym corresponds to. In order to achieve this, the SP, needs to locally authenticate the user, by challenging her for local SP credentials.
7. The user, provides valid credentials to the SP, and is identified by the SP as JoeB. The persistent pseudonym (azhqu67) is then stored and linked with the JoeB account, along with the name of the IdP that created the pseudonym. Clearly, this re-authentication will not be necessary in the future, and the SP, will be able to determine who was authenticated, by looking at the pseudonym: azhqu67.
8. A local logon session is then created for user JoeB and the SP may check if user JoeB has enough permission to access requested services. If JoeB has sufficient permissions, than he can use the requested service.



[Figure 4]. SP initiated identity federation with persistent pseudonyms.

2.1.5 SAML 2.0 Security Considerations

SAML 2.0 provides [4] several mechanisms that can be used to ensure various security properties of the overall SAML 2.0 architecture. More concretely, SAML 2.0 provides guidelines on the secure usage of each component of SAML 2.0 architecture (assertions, bindings, protocols, profiles). In this section, we present a short summary[9] of the major security techniques, which if used, offer effective protection against any component of the SAML 2.0 architecture. The security mechanisms offered by SAML 2.0 are mainly divided in three types: (1) application layer security mechanisms (SSL/TLS), and (2) message layer security mechanisms (XML signature and encryption), (3) ad-hoc mechanisms. First and second categories are mainly used to ensure security properties like confidentiality, authenticity and integrity of transported messages. However, the first category provides application layer security, which means that the security properties that it offers, are no longer maintained as soon as the messages go beyond the application layer, whereas the second category ensures that the security properties that it offers, are maintained throughout the lifecycle of transported messages; also note that these two technologies can be used together, in order to ensure the highest level of security. If SSL/TLS and XML Signature/Encryption is properly used in any SAML 2.0 deployment, then such a system is considered to be secure against most of the identified attacks. Finally, the ad-hoc mechanisms allow embedding of different SAML 2.0 message elements which as shown below can increase security of the overall system in certain situations.

[9] Note that, when SAML 2.0 is deployed in particular context, it is still advisable to look into the security guidelines of the components that are used in such a deployment; such guidelines can be found in [4].

Below, some of the most important ad-hoc security mechanisms used in SAML 2.0 for increasing security level of the overall system are described. For a more detailed list of such mechanisms specific to different components of SAML 2.0 architecture, the interested reader is referred to [4].

- **Assertions.** SAML entities should check the conformance of SAML assertions to their validity conditions (e.g., time periods, one time use assertions, etc.), in order to avoid replay attacks.

- **Protocols**. Since processing of SAML requests is a time consuming task, SAML suggests that SAML responders receive requests only from a limited set of trusted entities, in order to minimize the risk of Denial of Service Attacks; clearly, for this, a mechanism for authenticating SAML requesters needs to be employed, which can be realized using one (or both) of the mechanisms listed in the beginning of this section.

- **Bindings.** Attacks targeted to SAML bindings vary depending on the bindings used. Therefore, security considerations of SAML bindings will be classified according to each binding. (1) SOAP binding: In order to harden for the attackers the insertion (or deletion) of arbitrary SAML response messages, SAML entities should use the <InResponseTo> element, that determines to which request a particular SAML response corresponds to; note that usage of SSL/TLS can also counteract this attack. (2) HTTP Redirect/GET binding: In order to avoid scenarios where an attacker maliciously redirects a user to a resource that disrupts logout procedure, SAML IdP's should display a confirmation dialog, before logging the users out from all active sessions. (3) HTTP POST binding: If an attacker can steal a users SAML response, then she can construct an appropriate POST body herself, and forward it to the SP, and thus impersonate the real user. In order to avoid this, several measures can be taken: (a) If the assertion contains an authentication statement about a subject, the SAML responder could additionally ask the user to prove that she is indeed the stated subject of the assertion, (b) If a received authentication statement includes an element, with the IP address of the user, then the SP, may check this address against the user's real IP address; note that ensuring the confidentiality of transport connection will harden for the attackers the process of stealing an assertion. Also, since the SP obtains bearers SAML assertions, a malicious SP, could use the obtained assertion, in order to impersonate the user, at some new SP site. To avoid this, the new SP, should verify the <Recipient> attribute contained in the SAML response against her identity. Clearly, the integrity of the response message must be guaranteed, else the malicious web-site can change the <Recipient> value as desired. (4) HTTP artifact binding: In this binding, the major threat, similarly to previous binding, is the stolen artifact, which can be countered in a similar way as in the POST binding.

- **Profiles**. Most attacks that are applicable for various profiles can be countered using the mechanisms described before.

2.2 SAML 1.1 and Liberty ID-FF (Identity Federation Framework) 1.2

2.2.1 Main Differences Between SAML 2.0 and SAML 1.1

SAML 1.1 was a predecessor of SAML 2.0, and its development was mainly driven for solving the web single sign on problem. However unlike SAML 2.0, SAML 1.1 was not intended for supporting the (dynamic) name federation scenario.

The architecture of both versions is mainly the same, in the sense that SAML 1.1 also consists of assertions, protocols, bindings and profiles. However, there are several enhancements that were added to these components in SAML 2.0, and the major ones, will be highlighted here. Note that we will omit administrative and message content (like element and attribute names) related changes here.

- **Assertions.** Assertions are mainly the same in both SAML versions.
- **Protocols.** The only protocol that was supported in SAML 1.1 was the assertion request and query protocol. SAML 2.0 enhances SAML 1.1 protocols, by introducing: (1) Authentication request protocol, (2) Single logout protocol, (3) Artifact resolution protocol, and (4) Name identifier and management protocols. Due to the lack of (2)-nd and (4)-th protocols, SAML 1.1 was not able to offer users the ability to logout from all active sessions simultaneously, nor support use cases related to name identifier federation. Features supported by (1)-st and (3)-rd protocols were achieved in SAML 1.1, by means of profiles though.
- **Bindings.** The only binding supported by SAML 1.1 was the SOAP binding. Scenarios like web single sign on, that might have required user interaction, and correspondingly bindings like HTTP Artifact, HTTP redirect or POST bindings, were addressed separately in corresponding SAML 1.1 profiles. One of the major changes that SAML 2.0 had introduced over SAML 1.1 was to define bindings like SOAP, artifact, HTTP redirect and POST, as separate SAML 2.0 bindings. This way, SAML 2.0 had achieved a “plug and play” mechanism for defining profiles; i.e., where one profile can be defined in terms of different bindings.
- **Profiles.** The only major profile supported by SAML 1.1 was the web single sign on profile. However, this profile was split into separate profiles, depending on the type of transport channel used. These profiles[10] were: (1) Artifact based SSO profile, (2) HTTP GET based SSO profile, and (3) HTTP POST based SSO profile.

  Additionally, since SAML 1.1 supported only a single profile, features that were supported using SAML 2.0 profiles, like: (1) Identity discovery, (2) Ability to encode existing attributes in SAML assertions, (3) Ability to use a proxy for SAML interactions, and (3) Support for SP initiated message exchanges, were not present in SAML 1.1.

[10] As it can be noticed, SAML 2.0 has made its architecture more generic as compared to SAML 1.1, by separating notion of bindings from profiles. Thus in SAML 2.0, first typically some profile is defined (e.g. SSO Profile), and then this profile can be further defined in the context of different bindings (e.g. SSO Profile with GET binding); in SAML 1.1 one would simply have separate profiles defined for different bindings (e.g. directly SSO Profile with GET binding), so no separate notion of a binding would exist (e.g. GET binding).

2.2.2 Liberty Alliance ID-FF 1.2

2.2.2.1 Main Differences between ID-FF 1.2 and SAML 1.1

SAML 1.1 was followed by a Liberty Alliance standard, called ID-FF 1.2. The main motivation for the development of ID-FF was to support dynamic name federation use case scenario, which was not present in SAML 1.1. Liberty ID-FF had introduced such a feature, by allowing operations related to name identifier management; it had also introduced pseudonyms, as separate user name identifiers. Further enhancements of ID-FF related to SAML 1.1 where support for features like: identity provider discovery, single logout protocol and the ability to perform single sign on in the presence of proxy.

2.2.2.2 Main Differences between ID-FF 1.2 and SAML 2.0

SAML 2.0 serves as the convergence of SAML 1.1 and ID-FF, and correspondingly inherits several major features from them. These features are: name federation related operations, single logout, identity discovery, and, proxy enabled interactions. The architecture of ID-FF is also similar to SAML, since is uses: assertions, bindings, protocols and profiles. However, similar to SAML 1.1, ID-FF does not support separate bindings, that would enable it to have profiles defined in a “plug and play” manner. This main change was introduced in SAML 2.0. Also, ID-FF does not support separate attribute profiles that were introduced in SAML 2.0, which enables it to encode already existing user attributes into SAML assertions.

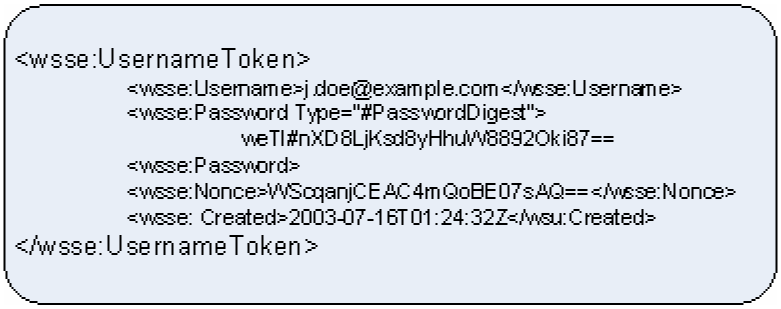
2.3 WS-* Security Framework and WS Federation

2.3.1 Introduction to WS-*Security Framework

Microsoft and IBM have initiated creation of a set of web service security specifications that would allow web services to communicate in a secure and interoperable fashion. This set of specifications cover security technologies, like: authentication, authorization, privacy, trust, integrity, confidentiality, secure communications channels, federation, delegation and auditing across various business environments. These specifications are built upon technologies like: SOAP, XML Digital signatures, XML encryption, and SSL/TLS. The main security approach offered in these specifications is to embed various security properties into transmitted messages, instead of providing security only on the transport layer, as it is done with SSL/TLS for instance. Such an approach would ensure end to end security of transmitted messages that would maintain their security properties even when they are moved beyond the transport layer by a proxy or some other (legitimate) intermediary for example. The specifications mentioned in the beginning are: WS-Security, WS-Trust, WS-Policy, WS-Secure Conversation, WS-Authorization, WS-Federation, WS-Privacy. Focus of this section is the WS-Federation specification, since it deals with managing trust and federating identities across different business domains. It has been approved as a candidate for an OASIS standard, but has not been recognized as such so far. WS-Federation is heavily based on other specifications, like: WS-Security, WS-Trust, and WS-Policy.

2.3.1.1 Definitions

- **Claim.** A claim is a (security) statement made about a user.
- **Security Token.** A security token [Figure 5] is a collection of claims. In a typical interaction it is created for a subject, which needs to introduce it to some web service, in order to access some services or resources. Note that various types of security tokens are supported, like: User Name tokens, X.509 Certificate tokens, and SAML assertion tokens.
- **Security Token Service (STS).** A security token service is a web service that creates and issues security tokens to users.
- **Trust.** Trust is the characteristic that one entity is willing to rely upon a second entity to execute a set of actions and/or to make set of assertions about a set of subjects and/or scopes.



```
<wsse:UsernameToken>
            <wsse:Username>j.doe@example.com</wsse:Username>
            <wsse:Password Type="#PasswordDigest">
                        weTI#nXD8LjKsd8yHhuW8892Oki87==
            <wsse:Password>
            <wsse:Nonce>WScqanjCEAC4mQoBE07sAQ==</wsse:Nonce>
            <wsse: Created>2003-07-16T01:24:32Z</wsu:Created>
</wsse:UsernameToken>
```

[Figure 5]. A User Name token, containing a username, a password, a nonce and the creation date of the token.

2.3.1.2 WS-* Security Framework Components

- **WS-Security.** WS-Security is used as the basis of all above mentioned specifications, and has two main goals: (1) the first goal, is to provide SOAP messages with security properties like integrity, authenticity and confidentiality, by describing how XML signatures and XML encryption headers can be attached to them, (2) the second goal, is to describe how (various) security tokens can be embedded inside SOAP messages. It should be highlighted that WS-Security can use SAML assertions as one of the types of its security tokens. Finally, it should be noted that WS-Security has been announced as OASIS standard in April 2004.
- **WS-Trust.** WS-Trust extends WS-Security, by introducing a Security Token Service that generates, validates, and renews security tokens that are embedded in WS-Security protected SOAP messages. Additionally, a request/response protocol is introduced that enables users to obtain (or exchange) security tokens from (or with) the STS; note that depending on the types of tokens requested, the users may need to be authenticated to the STS. Additionally, in order for an STS to be able to accept security tokens generated by other STSs, it may need to ensure the authenticity of such tokens, which can be achieved using XML signatures. Clearly for an STS to be able to verify a signature made by another STS, a trust relationship needs to exist between them, which can be based on a PKI infrastructure for instance. Finally, it should be noted that WS-Trust has been announced as OASIS standard in March 2007.

- **WS-Policy.** WS-Policy is used to define a flexible grammar for expressing capabilities, requirements and general characteristics of web services using XML based policies. For example, web services may declare what kind of security tokens (e.g., only SAML assertions) they are able to process. WS-Policy became a W3C recommendation since September 2007.
- **WS-SecureConversation**. Message authentication via signatures may not be efficient when multiple messages need to be exchanged between communicating parties. For such cases, communicating parties may prefer to establish a security context that will enable them to securely exchange messages without addressing security of each message separately. WS-SecureConversation describes mechanisms for establishing such a context. The established security context can be used to determine a secret between communicating parties, which can be used for deriving session keys. These session keys may then be used for supporting symmetric key encryption in subsequent WS-Security message exchanges, and thus maintaining the authenticity and confidentiality properties of transmitted messages; hence, the established security context operates similar to SSL/TLS.
- **WS-Authorization**. This specification describes how access policies for a Web service are specified and managed. In particular it will describe how claims may be specified within security tokens and how these claims will be interpreted at the endpoint.
- **WS-Federation**. The primary goal of this specification is to enable federation of identity, attribute, authentication and authorization information among different web services (environments). WS-Federation uses WS-Security, WS-Trust, and WS-Policy components for achieving above mentioned objectives. As mentioned, WS-Federation is set as an open OASIS standard candidate, but is not declared as a standard till now.
- **WS-Privacy**. Organizations creating, managing, and using Web services will often need to state their privacy policies and require that incoming requests make claims about the senders' adherence to these policies. WS-Privacy specification can be used to achieve these objectives.

2.3.2 Introduction to WS-Federation

The main goal of WS-Federation is to enable federation of identities among different web services. The basis for WS-Federation is provided by WS-Security, WS-Trust and WS-Policy specifications. Users can achieve different identity federation scenarios by obtaining, and communicating various security tokens from one web service to another. When security tokens are exchanged between secure token services (via users or directly), it is assumed that a trust relationship exists between these services that allows them to validate signatures created by peer security token services. A collection of such security token services that have established a trust relationship is called a federation. WS-Trust offers flexible mechanisms for obtaining and communicating security tokens between various entities. For instance, certain security tokens from one web service, can be exchanged for security tokens from another web service, and used for accessing some resources at a third web service.

WS-Federation also specifies two profiles that define how interactions between users and web services can take place. These profiles are: (1) HTTP passive requestor's profile: In these profiles, web services interact with each other via users, using HTTP GET or POST redirect bindings, as defined in SAML 2.0. (2) HTTP active requester's profile: In these bindings, users themselves obtain and communicate security tokens between different web services using direct bindings, like SOAP. Finally, similar to SAML 2.0, WS-Federation supports a single logout feature that is achieved by sending a corresponding message from the requester to the web service; however details on how to achieve this will not be treated in this section. Also, similar to SAML 2.0, WS-Federation supports expression of configuration and federation metadata using the WS-Policy specification.

2.3.2.1 Main Entities in WS-Federation

Below, the main entities involved in a typical WS-Federation scenario are described.

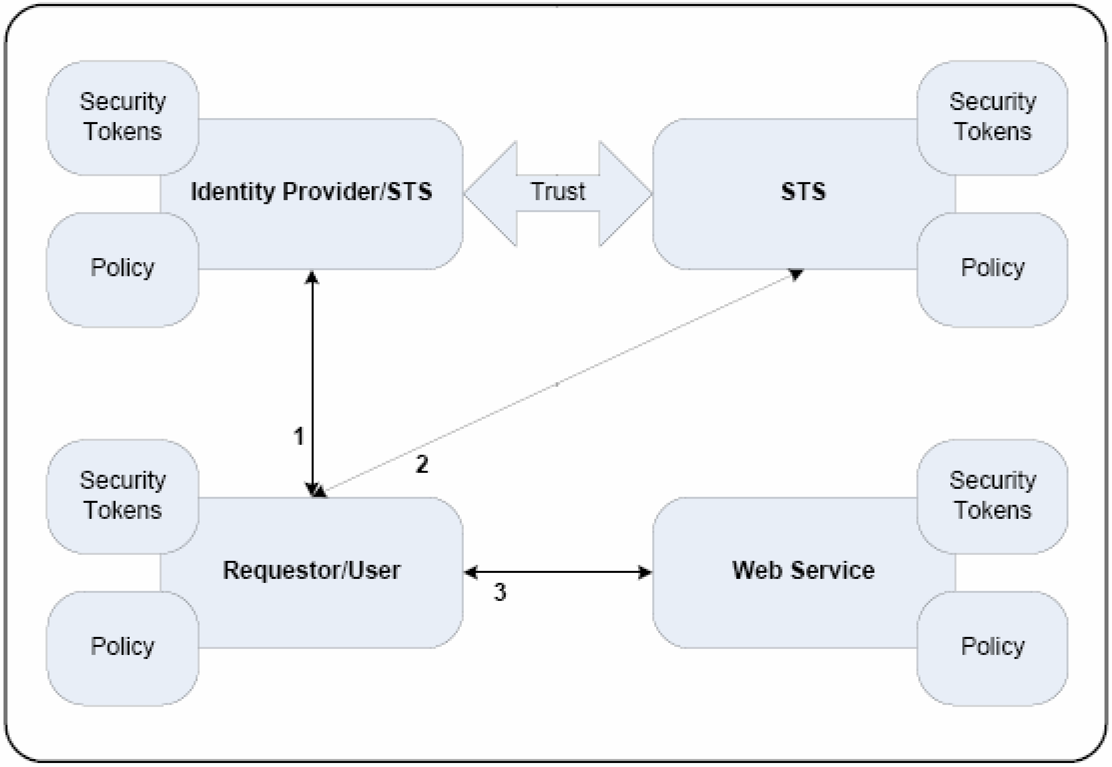
- **Identity Provider.** This entity, is typically an extension to the STS, and is used to authenticate users, and create corresponding security tokens.
- **Attribute Service.** This is a web service that maintains attribute information (similar to SAML 2.0 attributes) about users within federation; note that Attribute Service can be integrated into an STS.
- **Pseudonym Service.** This is a web service that maintains and manages alternate identity information (similar to pseudonyms in SAML 2.0) about principals within a federation; note that Pseudonym Service can also be integrated in an STS.

2.3.2.2 A Single Sign On Use Case

The scenario illustrated in [Figure 6], describes a case where a user wants to access a resource at some web service, for which the web service requires a particular access token. After obtaining such a token, the user must be able to access the resource without re-authentication.

Here are the main steps involved in below mentioned scenario. Note that user is using direct SOAP binding for communicating with the entities listed below.

1. In order to obtain the access token, the user needs to first contact the Identity Provider, in order to obtain her identity token, which she can present to the STS, who will issue her the required access token. The Identity provider authenticates the user, and generates and signs a token indicating the identity of this user.
2. The user presents this token to the STS, in order to obtain the access token required by the primary web service. The STS, after validating Identity Providers signature on the token, issues and signs the requested token to the user.
3. The user, finally, uses the obtained token in order to access the required resource and the primary web service.

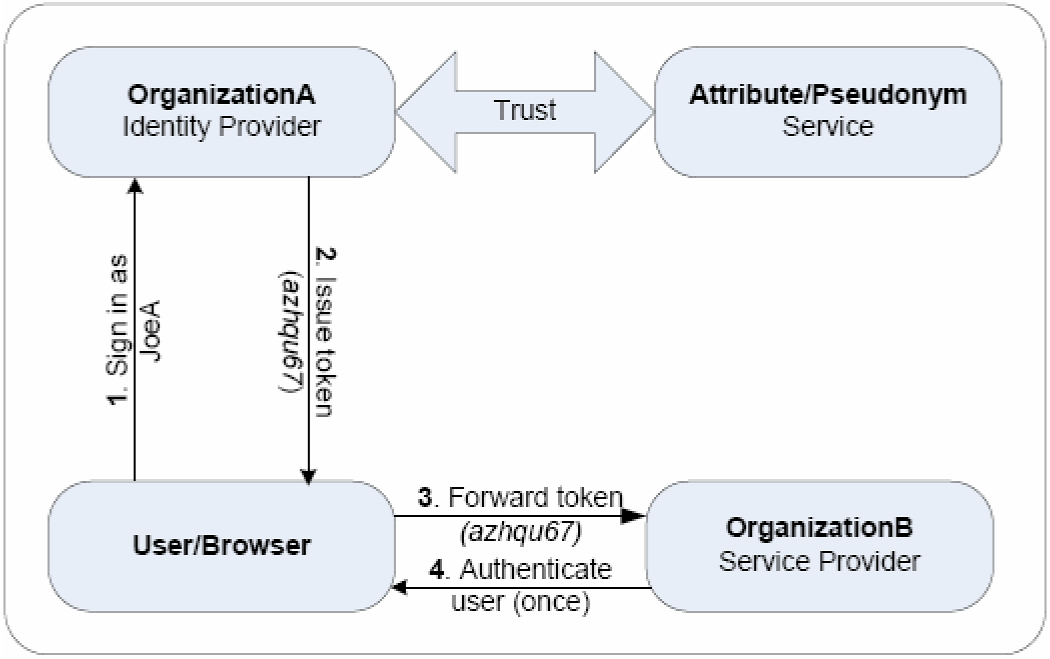


[Figure 6]. A basic interaction between a user, identity provider, STS and a web service in WS-Federation.

WS-Federation, based on WS-Trust, allows flexibility in ways that users and other entities can communicate and exchange different security and identity related information between each other. For instance, another variation of the above mentioned scenario could be that the primary web service would directly ask the user to prove her identity. In that case, step 2, would be totally omitted from the interaction, and the user would directly deliver the obtained identity token to the web service. Also, another variation could be that, similarly to the first scenario, the primary web service obtains an identity token from the user, which it would itself communicate with the STS, in order to ensure that the authenticated principal was eligible for obtaining the access token. Naturally, WS-Federation is not limited to only scenarios listed above, and other scenarios, involving a larger number of token services and identity providers, or larger number of direct and indirect interactions between different entities can be supported.

2.3.2.3 Federation Use Case with Pseudonyms

This scenario is similar to the scenario described in the Name Federation Profile of SAML 2.0. Recall that, there are two organizations, Organization A (Identity Provider), and Organization B (Web service offering some services), and user Joe, who has local accounts (JoeA, and JoeB) at both organizations. The goal of the scenario is that Joe wants to federate his local accounts at these organizations, by using persistent pseudonyms.



[Figure 7]. First time use (pseudonym) federation scenario in WS-Federation.

The steps for achieving such a scenario are illustrated below. Similarly to the previous scenario, the user uses a direct binding like SOAP for communicating with the entities involved.

1. The user, before accessing the service provider using a pseudonym, has to prove to her that she was authenticated at the identity provider. For this, the user authenticates herself at the Identity Provider (using its JoeA account), who requests a pseudonym (azhqu67) for the user from the Attribute/Pseudonym service.
2. After obtaining a pseudonym for JoeA, Organization A, based on the retrieved pseudonym azhqu67, generates a security token, indicating successful authentication of user with pseudonym azhqu67.
3. Joe forwards this token to the service provider. If federation based on such pseudonym has been established, the web service can grant Joe access to requested service. If such federation has not been established, then Organization B first needs to determine to which local account this pseudonym corresponds to.
4. For this, the Organization B, on a one time basis authenticates the user using her local account, and after successful authentication, links the supplied pseudonym with JoeB, and grants access to the requested services.

One of the main differences in name federation scenarios between SAML 2.0 and WS-Federation is that the management of user pseudonyms in WS-Federation is not performed at the service or identity providers, rather, unlike SAML 2.0, it is done by a separate entity called an attribute/pseudonym service.

Due to the flexibility of WS-Trust, several other scenarios related to name federation can be supported. In such scenarios the service providers can also be involved in interactions with the pseudonym/attribute services for requesting or registering pseudonyms or attributes of particular users. Also, if the users want to (and are allowed to) access identity or service providers anonymously, they may not be required to authenticate themselves at these organizations. Consequently, access to resources may be granted to them (solely) based on the pseudonyms or attributes they posses; for instance, an attribute may indicate that a user has some property, that entitles her to access particular resources. For other scenarios related to name federation an interested reader is referred to [12].

2.4 Comparison of Browser Based FIM Systems

First of all, it should be indicated that since SAML 1.1, SAML 1.0 and Liberty ID-FF are considered obsolete, and SAML 2.0 serves as a convergence of these standards, clearly it is more reasonable to use SAML 2.0 as an underlying FIM architecture for a given system when needed; as compared to above listed FIM standards. Further, regarding the comparison of SAML 2.0 and WS-Federation, the author would like to make the following remarks about their architectures. As it can be noticed, the architecture of SAML 2.0 is built to support specific use cases, which makes its architecture more fine grained than the one offered by WS-Federation (and WS-* Framework in general). WS-Federation, has a component based architecture (e.g., WS-Security, WS-Trust, WS-Policy), whose components can be used separately in order to achieve various web scenario requirements, but at the same time, they can interoperate between each other, and as a result achieve a wider class of supported features. Such architecture, gives WS-Federation more flexibility in the sense of supported web based usage scenarios, then SAML 2.0. However, it should be indicated, that from the implementation perspective, the flexibility (in terms of supported single sign on use cases) that WS-Federation has, can be an issue. Because, developers might prefer (from the simplicity perspective) to implement existing and well defined protocols, rather then to use the flexibility offered by WS-Federation and start to think how to adjust it to their needs and use cases. Consequently, SAML 2.0 might be easier and more straightforward to apply to particular use cases of different applications. Further, SAML 2.0 has been accepted as an OASIS standard, which is currently not the case with WS-Federation. Finally, if one needs to use WS-Federation for a basic single sign on scenario, one is required to use other technologies, like WS-Trust, WS-Security, and WS-Policy. Combining and using so many security components just for a single scenario, may be considered as not efficient approach in certain cases. So usage of WS-Federation maybe especially beneficial, when one also needs to use other components of WS-* Security Framework in order to achieve various security and identity management related requirements of different web based scenarios.

Chapter 3

Private Credential Systems and Pseudonyms for DigiD-2

This chapter describes and compares some of the existing private credential systems, like: Chen [17], Lysyanskaya [20], Camenisch [30], and Brands [21]. It also provides several pseudonym generation techniques that can be used by DigiD-2, in order to generate alternate user identifiers for her users. More concretely, in Section 3.1, we give an introduction to concepts like pseudonyms, credentials, and private credential systems. In Section 3.2, we survey above mentioned private credential systems, and compare them between each other. Section 3.3, compares private credential systems, against browser based FIM management systems and describes advantages and disadvantages of both systems. Section 3.3 is concluded with discussions on why we consider that a browser based FIM system best fits the new architecture of DigiD-2, as compared to a private credential system. Finally, since DigiD-2 is recommended to be deployed as a browser based centralized FIM system, in Section 3.4 we only suggest mathematical techniques that DigiD-2 can use in order to generate alternate user identifiers (i.e., pseudonyms); mathematical techniques for attribute generation are omitted, since in browser based FIM systems, such attributes are typically generated as string parameters.

3.1. Pseudonyms, Credentials, and Private Credential Systems

A pseudonym (Greek: ψευδώνυμον, a false name) represents a fictitious name of a person that can be used as an alternate to her real world identity (e.g., a name). In most of the cases, people use pseudonyms, when they want to remain anonymous (in the sense of not revealing their real identities) when performing certain activities. In the context of computer systems, pseudonyms (i.e., digital pseudonyms) were first introduced in [19], where they were suggested, in order to enable users to anonymously perform various online interactions with different organizations. One of the main advantages of using pseudonyms is that together with anonymity, they also provide other useful features. For instance, in the context of electronic commerce, users, on the one hand, can use (different) pseudonyms in order to anonymously set up online accounts with (different) organizations. On the other hand, organizations can use such pseudonyms, in order to authenticate their holders, and perform various authorization decisions, based on the privileges (or attributes) of online accounts to which the authenticated pseudonyms correspond to.

Additionally, organizations can issue digital credentials on such pseudonyms. Digital credentials typically contain statements about qualification or personal information about the holder of a credential. Normally, credentials are used by their holders, in order to prove certain attributes about themselves to various organizations. For instance, typical examples of non-digital credentials are: passports, cinema tickets, membership cards, driver's license, and etc. Note that in our context, digital credentials are issued on pseudonyms rather then real world identities of their holders. Clearly, one of the motivations of using such credentials is the same as using the pseudonyms themselves, i.e., to prove certain attributes about a person, without revealing her identity. Finally, a system where users: (1) can establish (anonymous) accounts with different organizations using different pseudonyms, (2) can obtain different credentials from these organizations based on their pseudonyms, and (3) can transfer obtained credentials from one organization, to another organization, is called a private credential system, and was first introduced in [19].

3.2. Private Credential Systems

3.2.1 Introduction

In the current literature, there are mainly two types of systems used for FIM [24]: (1) Browser based FIM systems: systems, where the organization issuing credentials (i.e., assertions), is in charge of transferring issued credentials to other organizations using typical web browsers and the users as intermediaries. For instance, identity management standards described in the second chapter of the thesis are normally used to implement such systems, since in those models the identity providers (IdP) are the entities that transfer issued credentials to service providers (SP) by embedding them into SAML assertions as string attributes. (2) Private credential systems: systems, where the user is in charge of obtaining, and transferring obtained credentials between different organizations. In such systems, unlike browser based FIM systems where credentials are typically represented by string attributes, credentials are typically represented using various cryptographic constructions. Note users typically need to have special software for private credential systems, in order to be able to store, transfer and perform cryptographic operations on the issued credentials.

As already noted, in most private credential systems, credentials are typically represented using various mathematical constructions. One of the reasons of such constructions is that in such systems, since users are in charge of obtaining and transferring credentials between different organizations, they are typically untrusted. Hence, credentials need to satisfy various validity properties that users have to prove (using mathematical techniques) to receiving organizations. Also, in such systems, credentials are normally desired to have various usability (e.g., one time showing vs. multiple time show of credentials) and privacy properties (e.g., selective disclosure of credential attributes), which can be elegantly achieved using such mathematical constructions.

One of the first private credential system was introduced by Chaum and Evertse [28]. However, in their system, one needs a trusted third party involved in all credential transfers, which from both efficiency and security perspective is undesirable [20]. Another pseudonym system was suggested by Damgard [29], however, it is heavily based on general zero knowledge proofs, and one way functions, and thus is mainly for theoretical interest. The first scheme that could be considered of both practical and theoretical interest was the scheme introduced by Chen [17], which is based on discrete logarithm assumptions. However, it also has some security and privacy drawbacks that will be elaborated further on. Further improvements were introduced by Lysyanskaya [20], Camenisch [30], and Brands [21], details of which will be explained in succeeding sections.

Finally, it should be noted that there are typically some common security and privacy requirements, in private credential systems, which are listed below: (1) Pseudonym Unlinkability: Organizations should not be able to link different pseudonyms of the same user with each other. Also, organizations should not be able to link a given pseudonym with the identity of its holder. (2) Credential Unforgeability: Users should not be able to show credentials to organizations that have not been properly issued to them.

3.2.2 A Private Credential System of Chen [17]

Overview. In Chen's system, there are three participants: a trusted center C, a set of organizations O_j $(j \in J)$, and a set of individuals I_k $(k \in K)$. The trusted center works as a notary center, and validates all the pseudonyms of users. Each user I_k, has different pseudonyms U_{kj} with different organization. Besides pseudonyms, each organization O_j, issues different kinds of credentials to its users. When organization O_j, issues a credential to user identified as U_{kj}, the latter may need to show the credential to another organization $O_{j'}$ where she is identified as $U_{kj'}$. In such a case, the user must be able to transfer a credential issued on U_{kj}, to pseudonym $U_{kj'}$.

Pseudonyms. Users' pseudonyms in Chen's system are generated by the user and the trusted center. A typical pseudonym is represented using mathematical construction (3.1), where, u_k is a personal identification number issued by C to I_k, g is a generator of some group with prime order, and s_{kj} is randomly and secretly chosen number from Z^*_q by I_k. After the generation of the pseudonym, a user registers it with the corresponding organization, by supplying her pseudonym and a validator of the pseudonym obtained from the trusted center.

$$U_{kj} = u_k g^{s_{kj}} \tag{3.1}$$

Credentials. After a user registers a pseudonym with an organization, and after the organization authenticates the user as the holder of the pseudonym, the organization may issue a digital credential on it. Each credential has a public key *(g,h)*, and a corresponding secret key $x=\log_g h$. The issued credential is represented as (3.2) and a signature based proof that (3.3) holds.

$$Z_{kj} = U_{kj}^{x} \tag{3.2}$$

$$\log_g h = \log_{U_{kj}} Z_{kj} \tag{3.3}$$

In order for the user to be able to show a credential Z_{kj}, issued on pseudonym U_{kj}, to an organization where the user is known under pseudonym $U_{kj'}$, she has to first modify her credential, such that it appears to be issued for the pseudonym $U_{kj'}$.

$$Z_{kj'} = U_{kj'}^{x} = Z_{kj} h^{s_{kj'} - s_{kj}} \tag{3.4}$$

and then prove somehow, that the new credential, was formed correctly. For this, the user can not use the proof from (3.3), since it is valid for credential Z_{kj}. However, the new proof, that will be valid for $Z_{kj'}$ can be achieved by diverting the proof of (3.3) to the receiving organization, with new parameters $(g,h,U_{kj'},Z_{kj'})$. Readers interested in divertibility of protocols, are referred to [17,33]. It should be noted, that transferring credentials this way, is not efficient, since every time a user desires to transfer a credential to another organization, she has to involve the issuing organization, in order to perform the divertability of the protocol, underlying equation (3.3). In order to fix this, an alternate method for transferring of a credential is suggested in Chen [17], in which the issuing organization issues a credential Z_{k^*} to a user (on some generic pseudonym $U_{k^*}=u_k g^t$), which the user shows to the receiving organization, together with the proof that the user knows (3.5).

$$\log_g\left(\frac{U_{kj'}}{U_{k^*}}\right) \tag{3.5}$$

*Discussion***.** First of all, Chen's scheme satisfies [17] both basic security and privacy requirements, stated in the beginning of this section (i.e., unlinkability and unforgeability). However, there are some security and usability related drawbacks that this scheme has.

- **Credential Sharing: Colluding Users**. It is possible that users collude, and one user $I_{k'}$ manages to use a credential of another user I_k. In order to achieve this, $I_{k'}$ has to borrow the identity number u_k, from I_k, and form his pseudonym similarly to (3.1), but using a different random number $s_{kj'}$, instead of s_{kj}. Then, she will be able to transfer the credential of user I_k, to his pseudonym, similarly, as it is done in normal situation. In order to avoid such a scenario, a process that restricts or discourages a user from sharing his identity number with other user needs to be employed.

- **Credential Sharing: Malicious Center**. Clearly, based on the previous scenario, since the trusted center is the entity that issues identifiers u_k to users (and hence knows their values), it can also perform transfer of credentials between users.

- **One Time Show Credentials**. Since, each credential has a corresponding unique public key, showing the same credential to different organizations, may be used to link users pseudonyms at corresponding organizations. In order to avoid this scenario, the user can request multiple copies of the same credential, and separately show them to the receiving organizations.

3.2.3 A Private Credential System of Lysyanskaya [20]

Lysyanskaya [20] had introduced a pseudonym system, which is an improvement to the previous systems in several aspects, and is based on the discrete logarithm and Diffie-Hellman assumptions. Specifically, the main improvement that the system of Lysyanskaya had presented was protection against credential sharing, both against colluding users, and a malicious center. The main idea behind her approach is to discourage users to share their credentials with others. That is, in her model, each user is supposed to have an external public/secret key pair, such that it is impossible for a user to share her credential with other users, without sharing her secret key. Also, among other improvements were new features like anonymity revocation, which implies that if a user performs some illegal actions, then it can be possible to extract her public/secret key pair from her pseudonym. This can be achieved due to the fact that all pseudonyms of a user are related to her public/secret key pair.

Overview. The pseudonym system of Lysyanskaya consists of the following three entities: the certificate authority (CA), organizations, and the users. Users reveal their identities to the certificate authorities, and register their public/secret key pairs with them. The role of the CA, is to guarantee to the organizations, that a user indeed possesses a public/secret key pair, and that it will be compromised if the users decide to cheat. Users compute their pseudonyms, by contacting corresponding organizations, and showing them a statement (i.e., a signature) from the CA, that they are valid system users (i.e., have a valid public/secret key pair). Note that if it is not necessary that the system protects against credential sharing, then the involvement of CA can be avoided. The organizations, similarly to Chen [17], may issue different credentials to their users using their pseudonyms. The users can then transfer obtained credentials, from one of their pseudonyms to other ones, in order to show obtained credentials to different organizations. Note that before a user obtains a credential from some organization, she has to first authenticate herself as the holder of the corresponding pseudonym.

Pseudonyms. First of all, a user's public key is represented as g^x, where $x \in Z^*_q$ is her secret key, and $g \in G$, where G is a group with prime order q. A user's pseudonym, is represented by: (a,b), where $b=a^x$, and $a=u^r$, where $u \in G$ is generated by the user, and $r \in Z^*_q$ is generated by the corresponding organization. For the full details about the actual pseudonym generation protocol, an interested reader is referred to [20]. In order for the user to authenticate herself as the holder of a pseudonym (a,b), she has to first supply the organization with a CA's signature on her public key g^x, and a proof that she knows the corresponding secret key (i.e $x=log_a b$) which was used for generating her pseudonym. In order for some party to extract the secret key of the user, she has to first obtain the users consent, and then ask her to perform the proof that $x=log_a b$ for the same commitment, but using different challenges, and then solve the system of equations: $y_1=r+c_1x$, and $y_2=r+c_2x$, in order to obtain the value of x.

Credentials. A credential of some organization, on users pseudonym *(a,b)*, is similarly to Chen [17], represented by the corresponding organizations signature on such a pseudonym, i.e., σ *(a,b)*. The signature on the credential, similarly to Chen, can be verified using the corresponding credential's public key. However, note that unlike Chen, in Lysyanskaya's system, all credentials are signed with the same secret credential key of the issuing organization. In order for a user to show a credential issued on her pseudonym *(a,b)*, to an organization, where she has a pseudonym *(a', b')*, the user has to show to this organization, the signature of the issuing organization on the pseusonym *(a,b)*, i.e., σ *(a,b)*, and prove that $log_a b=log_{a'} b'$. Due to the latter proof, a user can not share her credential with another user, without sharing her secret key.

Discussion. Lysyanskaya's system, similarly to Chen, satisfies both basic security and privacy requirements (unforgeability and unlinkability). Additionally, to the features that were already discussed, it also has a new property, as highlighted below.

- **Uniqueness Property**. It should be noted that pseudonyms used Lysyanskaya's system, unlike Chen [17], uniquely correspond to a single user, which means that the same pseudonym can not correspond to two different users. In Chen's construction, one could make the same pseudonym U_{kj} valid [22] for two different identifiers u_k and $u_{k'}$. An example how this can be achieved is demonstrated in (3.6). This scenario was not mentioned in Chen's paper, apparently due to the fact that even though technically this seems to be an issue, it is practically not a threat for the system. Since, in order for a user to use a pseudonym, she has to register it with an organization. But for this, she needs a validator from the trusted center on her pseudonym, which she would not be able to obtain on the same pseudonym twice. Also, it is very unlikely that $u_{k'}$, as constructed in (3.6), matches a different users identifier within such a system.

$$U_{kj} = u_k g^{s_{kj}} \text{, and also } U_{kj} = u_{k'} g^{s_{kj} - s_{kj'}} \text{, where } u_{k'} = u_k g^{s_{kj'}} \qquad (3.6)$$

- **One Time Show Credentials**. Similarly to Chen, Lysyanskaya's system has not achieved the feature of directly using multiple show credentials. Even though, in this system all credentials are issued using the same public/secret key pair, colluding organizations can still link different showings of the same credential. This is due to the fact that, if a user wants to show the same credential to two different organizations, she has to prove the relationship between the pseudonym on which the credential was issued *(a,b)*, and the pseudonyms she has with receiving organizations *(a',b')* and *(a",b")*. So the linking can be performed via the *(a,b)* pseudonym.

3.2.4 A Private Credential System of Camenisch, et. al [30]

Another pseudonym system was introduced by Camenisch, et al. [30], which is based on the strong RSA and Diffie Hellman assumptions. Its main innovation was to allow usage of multiple use credentials, which could not be (directly) achieved in any of the previously described schemes. The idea, that made usage of multiple credentials possible, was to use group signatures as a means for proving possession of credentials; using group signatures, each member of the group can make a signature, which can be verified using the group's public key, such that it does not reveal information about the actual signer. It should be also noted, that together with this innovation, other features that were introduced in Lysyanskaya, like protection against credential sharing, anonymity revocation, and pseudonym uniqueness, have been retained in this system. However, unlike Lysyanskaya, in this system protection against credential sharing can be achieved in two ways, in the first case, similarly to Lysyanskaya, sharing of a credential may imply sharing of some external secret key (PKI non transferability), in the second case, sharing of a single users credential may imply sharing of all her other credentials (All or nothing non transferability).

Overview. The main entities of the system are certificate authority (CA), users and organizations. The users are entities that establish pseudonyms with different organizations, and receive credentials based on established pseudonyms. The organizations are entities that issue and verify users' credentials. The organizations can verify both, a credential that they issued to a particular user using their own pseudonym, and a credential that other organizations had issued to the same user using a different pseudonym. Similarly to Lysyanskaya, if one wants to achieve PKI non transferability, then the involvement of the CA is required. In such case the CA will store users' external secret keys encrypted using the secret that is used in the formation of credentials. Then, sharing of a credential will imply sharing such a secret, which can be used for contacting the CA and decrypting the users' external secret key. For achieving *all or nothing non transferability,* organizations publish information needed for obtaining users' credentials, encrypted with the same secret as mentioned above. So, if a user shares her credential with other users, the others will be able to obtain all of her other credentials.

Pseudonyms. A user *U*'s pseudonym at some organization *O* has the form as written in (3.7), where a_O and b_O are parameters generated by the organization, x_U, is the user's secret, and $s_{U,O}$ is a random number to which both the user and organization contribute with randomness. Note that pseudonyms generated in such a way, uniquely correspond to a corresponding user.

$$P_{U,O} = a_O^{x_U} b_O^{s_{U,O}} \tag{3.7}$$

Credentials. A users credential issued on a pseudonym $P_{U,O}$ is a tuple ($e_{U,O}$, $c_{U,O}$) where $e_{U,O}$ is a sufficiently long prime and such that condition (3.8) holds. The proof of possession of a credential is realized by a proof of knowledge of a correctly formed tag $P_{U,O}$, and a credential on it. This is done by publishing statistically secure commitments to both the pseudonym, and the credential, and proving relationships between these commitments.

$$c_{U,O}{}^{e_{U,O}} = P_{U,O} d_O \tag{3.8}$$

3.2.5 Brand's Credentials [21]

Even though Brand's system was introduced earlier than the system of Camenisch, the author considered more relevant to discuss it in the end, since it is uses a completely different approach then the systems described before. The main difference between Brand's system and the systems described before is that in this system, credentials are more generic, and there is no need that the issued credentials are tied to users' pseudonyms. However, note that it is still possible that a credential contains users' pseudonyms; in such case, generation of the pseudonyms is out of scope of Brand's system. Also, one of the main innovations that Brand's system offers is that it allows its users, instead of fully disclosing a credential, to partially disclose certain attributes about their credentials. Protection against credential sharing is also possible in this system. However, similarly to the previous systems (except Camenisch [30]), Brand's scheme does not support multiple use credentials, and if one wants to use a credential multiple times, she has to obtain several copies of the same credential, in order to avoid linkability of the same users' actions between colluding organizations. Finally, Brand's had introduced another feature called discarding of credentials, which allows CA to encode certain attributes into credentials, based on which organization may refuse to accept such credentials.

Overview. In Brand's system, similarly to the previous systems, there are three participants: the certificate authority (CA), organizations and users. However, unlike other systems, in Brand's, the CA is the one who issues credentials to users. The users can then show obtained credentials (or parts of them) to any other organization.

Credentials. In Brand's system, a credential is represented by a credential public key (3.9), whose private key α is only known to the user. Values $x_1,..,x_l$ are attributes encoded inside the credential by the CA. Values $g_1,\dots,g_l,h_o$ are also generated by the CA. The CA, also separately attaches a signature to the credential *sign(h)*.

$$h = g_1^{x_1} \dots g_l^{x_l} h_o^{\alpha} \tag{3.9}$$

When a user decides to prove a possession of a credential to an organization, she has to first show to the organization the CA's signature on it, and then prove, using zero knowledge proofs, that she knows the secret key ($x_1,\dots,x_l,\alpha$) corresponding to the public key *h*. However, just proving the proof of possession of a credential is not enough for proving that the credential contains certain attributes. This can be achieved by a technique presented in Brand's [21], where a user together with proving the possession of the secret key, manages to prove that certain attributes are equal to certain values, e.g., $x_1=y_1$. Also, In order to protect the system against credential sharing, the CA typically encodes some important attributes (e.g., credit card number) in a user's credential. Then, if a user decides to share a credential with another user, this implies that she has to share all values $x_1, \dots, x_l, \alpha$ with her. This implies that the user shares her credit card number (for instance) with other users, which in typical situations is undesirable.

	Chen [17]	Lysyanskaya [20]	Camenisch [30]	Brands [21]
Pseudonym Unlinkability	+	+	+	+
Credential Forgery Protection	+	+	+	+
Credential Sharing Protection		+	+	+
Anonymity Revocation/Credential Discarding		+	+	+
Pseudonym/Credential Uniqueness		+	+	+
Multiple Use Credentials			+	
Credential Selective Disclosure				+

[Figure 8]. A summary table with various features of previously discussed private credential systems.

3.3 Private Credential vs. Browser Based FIM systems

3.3.1 Browser Based FIM Systems

Disadvantages. One of the main disadvantages of the browser based FIM systems is that the server (i.e., identity provider) is the entity who solely generates and transfers credentials between different organizations. Hence, the users have to trust, that she performs her obligations in a legitimate way. Else, the server can: (1) Impersonate the user at the service providers [27], (2) Provide the service providers with wrong details about the users [27], and (3) Monitor which service providers a particular user is conducting transactions with. Note that the latter can be an issue even if the server is trustworthy. Additionally, there are following two drawbacks: (1) since the server is always involved in any credential related transactions, she needs to be constantly available all the time (i.e., single point of failure), and (2) attackers can try to steal users' passwords that they use for authentication at the server (i.e., identity provider).

Advantages. One of the main advantages of browser based FIM systems is that the users do not need special software for performing credential related transactions, and hence are easy to use. Note that this is an important benefit, since in most of the cases users prefer to avoid installation of additional software for performing electronic commerce related transactions [18]. Also, normally, credentials generated by the servers, have a short validity period, which means that the attacker who somehow manages to steal or forge a credential, has a short time interval for reusing or sharing it [23]. Finally, since in browser based FIM systems, credentials are typically represented by string attributes, separate mathematical constructions are not necessary for generating credentials, which means that such systems are less complex to implement.

3.3.2 Private Credential Systems

Advantages. One of the main advantages of the private credential systems is that unlike the browser based FIM systems, the involvement of the issuing organization is not necessary during the credential transfers. Hence, there is no need for the main server to be constantly available, and she can not monitor which service providers the user is performing transactions with. Also, since in private credential systems, typically users' attributes are certified by a trusted third party, the identity providers can not easily impersonate users at the service providers.

Disadvantages. One of the main disadvantages of the private credential systems is that users, typically need a software client in order to perform transfer and storage of obtained credentials, and hence are relatively harder to use then server centric pseudonym systems. Also, due to the complex cryptographic infrastructure that typically underlies such systems, they are more complex for implementation than server centric pseudonym systems. Finally, protection against credential forgery and credential sharing should be a necessary condition in such systems, since in most of the cases users have a longer time interval for trying to forge credentials than in server centric systems.

3.3.3 Selecting FIM System Type for DigiD-2

As the reader could observe, the main characteristics of browser based FIM systems are that they: (1) are relatively easy to implement, (1) are relatively easy to use (e.g., they do not need additional software for transactions), (3) and it is important that the main server in such systems is trustworthy. Additionally, regardless of the fact whether the server is trustworthy or not, she can still (passively) monitor users' interactions with service providers, which can be a violation of users privacy. On the other hand, private credential systems: (1) are relatively more complex for implementation, (2) are relatively more complex for use (e.g., they need additional software for performing), but the server is not required to be trustworthy, and also the server can not monitor users' behavior.

Further, If one makes an assumption, that in the browser based FIM systems, the main server is relatively trustworthy, then the main difference between the browser based FIM and private credential systems (from the privacy and security perspective), will be that in browser based FIM systems, the main server is able to monitor users behavior. In order to avoid this issue, an additional trusted third party can be added to the system, who will act as an anonymizer proxy between the service providers and the main server (i.e., identity provider). This third party will be able to hide to which service providers a user is performing transactions with, by replacing the identity and URL of requesting service providers, with her URL and identity. More details on how this can be achieved will be elaborated in Chapter 5.

Finally, due to the fact that we consider DigiD as a relatively trustworthy entity, and due to the fact that in such a case a browser based FIM system offers relatively same security properties as a private credential system, and at the same time, it is relatively easy to use and implement, we think that a browser based FIM system best fits the new context of new DigiD. Note that the ease of use is rather an important condition in our situation, since new DigiD has to be used by millions of Dutch citizens, therefore a cumbersome to use system might not be well accepted by them. Also, since current DigiD does not use local software for transactions, it might be a good idea to maintain the architecture of the old version of DigiD, in order not to spend additional effort in educating people how to use DigiD-2.

3.4 Pseudonyms for DigiD-2

Currently, DigiD can be classified as a single sign on system, without the support for pseudonyms and credentials. But, as it is planned to suggest its extension with pseudonyms and credentials, DigiD-2 will clearly become a centralized browser based FIM system. However, since in server centric systems, credentials are represented using string attributes, only (mathematical) techniques for generation of pseudonyms will be described in this section.

3.4.1 System Overview

In DigiD-2, there are three participating parties: users, DigiD-2, and organizations. Both users and organizations have their identifiers within the system, by which they are known to DigiD. Also, each user is known by a different pseudonym to each organization. Each pseudonym that a user has with some organization is (cryptographically) based on her identifier, and the organization's identifier. This section describes the contents of users' identifiers, and introduces notations for users' pseudonyms and organizations' identifiers.

- **Users' Identifiers**. Users have their own identifiers in the system, which are represented by binary string representations of their BSN numbers, concatenated with zeros, such that the length of the extended identifiers are 128 bits; these identifiers are denoted as:

$$Y_i = (BSN_i \,||\text{"1"}||\text{"0"}^{127-length(BSN_i)}) \in \{0,1\}^{128} \tag{3.10}$$

  where*, $1 \le i \le N$,* and N is the total number of participating users*.* In order to remove possible algebraic dependencies[11] between BSN numbers, we will, in some cases (depending on the pseudonym generation technique), use: (1) either a symmetric cipher (e.g., AES) or a (2) keyed hash (e.g., HMAC-SHA1) both with 128 block sizes[12], in order to obtain two new separate sets of identifiers, comprising of either (1) S_i*=Encrypt(*Y_i*,K), 1≤i≤N*; (2) or s_i*=Hash(*Y_i*,K'), 1≤i≤N;* note that the same cryptographic key (either K or K') is used for each type of all identifiers. Finally, in order to be able to use identifiers S_i or s_i in mathematical constructions, we will use their numerical representations denoted as U_i or u_i respectively; clearly both identifiers U_i and u_i are 128 bit integers, and *1≤i≤N.*

- **Organizations' Identifiers**. Organizations also have their identifiers O_j (*1≤j≤M,* where M is the total number of participating organizations) with DigiD-2. These identifiers are also generated by DigiD, but are kept secret all the time, and are never revealed to any of the organizations. Since the format of these identifiers varies based on the pseudonym generation technique, their details will not be discussed here.

- **Users' Pseudonyms**. Further, each user with identifier x will have a different pseudonym $P(x,O_j)$ with each organization O_j. These pseudonyms are generated by DigiD-2 as well. Similarly to organizations' identifiers, the format of these pseudonyms will not be addressed here.

[11] i.e., if one knows that: BSN1 o BSN2=BSN3, then one can deduce that for instance Pseudonym(BSN1) □ Pseudonym(BSN2)=Pseudonym(BSN3), which may be used to deduce the pseudonym of user who is the owner of BSN3.

[12] In case of HMAC-SHA1, one could truncate the final hash which is a 160 bit string, to a string of 128 bits.

3.4.2 Pseudonym Requirements

- **Pseudonymity.** It should not be possible (to any other party than DigiD-2) to determine an identifier *x* of a user from her pseudonym $P(x,O_j)$.

- **Unlinkability.** It should not be possible for colluding organizations, to link different pseudonyms of the same user, i.e., to decide whether $x=y$, given $P(x,O_j)$ and $P(y,O_k)$ $j \neq k$.

- **Offline Merge.** It should be possible, when desired, and consented by DigiD-2, for organizations to merge their pseudonym databases, using a key given by DigiD-2.

- **Anonymity Revocation.** It should be possible only for DigiD-2, to efficiently determine the identifier of a user *x* from her pseudonym $P(x,O_j)$.

3.4.3 Pseudonym Generation Technique (PGT)-1

This is a basic pseudonym generation technique, which is based on symmetric encryption and does not fulfill all pseudonym requirements as listed above. However, due to its efficiency reasons, both in implementation and storage, the author considered it relevant to be discussed here.

- **System Setup**. Consider there is a symmetric encryption block cipher *E*, which can for instance be the AES (Advanced Encryption Standard) [34], with a block size 128 bits, and key size 128 bits. Each organization has an identifier O_j which serves as the key of this cipher. These keys are generated by DigiD-2, and never revealed to the organizations. Users' identifiers are represented as Y_i as described in Section 3.4.1.

- **Pseudonym Generation**. A user Y_i's pseudonym at organization O_j, is generated by DigiD-2, and represented by the formula (3.11). Literally, the pseudonyms are generated by encrypting users' identifiers using corresponding organizations keys.

$$P(Y_i,O_j)=E_{O_j}(Y_i) \qquad (3.11)$$

(i). Pseudonymity property holds, since if the key of the encryption cipher is not available, it is impossible to decrypt the encrypted cipher text, and extract the value of Y_i, from it.

(ii). Unlinkability property holds as well. Since, encryption keys are generated randomly by DigiD-2, and all of them differ from each other, hence, corresponding cipher texts appear random as well, and hence, can not be linked between each other by an entity that does not possess the encryption keys.

(iii). Offline Merge can not be achieved by this model.

(iv). Anonymity revocation can be achieved by an entity knowing the encryption key, as shown in (3.12), where *D* is the corresponding decryption function of our encryption function. Literally, an entity knowing the encryption key of a certain organization, can obtain the identity of a user, by decrypting a given pseudonym of that organization, with the corresponding encryption key.

$$D_{O_j}(E_{O_j}(Y_i)) = Y_i \quad (3.12)$$

3.4.4 PGT-2 (based on [35])

This technique uses multiplicative properties of algebraic groups in order to generate pseudonyms, and unlike the previous technique, satisfies all pseudonym requirements listed in the previous section. Also, it is assumed that in this group, Discrete Logarithm and Diffie Hellman assumptions [Appendix A.2.18] hold.

- **System Setup**. Consider we have a cyclic group G, with some small prime order q $(q>k)$, i.e., length of q is *192* bits, and a generator g, such that $G=<g>$. Users' identifiers will be denoted as u_i, as described in Section 3.4.1. Also, there is a function H that maps users' identifiers to elements of our group G. $H: \{u_1,\ldots,u_n\}\to G$, $H(u_i)=g^{u_i}$. Organizations identifiers are random numbers generated by DigiD-2, $O_j\in Z^*_q$.

- **Pseudonym Generation**. A user u_i's pseudonym at organization O_j, is generated by DigiD-2, and represented by the formula (3.13).

$$P(u_i,O_j) = H(u_i)^{O_j} \quad (3.13)$$

(i). Pseudonymity and **Unlinkability** properties hold, and can be proved as in [35].

(iii). In order for organizations O_j to perform **Offline Merge** of pseudonyms with O_k, we will transform pseudonyms of users at organizations O_k, to the pseudonyms that will be valid for users at organization O_j. First of all, we will make an assumption that all users registered at both organizations have valid DigiD-2 pseudonyms. Further, in order to transform users' pseudonyms from organization O_k, to pseudonyms of organization O_j, O_k will exponentiate (3.14) her pseudonyms using key $K_{jk}=O_jO_k^{-1}$ obtained from DigiD-2. Finally, the newly obtained pseudonyms (3.14) will be united (in the sense of set union) with O_j's pseudonym database.

$$P(u_i,O_k)^{K_{jk}} = (H(u_i)^{O_k})^{O_jO_k^{-1}} = H(u_i)^{O_j} = P(u_i,O_j) \quad (3.14)$$

(iv). In order for DigiD-2 to perform **Anonymity Revocation** of a pseudonym denoted by (3.12), she will first exponentiate it to the power O_j^{-1}, obtain $H(u_i)$, and apply the same hash function H, as described in this section, on all users' identifiers u_i's, in order to find a match with the value of $H(u_i)$.

3.4.5 PGT-3

This technique can be considered as a sub type of the previous technique. The main differences are that in this technique, instead of an abstract group G, one has a concrete additive group generated by a given elliptic curve, and correspondingly a different mapping function then considered in the previous method.

- **System Setup**. Consider there is an elliptic curve *E*, defined over field Z_p, where *p* is a *192* bit prime. Sources from where possible candidates for such elliptic curves can be chosen are mentioned in [Appendix A.4.13]. Let's denote the algebraic group generated by the points of this elliptic curve, with $E(Z_p)$; consider that the order of $E(Z_p)$ is a prime integer *r.* Also, organizations have their identifiers O_j which are represented as random integers from Z^*_r, and kept secret by DigiD-2 all the time. Users' identifiers are denoted by U_i (see Section 3.4.1), and are embedded as points P_i on our elliptic curve, using method $H:\{U_1,\ldots,U_n\}\rightarrow E(Z_p)$ as suggested by Koblitz [36] and as elaborated in the appendix of this thesis [Appendix A.4.5].

$$E : y^2 = x^3 - 3x + b \bmod p \qquad (3.15)$$

- **Pseudonym Generation.** A pseudonym of a user U_i, at organization O_j, is represented using the formula (3.16). Literally, a pseudonym is an elliptic curve point obtained by multiplying an elliptic curve point $P_i=H(U_i)$ corresponding to a user's identity U_i, by a scalar O_j that represents some organization's identifier.

$$P(U_i, O_j) = O_j P_i \qquad (3.16)$$

(i). Pseudonymity holds, since in order for an entity to obtain the value of P_i, from (3.16), she has to find a scalar O_j^{-1} such that $O_jO_j^{-1}=1 mod\ r$. However, in order to compute such a value, she first has to guess the value of O_j itself, which is infeasible, since it is equivalent to solving the Elliptic Curve Discrete Logarithm problem [Appendix A.4.8].

(ii). Unlinkability holds as well, since as organization's identifiers are randomly generated numbers, all pseudonyms are randomly generated as well. Hence, an attacker who does not possess the value of O_j^{-1}, can not distinguish generated pseudonyms from each other.

(iii). Offline Merge of pseudonyms can be achieved in the same way, as described in the pseudonym generation technique 2, using the key $K_{jk}=O_jO_k^{-1}$, where $O_k^{-1}O_k=1 mod r$, and formula (3.17) is used instead of formula (3.14).

$$K_{j,k}P(u_i, O_k) = O_j O_k^{-1} O_k^{-1} P_i = O_j P_i = P(u_i, O_j) \qquad (3.17)$$

(iv). In order for DigiD-2 to perform **Anonymity Revocation** of pseudonym represented as (3.16), she has to compute the value O_j^{-1}, such that $O_j^{-1}O_j=1 mod r$, and perform operations as shown on (3.17). Further, the value of U_i, will be extracted from the point using the method as suggested by Koblitz, and elaborated in the appendix [Appendix A.4.5]. Finally, U_i can be converted back to the corresponding BSN number of the user, by converting it to S_i, and applying a decryption function of the corresponding symmetric cipher as described in Section 3.4.1.

$$O_j^{-1} P(U_i, O_j) = P_i \qquad (3.17)$$

3.4.6 PGT-4

This system is equivalent to the PGT-2, but with a difference that it is setup using the RSA [31] setting.

- **System Setup**. Consider we have an RSA modulus $n=pq$, with length 2048 bits, and where p and q are distinct random prime numbers. Let's denote m=$\varphi(n)=(p-1)(q-1)$, where φ is the Euler's totient function [32]. Organizations' identifiers O_j, will be random numbers from Z_n, such that $1<O_j<\varphi(n)$, and $(O_j,m)=1$, $1\leq j\leq M$. Note that as in the previous cases, O_j's are never revealed to organizations. Similarly to the previous case, users' identifiers will be denoted as U_i (see Section 3.4.1).

- **Pseudonym Generation.** A pseudonym of a user U_i, at organization O_j, is represented using the formula (3.19).

$$P(U_i,O_j)=U_i^{O_j} \bmod n \tag{3.19}$$

(i). Pseudonymity holds, since in order for an entity to obtain the value of U_i, from (3.19), she has to find a number O_j^{-1} such that $O_jO_j^{-1}=1 mod\ \varphi(n)$. Finding such numbers is considered to be hard, if one does not know the value $\varphi(n)$ [32].

(ii). Unlinkability holds as well, since as organization's identifiers are randomly generated numbers, all pseudonyms are randomly generated as well. Hence, an attacker who does not possess the value of O_j^{-1}, can not distinguish generated pseudonyms from each other.

(iii). Offline Merge of pseudonyms can be achieved in the same way, as described in the pseudonym generation technique 2, using the key $K_{jk}=O_jO_k^{-1}$, where $O_k^{-1}O_k=1 mod \varphi(n)$, and formula (3.20) is used instead of formula (3.14).

$$P(U_i,O_k)^{K_{jk}} \bmod n=(U_i^{O_k})^{O_jO_k^{-1}} \bmod n=U_i^{O_j} \bmod n=P(U_i,O_j) \tag{3.20}$$

(iv). In order for DigiD-2 to perform **Anonymity Revocation** of pseudonym represented as (3.19), she has to compute the value O_j^{-1}, such that $O_j^{-1}O_j=1 mod \varphi(n)$, and perform operations as shown on (3.21). Note that since DigiD-2 knows $\varphi(n)$, she can easily compute such value using the Extended Euclidean Algorithm [32]. Finally, U_i can be converted back to the corresponding BSN number of the user, by converting it to S_i, and applying a decryption function of the corresponding symmetric cipher as described in Section 3.4.1.

$$P(U_i,O_j)^{O_j^{-1}} \bmod n=(U_i^{O_j})^{O_j^{-1}} \bmod n=U_i \bmod n \tag{3.21}$$

3.4.7 PGT-5

This technique, similarly to PGT-3, can also be considered as a sub type of the PGT-2. The main differences are that in this technique one has a group Z^*_p, with a large prime number p, i.e., 1024 bits length, and correspondingly a different mapping function then considered in the third method. Also, it is assumed that in this group, Discrete Logarithm and Diffie Hellman assumptions hold.

- **System Setup**. Again, let U_i denote users' identifiers as described in Section 3.4.1. Consider the mapping function H, which maps U_i elements to the elements of group Z^*_p, $H:\{U_1,\ldots,U_n\}\rightarrow Z^*_p$ as specified in equation (3.22). Organizations identifiers are random numbers generated by DigiD-2, $O_j \in Z^*_q$ and kept secret by DigiD-2 all the time.

$$H(U_i)=U_i^{\frac{(p-1)}{q}} \bmod p \tag{3.22}$$

- **Pseudonym Generation**. A pseudonym of a user U_i, at organization O_j, is represented using the formula (3.23).

$$P(U_i,O_j)=H(U_i)^{O_j} \bmod p \tag{3.23}$$

(i). Pseudonymity holds, since in order for an entity to obtain the value of u_i, from (3.23), she has to find a number O_j^{-1} such that $O_jO_j^{-1}=1 mod\ p\text{-}1$. Even though p is public, since the values of O_j are never revealed, one can not obtain its inverse either.

(ii). Unlinkability holds as well, since as organization's identifiers are randomly generated numbers, all pseudonyms are randomly generated as well. Hence, an attacker who does not possess the value of O_j^{-1}, can not distinguish generated pseudonyms from each other.

(iii). Offline Merge of pseudonyms can be achieved in the same way, as described in the pseudonym generation technique 2, using the key $K_{jk}=O_jO_k^{-1}$, where $O_k^{-1}O_k=1 mod\ p\text{-}1$, and formula (3.24) is used instead of formula (3.14).

$$P(U_i,O_k)^{K_{jk}} \bmod p=(H(U_i)^{O_k})^{O_jO_k^{-1}} \bmod p=H(U_i)^{O_j} \bmod p=P(U_i,O_j) \tag{3.24}$$

(iv). In order for DigiD-2 to perform **Anonymity Revocation** of pseudonym represented as (3.23), she has to compute the value O_j^{-1}, such that $O_j^{-1}O_j=1 modp\text{-}1$, and perform operations as shown on (3.25), and raise the result to the power $q(p\text{-}1)^{-1} modp\text{-}1$. Note that since DigiD-2 knows both $p\text{-}1$ and O_j, she can easily compute such value using the Extended Euclidean Algorithm [32]. Finally, U_i can be converted back to the corresponding BSN number of the user, by converting it to S_i, and applying a decryption function of the corresponding symmetric cipher as described in Section 3.4.1.

$$P(U_i,O_j)^{O_j^{-1}} \bmod p= (H(U_i)^{O_j})^{O_j^{-1}} \bmod p=H(U_i) \bmod p \tag{3.25}$$

3.4.8 Discussion

3.4.8.1 Some General Issues

In all of the PGTs, if an attacker obtains a key for merging pseudonyms of organizations A and B, and a key for merging pseudonyms of B, and C, then if one performs operations denoted in (3.26), a key for merging pseudonyms with A and C can be obtained[13].

$$K_{a,c} = K_{a,b} \circ K_{b,c} = O_a^{-1} \circ O_b \circ O_b^{-1} \circ O_c = O_a^{-1} \circ O_c \tag{3.26}$$

It should be indicated however that even in the cases where the attacker can perform such an operation, in the context of DigiD, this is not a threat. Since, organizations are allowed to merge their pseudonym databases, only when they want to merge as organizations themselves. Hence, if A and B merge their pseudonyms, this means that they have become one organization, and now, if B decides to merge with C, this means that implicitly A is merging with C as well, and hence the fact that by the previous merging, the key for merging A and C is leaked is not really an issue.

3.4.8.2 Comparison of Pseudonym Generation Techniques

PGT-1. As the reader had observed, the first pseudonym generation technique, does not fulfill the offline merge requirement. However, it is more lightweight for implementation than other techniques. So, if in the business model of DigiD-2, one considers that the merging property is not crucial, then this scheme is recommended for use as pseudonym generation technique. The security of the scheme is also reasonably adequate, especially if one uses AES (with block and key sizes 128 bits) as the symmetric encryption block cipher. Finally, note that the lack of the merging property can be justified, by letting the merging organizations merge their users' databases, as they would do in a usual way, and then reissue fresh pseudonyms for the new user database of the merged organizations.

PGT-2. PGT-2 is mainly employed for, together with other properties, to also allow for offline merge of pseudonyms. However, it requires generation of a group with prime order, where certain cryptographic assumptions hold. Also, the anonymity revocation property is not elegantly achieved, as one has to perform a brute force of all available identities of users; however, in the context of BSN numbers, this is computationally easily feasible, since one has to perform roughly comparison of 2^9 available BSN numbers.

PGT-3. PGT-3, which is a subtype of pseudonym generation technique two, suggests the generation of pseudonyms as points on an elliptic curve. The main advantage of this scheme over other schemes (as it will be elaborated further on) is that, pseudonyms have a small length, i.e., 192 bits. Also, anonymity revocation can be elegantly achieved; e.g. without performing a brute force as in the previous technique. Finally, in order to implement this scheme, an appropriate elliptic curve has to be selected first; the sources from where such elliptic curves can be chosen are mentioned in [Appendix A.4.13]. Further, in order to select an elliptic curve which generates a group where against the corresponding elliptic curve discrete logarithm problem several of the identified attacks are not feasible, one should follow the guidelines as specified in [Appendix A.4.9].

[13] Note that the notation: o, in the formula (3.24), refers to modulo multiplication with according modulus depending on the pseudonym generation technique used.

PGT-4 and PGT-5. PGT-4 can also be considered as a subtype of PGT-2, however, similarly to the previous technique it is also employed for allowing for a more elegant anonymity revocation technique than PGT-2. Further, since it is setup in the RSA setting this has following implications. If a situation in Section 3.4.8.1 takes place, then an attacker will be able to factor modulus *n*. This however means that the respective discrete logarithm problem will be reduced to the prime factors of modulus *n,* that is *p* and *q*. So to at least prevent this problem from happening, one should (apart from not providing keys that may lead to a 3.4.8.1 scenario occur) take prime numbers *p* and *q* each at least 1024 bits long. However, this has implications that one ends up with pseudonyms that are relatively expensive from storage perspective. Further, PGT-5 also allows for elegant anonymity revocation, however it is also considerably expensive from storage perspective, since the generated pseudonyms should be at least 1024 bits long; due to the size of the modulus.

3.4.8.3 Motivations for Using BSN Numbers

As it can be noticed, in each pseudonym generation technique, a user's pseudonym with each of the available organizations is based on her BSN number. Such a construction has following motivations. First of all, since we have identified the anonymity revocation property for the pseudonyms, this means that we need an efficient means using which we can obtain the identity of the user from the specified pseudonym. Using BSN numbers as underlying identifiers for our pseudonyms, this requirement can be easily achieved. Note that we could have used alternate random user identifiers instead of BSN numbers in our current pseudonym constructions, however we would then need to securely store the tables which would hold the associations of BSN numbers with such identifiers. But, since in our constructions, the pseudonyms do not let the attackers obtain any information on the underlying BSN numbers, maintaining separate random identifiers instead of BSN numbers in such a case, would be an a un necessary operation, not adding considerable value to the security of the overall system.

Further, it should be indicated that we could avoid using usage of common identifiers (e.g. BSN numbers) for users' pseudonyms with each organization at all, and randomly generate separate pseudonyms for each user and for each organization. However, clearly, then we would need to maintain tables for each user, which would store the relationships between the users' identity and her pseudonyms with each organizations. Maintaining such tables in not efficient both from the storage space and security perspectives. From the security perspectives, if such tables were implemented, then one would need to put strong efforts in order to ensure their secure storage and enforce appropriate access control rights on them. Further, if for some reasons one of the tables, or many of them would get deleted, then the link between the users' identifiers and their pseudonyms with any organizations would be lost.

Chapter 4

DigiD/A-Select and Other Single Sign On Systems

This chapter gives background information on the architectures of DigiD and A-Select. Further, it briefly describes other SSO systems like, Kerberos and Microsoft Passport, and compares them between each other. More concretely, in Section 4.1, we give a technical description of A-Select, and provide a typical A-Select SSO use case, using generic parameters. In section 4.2, we give a description of current DigiD, and similarly to A-Select, provide a typical DigiD SSO use case, however, by using concrete parameters. Next we discuss several functional and security limitations of current DigiD. Finally, in Section 4.3, we survey other SSO systems like Kerberos and Microsoft Passport, and compare them between each other.

4.1 A-Select

4.1.1 Overview of A-Select

A-Select is a web-based user authentication framework, used by various web applications, in order to authenticate their users in a single sign on (SSO) manner. The main idea behind the A-Select framework that allows the usage of single sign on features is that typically a group of various A-Select aware applications use a single A-Select server in order to authenticate their users. After authentication, such an A-Select server generates a global ticket, called the ticket granting ticket (TGT), and sends it to the user in an encrypted form – the encrypted value is also called an A-Select credential. A user can then present her TGT to various A-Select aware applications, in order to obtain different application tickets. These application tickets can then be used by such a user in order to access issuing applications, without the need to re-authenticate each time -- and thus maintain single sign on experience. When an application ticket expires or gets revoked, a user will again be forwarded to the corresponding A-Select server in order to obtain a new credential, which she can as described before use for obtaining new application tickets. Before the A-Select server issues a new credential to the user, she will ask her to re-authenticate herself. Note that if a user possesses an existing valid TGT such re-authentication will no longer be necessary[14]. Finally, below are the components of the A-Select framework that interoperate in various ways, in order to achieve above described scenarios.

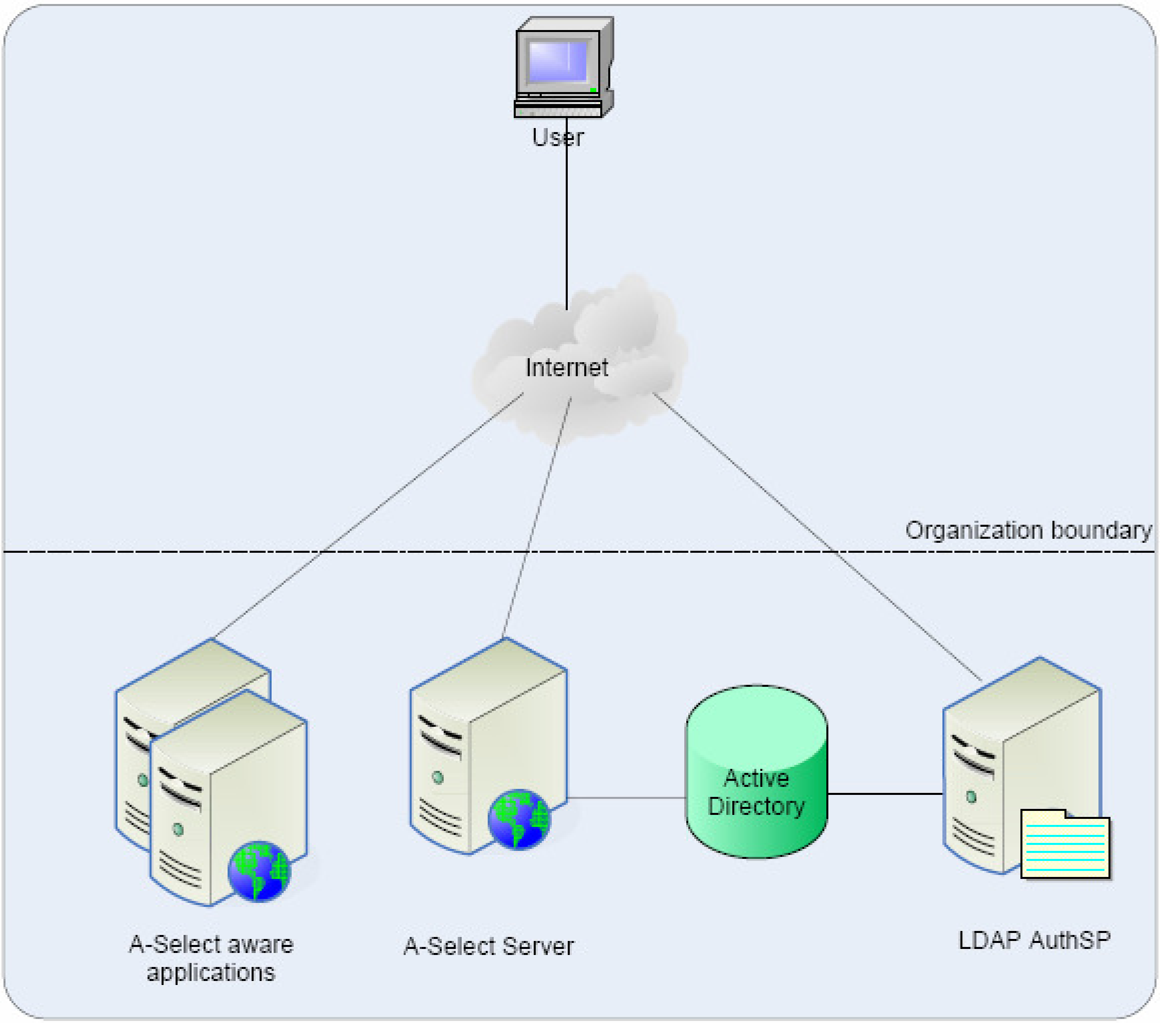
- **A-Select Aware Applications**. These are the applications who desire to authenticate their users using A-Select servers. After successful authentication, a user obtains a credential (which is an encrypted TGT) from the A-Select server, whose validity the application has to verify with the A-Select server before she generates an application ticket for her; an application ticket can then be used by her holder in order to access issuing application without the need to re-authenticate.

- **A-Select Agents.** A-Select agent is software or a daemon that runs on the same server as A-Select aware applications. It offers convenient Application Programming Interface (API) that allows A-Select aware applications to communicate with A-Select Servers. The main purpose of the A-Select agent is to implement session management and generation of application tickets.

- **A-Select Server.** A-Select server is the entity who receives user authentication requests from different A-Select aware applications. It maintains LDAP [44], as a user database, which contains users' identifiers, their attributes, and allowed authentication methods. The A-Select server itself, does not authenticate the users, rather she forwards them to the corresponding Authentication Service Providers (AuthSP) for authentication. After successful authentication, A-Select server issues a credential to the user, which the user has to present to the web application that requested her authentication. Afterwards, such application must request verification of the credential with the A-Select server, as a result of which, she is supplied (together with other parameters) with the status of the credential together with the authenticated user's identity (e.g., a name).

[14] The author would like to highlight, that this feature (as it will be elaborated later on) is not used by DigiD; hence, every time a user is forwarded to DigiD, she has to re-authenticate herself each time.

- **A-Select Authentication Service Provider (AuthSP).** The AuthSP is a server that knows how to authenticate users. There can be different types of AuthSPs, depending on the method of authentication they provide, e.g., a LDAP AuthSP, authenticates users by verifying the password that the user provides against Active Directory, an SMS AuthSP, authenticates the users using one time passwords based mechanisms implemented in their mobile phones, and etc.



[Figure 9]. A-Select deployment scenario, within a single organization, and a single LDAP AuthSP.

4.1.2 A-Select Features

- **A-Select Attributes.** A-Select offers fetching of users' attributes to requesting A-Select aware applications. A-Select attributes are a part of users' identities, and may contain any information about the user, varying from personal to qualification information. Attributes are obtained after a user gets authenticated by the A-Select server. Usually they are stored in a LDAP or a separate web service. Attributes are released to applications or other A-Select Servers according to an Attribute Release Policy (ARP). An ARP determines which attributes remote entities are allowed to see. This helps to protect the privacy of users, especially when cross authentication is used [38].

- **A-Select Tickets.** A-Select uses concept of tickets in order to keep track of authenticated users. Two types of tickets are used: (1) Ticket Granting Ticket (TGT): this is a global ticket that is issued by A-Select server after a user gets authenticated, and (2) Application ticket: this is a local ticket that A-Select aware application issues to a user after she presents a valid TGT to her. TGTs are used by A-Select aware applications, in order to generate application tickets, which users can use for accessing web services offered by these applications. Both tickets are implemented as non-persistent cookies, and hence are visible only to the servers that set them. Due to this, A-Select aware applications need to contact A-Select servers in order to ask them to verify received TGTs. Also, typically, TGTs have a longer lifecycle then application tickets, in order to maintain single sign on experience.

- **Cross Authentication**. Consider there are two different organizations, A and B, that use two separate A-Select deployments in order to manage authentication of their users, in a single sign on fashion. If a user from Organization A, decides to use a web service that is run by a server from Organization B, she will be faced with a problem, when such a web service will forward the user for authentication to the A-Select server that runs at Organization B. This is due to the fact, that the A-Select server that runs on Organization B, does not know users from Organization A. However, due to the cross authentication feature of A-Select, the A-Select server from Organization B, can forward the user to the A-Select server located at Organization A, and obtain the result of the authentication from her. Afterwards, the A-Select server located at Organization B, will notify about the result of the authentication to the corresponding web service located at Organization B, which will issue a corresponding application ticket to the user, and grant access to desired services. Note that this way, A-Select servers do not need to maintain information about the users from allied organizations, and can delegate user authentication to A-Select servers from the organizations where the users are originating from.

- **Forced Log On and SSO Groups.** A-Select also supports two additional features: (1) Forced Logon: Applications can set this option when they initiate authentication of users with the A-Select server. If this option is set, then the A-Select server will force her users to re-authenticate themselves, regardless of the fact whether they possess a valid TGT or not, and (2) SSO Groups: Using this feature, an A-Select server can group various A-Select aware applications in different SSO groups. Then, if a user obtains a valid TGT from the A-Select server as a result of authentication request from an application that belongs to a certain group, and afterwards she desires to access an application that is a member of another group, the A-Select server will ask her re-authenticate herself once again; clearly this will not be the case, if the application the user decides to access afterwards belongs to the same group as the initial application. For more clarity, consider one has three applications: app1, app2, and app3, and there is a group containing: app1 and app2. Then, if a user gets authenticated by the A-Select server as a result of request from app1, and afterwards she desires to access app3, the A-Select server will ask her to re-authenticate, even if she possesses a valid TGT. This will however not be the case, if the user decides to access app2.

4.1.3 A General A-Select Single Sign On Use Case

Below is a typical A-Select single sign on use case [Figure 9], where a user, not having an application ticket, wants to access some web service offered by a A-Select aware application. For this, the A-Select aware application initiates the user authentication procedure with the A-Select server. The A-Select server authenticates the user and generates a TGT for her, which she can use for obtaining an application ticket. Finally, as long as the application ticket remains valid, the user can use it in order to access the corresponding web service, without the need to re-authenticate.

- **Step 1**. The user, without having a valid application ticket, desires to access some web services at an A-Select aware application.

- **Step 2**. The A-Select aware application initiates user authentication. For this, she contacts the corresponding A-Select server, and invokes the "initiate user authentication" API call with the following parameters: (1) her own identifier and URL (app_id, app_url), (2) optional: the identifier of the user who must be authenticated (uid), (3) optional: the identifier of the organization from where the user originates (r_org) – this parameter can be used for cross authentication, in order to indicate the initial organization where use originates from, and (4) optional: the forced logon parameter (f_logon).

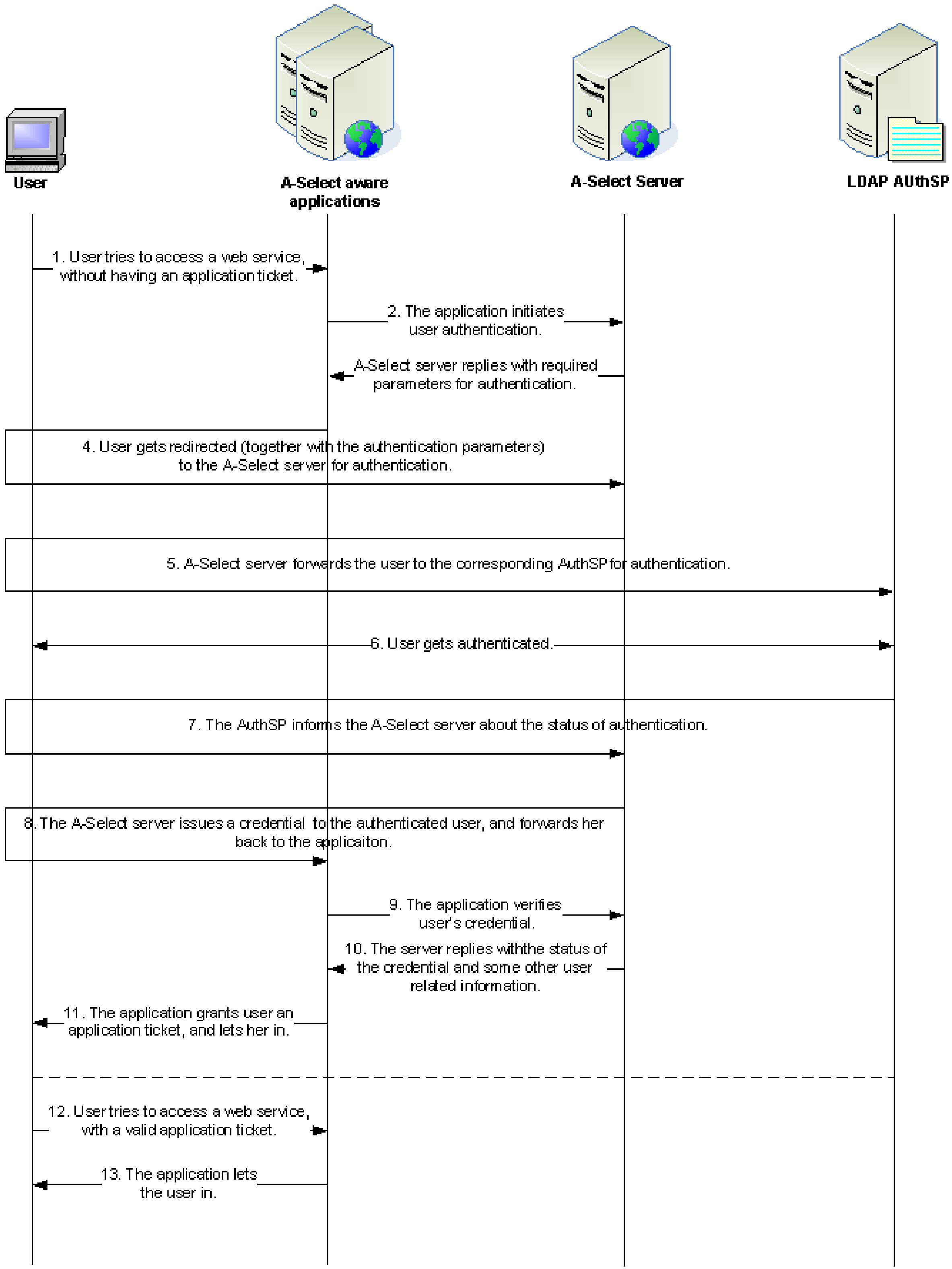
- **Step 3**. If the A-Select server accepts user authentication request, she will generate and send parameters required for user authentication to the requesting applications. Generated parameters are: (1) the identifier and the URL of the A-Select server (a-select-server, as_url), (2) and a unique number called request identifier (rid): this parameter is used for keeping track of the authentication sessions.

- **Step 4.** After the A-Select aware application obtains parameters sent above, she will redirect the user to the A-Select server, using the URL it had received in the previous step (as_url). Additionally, the rid parameter received in the previous step is attached to the redirection URL.

- **Step 5.** As soon as the user gets redirected to the A-Select server, she is asked to enter her identity in a form (if it was not received as part of step 2). Afterwards, the server looks up her name in the database, obtains a corresponding authentication method for her and forwards her to the respective authentication service provider for authentication.

- **Step 6-7.** The authentication service provider engages into an authentication session with the user, and forwards her back to the A-Select server after successful authentication.

- **Step 8**. The A-Select server, after evaluating the response from the AuthSP, generates a credential for the user, which contains her Ticket Granting Ticket, encrypted under the secret key of the issuing A-Select server. The user is then forwarded back to the application she originated from, together with the credential, attached to the redirection URL.



1. User tries to access a web service, without having an application ticket.

2. The application initiates user authentication.

A-Select server replies with required parameters for authentication.

4. User gets redirected (together with the authentication parameters) to the A-Select server for authentication.

5. A-Select server forwards the user to the corresponding AuthSP for authentication.

6. User gets authenticated.

7. The AuthSP informs the A-Select server about the status of authentication.

8. The A-Select server issues a credential to the authenticated user, and forwards her back to the applicaiton.

9. The application verifies user's credential.

10. The server replies withthe status of the credential and some other user related information.

11. The application grants user an application ticket, and lets her in.

12. User tries to access a web service, with a valid application ticket.

13. The application lets the user in.

[Figure 10]. A Typical A-Select SSO use case, with successful authentication.

- **Step 9**. Now, the A-Select aware application has to verify whether the user's credentials that were presented to her are valid. For this, she contacts the corresponding A-Select server, by initiating a "verify credentials" API call with the following parameters: (1) rid, and the (2) received credentials (aselect_credentials).

- **Step 10**. The A-Select server decrypts received user's credential, extracts TGT from it, and verifies whether it is valid or not. In case the user's TGT is valid, the A-Select server responds to the A-Select aware application with information that indicates that the user was successfully authenticated. Such information consists of parameters like: (1) rid, (2) identifier of the authenticated user (uid), (3) authentication method used (auth_level), (4) expiry time of the user's TGT (tgt_expire_time), (5) optional: user attributes (attributes), and (6) optional: the identifier of the organization that use originates from (r_org).

- **Step 11**. The A-Select aware application, based on the response from the A-Select server, generates and issues an application ticket to the user. As a result, she also grants access to her to the desired web services.

- **Step 12**. In this step, user decides to access the application at a later time, after the application ticket was issued to her. For accessing the web service, she simply presents her application ticket to the corresponding web application.

- **Step 13**. The A-Select aware application validates user's application ticket, and if it is valid, grants access to her for the desired services.

4.2 DigiD

4.2.1 Overview of DigiD

DigiD [42] is a single sign on system, based on A-Select, which is used by Dutch governmental organizations, for authenticating users who want to use their services online (e.g., electronically applying for a grant, accessing personal electronic records, paying taxes, etc.). DigiD uses BSN numbers of its users, as their name identifiers. That is, any person having a valid BSN number can apply for a unique account at DigiD, and obtain login information, which she can use for authenticating with DigiD; note that DigiD supports different levels of authentication as it will be elaborated further on. After authenticating with DigiD, a user can access specified governmental organization, without the need to re-authenticate once again. Note that when a user gets authenticated at DigiD, DigiD informs corresponding governmental organization about the BSN number of the authenticated user, together with the authentication method used. These BSN numbers are then used by governmental organizations as a key in the citizen's database (GBA), in order to know who was authenticated and engage in an online transaction with her, if necessary.

The authentication methods used by DigiD can be divided into following three levels. (1) Basic: this level provides the organizations with a limited degree of certainty about the authenticated users, and can be used for services, where the accessed information primarily is of importance for the specific applicant. Typically, this authentication level is achieved using a username/password based authentication. (2) Medium: this level provides the organizations with a medium degree of certainty about the authenticated users, and can be used by services where the user accesses a small amount of special and/or financial and economic personal data, and/or a small amount of confidential business or Government data. Typically, this authentication level is achieved using a one time password based scheme, implemented in mobile phones of the users. (3) High: this level provides the organizations with a high degree of certainty about the authenticated users, and can be used for services where a user wants to access a significant amount of special and/or financial economical data, and/or secret corporate or government data. Currently, this authentication level is not used by DigiD.

In the context of A-Select, DigiD is represented as an A-Select server, and governmental organizations are represented as A-Select aware applications. Hence, all authentication flows are similar to standard A-Select use cases, however with DigiD specific parameters. For instance, as mentioned above, DigiD uses BSN numbers as user identifiers that are used in the A-Select framework. Also, as it has been highlighted above, DigiD supports two types of authentication service providers (AuthSPs), with simple username/password based authentication, and one time password based authentication scheme implemented using mobile phones. Finally, governmental organizations can use either SOAP or CGI for communicating with DigiD.

4.2.2 A Concrete DigiD SSO Use Case, based on CGI Parameters

This section describes a practical single sign on use case based on DigiD specific parameters extracted from [43]. Since DigiD is based on A-Select, the use case [Figure 11] described here is the same as the use case described in the previous section, with some differences with respect to parameters used. Also, for more clarity, this use case is represented using practical values. Consider a service run by some governmental organization that a user wants to use. The URL and identifier of this organization are: https://diensten.hengelo.nl/secureportal and hengelo_digid_portal respectively. This organization uses DigiD in order to authenticate her users. The URL and the identifier of DigiD are https://as.digid.nl/was/server and DigiD1 respectively. Also, there is a shared secret (in our example) which DigiD and this organization use, for communicating with each other. The value of this secret is 123456-kd2s-s3kg-72kf-k2f3-mk2e-aoe3. Finally, the governmental organization's application uses CGI for communicating with DigiD.

- **Step 1**. The user, without having a valid application ticket, desires to access a governmental application called hengelo_digid_portal, in order to use certain electronic services.

- **Step 2**. The organization initiates user authentication. For this, she contacts DigiD, and calls the initiate authentication API call, using the following parameters: (1) identifier and URL of the application (app_id, app_url), (2) shared secret (shared_secret), and the (3) identifier of DigiD (aselect-server). The request is sent as a CGI request, with following URL.

  https://as.digid.nl/was/server?request=authenticate&app_url=https%3A%2F%2Fdiensten%2Ehengelo%2Enl%2Fsecureportal &app_id=hengelo_digid_portal &shared_secret=123456-kd2s-s3kg-72kf-k2f3-mk2e-aoe3&aselect-server=DigiD1

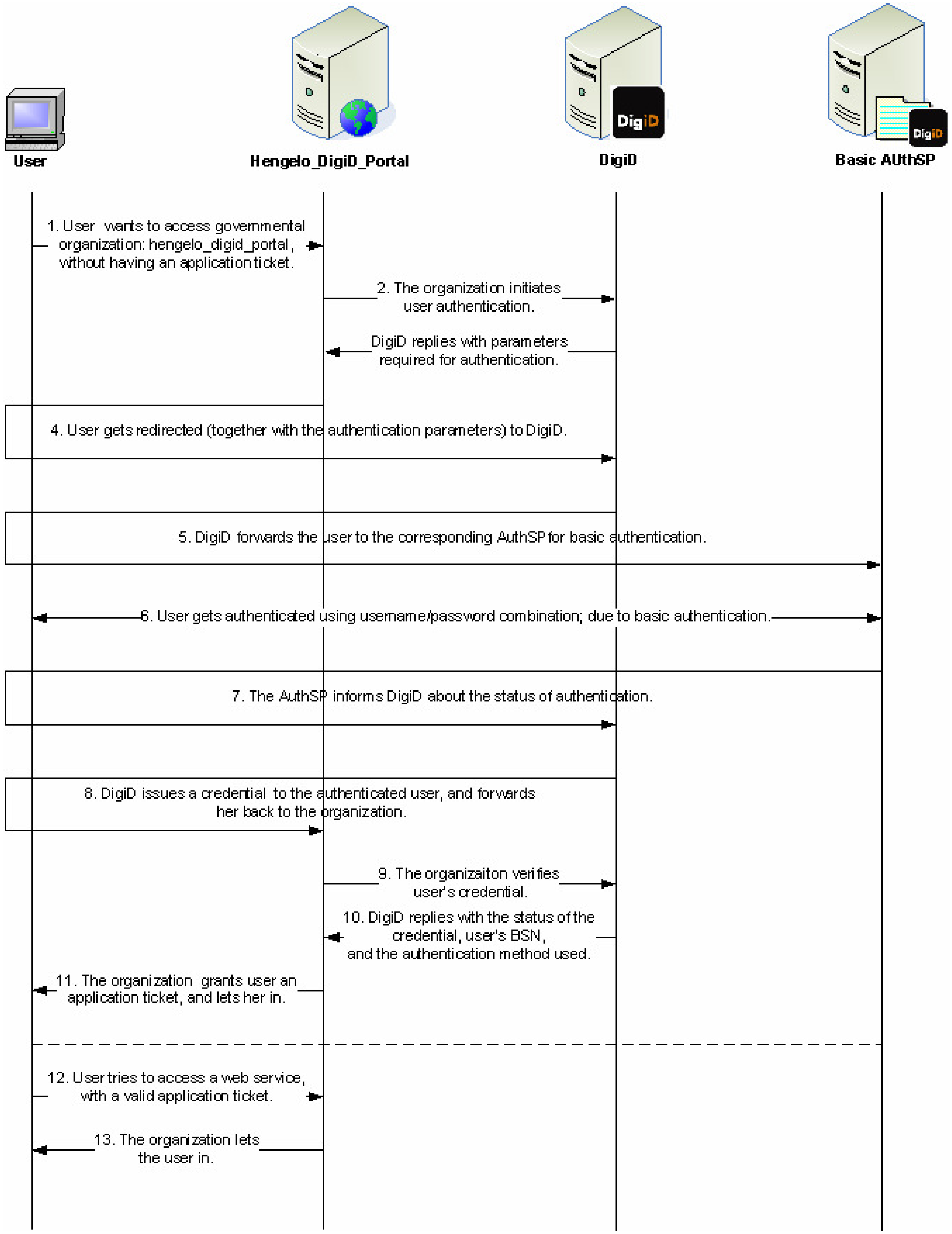
- **Step 3**. If DigiD accepts user authentication request, she will generate parameters required for authenticating the user. Generated parameters are: (1) DigiD's identifier and URL (a-select-server, as_url), (2) and a unique number called request identifier (rid). Values of these parameters can be viewed below.

  rid=A77C582B33C03912&as_url=https://as.digid.nl/aselectserver/server?request=login1&a-select-server=DigiD1&result_code=0000

- **Step 4.** After the organization receives these parameters, she will redirect the user to the DigiD, using the URL it had received in the previous step (as_url). Additionally, all other parameters received in the previous step will be attached to the URL:

  https://as.digid.nl/aselectserver/server?request=login1&rid=A77C582B33C03912&a-select-server=DigiD1

- **Step 5.** As soon as the user gets redirected to DigiD, she is asked to enter her identity in a form. Afterwards, DigiD chooses a corresponding authentication method for her, and forwards her to such AuthSP. Note that it is not necessary that an AuthSP resides on a separate server, as DigiD.


User
Hengelo_DigiD_Portal
DigiD
Basic AUthSP
1. User wants to access governmental organization: hengelo_digid_portal, without having an application ticket.
2. The organization initiates user authentication.
DigiD replies with parameters required for authentication.
4. User gets redirected (together with the authentication parameters) to DigiD.
5. DigiD forwards the user to the corresponding AuthSP for basic authentication.
6. User gets authenticated using username/password combination; due to basic authentication.
7. The AuthSP informs DigiD about the status of authentication.
8. DigiD issues a credential to the authenticated user, and forwards her back to the organization.
9. The organizaiton verifies user's credential.
10. DigiD replies with the status of the credential, user's BSN, and the authentication method used.
11. The organization grants user an application ticket, and lets her in.
12. User tries to access a web service, with a valid application ticket.
13. The organization lets the user in.

[Figure 11]. A typical DigiD SSO use case, with successfull basic authentication level.

- **Step 6-7.** The authentication service provider engages into an authentication session with the user, and forwards her back to the DigiD after successful authentication.

- **Step 8**. DigiD, after evaluating the response from the AuthSP, generates a credential (aselect_credentials) for a user, which contains an encrypted[15] Ticket Granting Ticket. The user is then forwarded back to the application she originated from, together with the generated credential, attached to the redirection URL. Note that the rid and the identifier of DigiD (a-select-server) are also attached to this URL together with the credential.

  https://diensten.hengelo.nl/secureportal?aselect_credentials=X&rid=A77C582B33C03912&a-select-sever=DigiD1

- **Step 9**. After the governmental organization receives user's credential, she wants to verify whether it is valid or not. For this, she has to contact DigiD, by initiating a verify credentials API call with the following parameters: (1) rid, and the (2) received credentials (aselect_credentials=X). The URL of the request has the following value.

  https://as.digid.nl/was/server?request=verify_credentials&aselect_credentials=X&rid=A77C582B33C03912&shared_secret=123456-kd2s-s3kg-72kfk2f3-mk2e-aoe3&a-select-server=DigiD1

- **Step 10**. DigiD, decrypts received user's credential, extracts TGT from it, and verifies whether it is valid or not. In case the user's credential is valid, DigiD responds to the governmental organization with the information that indicates successful authentication of the user. Such information consists of parameters like: (1) rid, (2) identifier of the user, which is represented by her BSN number (uid=BSN), (3) authentication method used (auth_level), (4) the identifier of DigiD (a-select-server), (5) identifier of the application (app_id), and (6) identifier of the organization where the user originates from (r_org).

  rid=A77C582B33C03912&uid=190382582&app_id=hengelo_digid_portal&auth_level=10&r_org=DigiD&aselectserver=DigiD1&result_code=0000

- **Step 11**. The organization, based on the response from DigID, generates and issues an application ticket to the user. As a result, she also grants her access to the desired web services.

- **Step 12**. In this step, user decides to access the application at a later time, after the application ticket was issued to her. For accessing the web service, she simply presents her application ticket to the corresponding web application.

- **Step 13**. The organization validates user's application ticket, and if it is valid, grants access to her for the desired services.

[15] Clearly the TGT is encrypted under the key which is only known to DigiD.

4.2.3 Discussion about Functionality of DigiD

First of all, the author would like to make one note about the single sign on features of DigiD. After practically testing DigiD, it appears that the only thing that makes DigiD single sign on, is a single user name password combination. This is due to the fact that DigiD (or the applications using DigiD) apparently uses “forced logon” feature of A-Select, which means that each time a user is forwarded to DigiD, she is asked to re-authenticate herself[16] regardless of the fact whether she was authenticated with DigiD some time ago or not. From the usability perspective, it would be nice to have this feature removed, and allow the users to fully benefit from single sign on features available within A-Select, i.e., let users use their TGT’s as long as they are valid. From the security perspective, this method does not bring any important benefits either. Since, if a user’s password has been stolen, this method does not protect her from someone accessing her account, rather it ensures that if someone accesses a computer while she is logged in at DigiD, then the attacker will not be able to access other governmental applications using her active session with DigiD. Such a scenario, is first of all uncommon, and secondly it is unlikely that an attacker manages to perform such an attack on a planned basis; this attack more has a nature to occur by chance, which both reduces the likelihood of it happening, and the impact it can have, even if it occurs. Moreover, DigiD could further decrease both the likelihood and the damage that this attack could cause, by automatically logging out the users, who have been logged into DigiD for more then a certain number of minutes, for instance. Finally, it needs to be indicated that due to the lack of this feature, current DigiD does not also allow for a single sign off feature, which would allow users, to after logging out of DigiD, to automatically log out from all SP where they had initiated active sessions.

Secondly, even though A-Select supports features of fetching users attribute data to requesting organizations, DigiD has currently not implemented this feature. It should be highlighted that attributes can be rather beneficial feature in certain situations, since organizations can request different personal information about their users from DigiD and obtain them without asking the user to type them in; this way, organizations would also be sure that the information provided by DigiD is adequate and valid. However, note that in case the attribute release is implemented, it is very important that their release is controlled and enforced by the users, in order to avoid the violation of their privacy. A-Select offers a means that allows its users to define attribute release policies, based on which, the A-Select servers decides which attributes it may release to which applications.

4.2.4 Security Considerations of DigiD

4.2.4.1 Usage of SSL/TLS

Current version of DigiD/A-Select, uses the SSL/TLS suite, in order to achieve security properties, like confidentiality, authenticity (of business partners), and integrity properties of data transported on the application level. More precisely, SSL/TLS is being used in the following way. Any party, except the users themselves, has to present a certificate to the interacting party, in order for the latter to authenticate the former, as a legitimate party bearing a given public key. Further, when these applications are interacting with each other, they have to use SSL with bilateral authentication, that is, they have to be mutually authenticated.

[16] This can be also be seen from the fact, that DigiD does not even create, neither persistent, nor non-persistent cookies, once a user gets authenticated with her – i.e., she does not track the authentication state of her users.

4.2.4.2 Possible Attacks on DigiD and their Countermeasures

- **A SP Impersonates a User With Another SP.** A malicious but legitimate SP Organization A, can impersonate a user Joe, at some other SP Organization B, if certain assumptions[17] hold. For instance, when Joe desires to access Organization A, the latter will have to initiate authentication of Joe with DigiD. For this, Organization A, typically has to send the following two parameters (together with other parameters) to DigiD: her app_id, and app_url (step 2, Section 4.2.1). Now, if we assume that DigiD only uses the app_url, as an address where she has to send the user back after authentication, Organization A can leave such address unchanged, and modify the app_id, to the ID of the SP (i.e., Organization B) with whom she wants to impersonate user Joe. Next, after the user gets authenticated, DigiD will forward the user with necessary parameters back to the Organization A (due to the received app_url parameter and our assumption). Organization A, will however, reject access to the user, and send the received credentials and a RID to Organization B. Organization B, as a result of verifying user's credentials, will be notified by DigiD (i.e., with app_id parameter, step 10, Section 4.2.1), that the credential was indeed issued for her organization (since Organization A had modified app_id parameter as described above), and grant access to the malicious SP. *Countermeasure*: Organization B, can verify whether the received RID was generated as part of her initiation of user authentication, in order to render this attack unsuccessful. Note that this attack will also not work, if DigiD will initially verify whether the received app_id corresponds with the received app_url.

- **Faking Web-Site of DigiD**. DigiD should use strong efforts, in order to protect its web site from phishers, i.e., persons who could make a fake copy of the web-site, and ask users (based on various motivations) to type their login information inside; clearly, in the mean time the attacker would be able to copy login information of thousands of users of DigiD. Even though the attacker might not cause much harm with it – since password/username based authentication is used only for unimportant transactions – this is something DigiD should try to avoid; both from the users' security perspective and with respect to the image that DigiD has among its users. *Countermeasure:* As a suggestion, in order for DigiD to be able to protect against scenarios mentioned above, she (together with maximally educating her users) should try to use sign on seals[18] [45], as used by services like Yahoo email for instance; this way users would be able to detect whether they are on a legitimate DigiD web site or not.

- **Different Applications within the Same Organization**. If an organization runs different applications, on the same server, then one malicious application can forward the received credentials to another application within the same organization, and illegitimately gain access to it. *Countermeasure:* As it has been highlighted in [47], it is important, that the organization which desires to initiate user authentication, informs (using app_id parameter) DigiD, about the identity of the particular requester resource, or application, instead of the domain URL.

[17] To the author's best knowledge, the documentation of DigiD does not make clear whether these assumptions hold or not; assumptions that are referred to here will be highlighted throughout the specified paragraph.

[18] Sign on Seal is bitmaps, which each user of a web site can have individually chosen, and which she can use in order to check whether the web-site she has visited is legitimate. i.e., by verifying the displayed bitmap against her chosen one.

4.3 Other Single Sign on Systems

4.3.1 Kerberos

Kerberos [55] is one of the first single sign on systems, developed at the Massachusetts Institute of Technology. It uses symmetric key cryptography for achieving required security objectives, due to what users are entitled to use local software called a Kerberos client during Kerberos transactions, in order to be able to perform different symmetric key cryptography related operations. Further, several versions of Kerberos exist, but Kerberos v5 is the newest and is considered as the most secure version as compared to other versions of Kerberos; thus, in this section we will concentrate on Kerberos v5. In Kerberos, similarly to any single sign on system, a user typically wants to authenticate once with an authentication server (e.g., identity provider IdP), and during the validity of the authentication session, be able to use services offered by different applications (e.g., service providers SPs) without the need to re-authenticate (single sign on). Finally, the main entities involved in a typical Kerberos transaction are the following. (1) Local Kerberos Client Software: A component which is installed on a user's machine, and is used to perform initial user authentication (e.g., with username/password), (2) The Authentication Server (AS): an entity that simply grants a Ticket Granting Ticket (TGT) to the authenticated user, (2) The Ticket Granting Server (TGS): an entity which receives a TGT from a user, and in case it is valid, issues a Service Granting Ticket (SGT) to her, and (3) Service Provider (SP): an entity that receives a SGT from a user, and in case it is valid, grants her access to the specified service. The SSO functionality is achieved by the fact that unless user possesses a valid TGT, the local Kerberos software does not request her to re-authenticate. A brief description of a typical Kerberos single sign on use case is given below; note that some of the exchanged parameters are omitted for better readability.

- **The User Logs into the Local Kerberos Client Software**. A user enters her username and password on the client machine. The Kerberos client performs a hash on the entered password, and this becomes a secret key of the user, which she will use with the further interactions with the AS.

- **The AS Issues a TGT to the User**. The Kerberos client informs the AS that the user would like to access a certain SP. For this, the AS verifies if the client is in the database, and sends the following messages to her: (1) Message A: Client/TGS session key encrypted using the secret key of the user. (2) Message B: TGT, which contains, together with other parameters, the Client/TGS session key. The TGT is encrypted using the secret key of the TGS. Once the user receives such messages, she will first decrypt Message A, and obtain the Client/TGS session key, which she will use for further interactions with the TGS.

- **The User Presents Received TGT to the TGS.** At this stage, the user will go to the TGS, present the received TGT from the previous step, and obtain an SGT which she will present to the service providers later on. More concretely, the user sends following parameters to the TGS: (1) Message C: which contains Message B, and an ID of the requested service provider, and (2) Message D: an authenticator (typically this parameter contains the client ID and a timestamp) encrypted with the Client/TGS session key.

- **The TGS Issues a SGT to the User.** The TGS obtains the Client/TGS session key from Message B, and uses the newly obtained Client/TGS session key, in order to decrypt the authenticator from message D. Afterwards, she replies with the following parameters to the client: (1) Message E: SGT, which together with other parameters, contains the Client/Server session key. The SGT is encrypted using the secret key of the corresponding service provider. (2) Message F: Client/Server session key, encrypted with the Client/TGS session key.

- **The User Presents the SGT to the SP**. The client connects to the SP, and sends her the following messages. (1) Message G: which contains Message E, from the previous step, and (2) Message H: which contains a new authenticator, encrypted using the Client/Server session key. The SP, obtains the SGT from Message G, decrypts is using her secret key, and obtains a Client/Server session key. Further, she uses the Client/Server session key, in order to decrypt the received authenticator, and informs the user that she is willing to grant her access to the desired services.

4.3.2 Microsoft Passport

Microsoft Passport [62] is a web based single sign on system offered by Microsoft since 1999. The Microsoft Passport server acts as an entity that authenticates the users (e.g., identity provider), and informs about the result of the authentication to the corresponding service providers. Similarly to A-Select, Microsoft Passport is also somewhat similar to Kerberos, and moreover, similarly to A-Select, it also uses SSL/TLS suites, in order to achieve required security objectives. A typical Microsoft SSO use case, where a user wants to be able to access different applications, by only authenticating with the Passport Server, is described in the following way.

- **User Tries to Access a SP.** A user who does not have an existing authentication context with a given service provider, tries to access some services offered by her. As a result, the SP redirects the user to the Passport server for authentication.

- **The Passport Server Authenticates the User.** The Passport server verifies if the user already has a Ticket Granting Cookie (TGC). If the cookie is found, then this step is skipped, else, the Passport server decides to authenticate the user. If the user gets authenticated, then the Passport server saves a TGC encrypted under the secret key of the Passport server, in the user's browser cache.

- **Passport Server Issues a Service Ticket to the User.** The passport server saves a set of cookies in the browser's cookie cache. Such a cookie set is encrypted using the shared key of the Microsoft Passport Server and the corresponding service provider. These cookies are similar to service tickets from Kerberos.

- **User Accesses the SP.** Finally, the user is redirected back to the application which she originated from. As a result, the application decrypts the corresponding cookie, from the previous step, and grants access to the user, to her desired services.

4.3.3 Differences between Kerberos, Microsoft Passport and A-Select

In this section we will summarize the major differences between Kerberos, Microsoft Passport and A-Select. First of all, as it was highlighted in the previous section, one of the most important differences between these systems is that Kerberos, unlike A-Select and Microsoft Passport, uses symmetric key cryptography in order to achieve required security objectives; A-Select and Microsoft Passport use SSL/TLS. As a result, a user who uses Kerberos needs to use Kerberos client software which takes care of conducting exchange of cryptographically secure messages between various Kerberos entities. On the other hand, with A-Select and Microsoft Passport, a simple browser is enough for performing necessary internet transactions. Secondly, as the reader could observe, in A-Select, unlike Kerberos, or the Microsoft Passport, the SP has to contact the A-Select server, in order to verify the validity of the received TGT; which is clearly not the case with Kerberos and A-Select. Thirdly, in A-Select, the authentication service provider does not generate the TGT, rather she just authenticates the user, and informs about the result of the authentication to the A-Select server. The A-Select server then generates a TGT for the user. In Kerberos and Microsoft Passport, a TGT is generated by the server which performs user authentication.

Chapter 5

Extending DigiD to the private sector (DigiD-2)

This chapter describes DigiD-2, and provides its specification in the context of SAML 2.0. More concretely, in the beginning of Section 5.1, advantages of extending DigiD to the private sector are discussed. Next, an overview of DigiD-2 is given, and ways to allow organizations to link supplied pseudonyms with corresponding local accounts are provided. Further, a typical DigiD-2 SSO use case is described. In Section 5.2, the description of functional and non-functional requirements that DigiD-2 should satisfy are described and in Section 5.3, SAML 2.0 is suggested to be used as a new kernel for DigiD-2 (instead of A-Select), and several motivations for such a suggestion are given. Also, a specification of the DigiD-2 SSO use case described in section 5.1.5 is given using SAML 2.0 Web SSO Profile with persistent pseudonyms and Artifact Binding. Finally, ways to enforce the control of release of users attributes are discussed, and a new mechanism that allows the users to hide the identities of the organizations they are accessing from DigiD-2 is suggested, followed by discussions about the security considerations of DigiD-2 in Section 5.4.

5.1 DigiD-2

5.1.1 Benefits of Extending DigiD to the Private Sector

Currently, the deployment of DigiD by the Dutch government can be considered as a successful initiative, due to the large number of both, governmental organizations, and Dutch citizens, that have started to use it as an authentication framework. Furthermore, the number of such users is growing as time goes by. This is due to the fact that both, governmental organizations, and Dutch citizens, have realized the benefits of using a single sign on authentication framework for performing various internet based transactions with each other. However, as it has been highlighted, the scope of organizations that can currently use DigiD, is only limited to Dutch governmental organizations. Therefore, it becomes reasonable to think of the advantages that extension of DigiD could bring, if additionally to Dutch governmental organizations, also Dutch private companies, and correspondingly their users would start to use it as an authentication framework. It should be indicated however, that together with the private companies, and their users, Dutch government would also obtain certain benefits from such an initiative. Hence, some of the major benefits that all the three parties would receive are listed below.

- **Multiple Authentication Methods**. Various organizations could use different methods of authentication offered by DigiD (i.e., username and password authentication, two factor authentication, etc.) based on the sensitivity level of the applications they would offer to their users. This benefits both users and organizations since they can use authentication mechanisms which are adequate to the sensitivity level of the corresponding application.

- **Single Sign on Authentication**. A user would be able to authenticate once using DigiD, and throughout the validity of her authentication session, access various applications without the need to re-authenticate. Additionally, users would not need to keep track of different authentication credentials for each organization. This clearly benefits the users since the ease of use of the overall system is increased.

- **Enhanced Security**. By maintaining a single authentication framework, users who use passwords for authentication, will more likely select strong passwords, and may start changing them on a frequent basis. This will benefit the security level of the overall system in general, and users' in particular.

- **Reduced Costs**. Companies would not need to implement their own authentication frameworks, and thus benefit from avoiding all costs related to building, maintaining and operating different authentication services. For example, according to one of the researches by Enterprise Management Associates (EMA), password management costs include $250 per year on average, for every computer user in most of the private organizations [48].

- **Fraud Detection and Prevention**. Extending DigiD to the private sector can benefit Dutch government by improving fraud detection and prevention activities, since, in case of suspected clients, private sector can provide the Dutch government with the identifiers of such users. DigiD would later be able to use such identifiers in order to obtain the real world identities of their holders, and conduct necessary legal activities with them whenever necessary.

- **Attribute exchange**. Additionally to the benefits listed above, private companies and their users could further receive other benefits from DigiD-2, if DigiD-2 would support a new feature like attribute exchange. That is, if based on the request of organizations, and the consent of corresponding users, DigiD-2 would supply participating organizations with various personal information of her users. This way, users would avoid typing such information each time it was requested, and the organizations would be ensured that the information provided by DigiD-2 is valid.

5.1.2 Overview of DigiD-2

In DigiD-2, similarly to current DigiD, there are three parties: (1) users, (2) DigiD-2 (i.e., identity provider - IdP), and (3) participating organizations (i.e., service providers - SPs). Service providers use DigiD-2, in order to authenticate their users, in a single sign on manner. That is, when a user desires to access an application at some SP, she gets redirected to DigiD-2 for authentication. The latter, after authenticating the user, supplies the corresponding SP, with the information related to the authenticated user. Such information can later be used by the SP, in order to know who was authenticated, and engage in an online transaction with her, without the need for re-authentication.

5.1.3 Pseudonyms in DigiD-2

In the context of current DigiD, the authentication information that DigiD sends to the service providers after she authenticates her users, typically (together with other parameters) contains the BSN number of the authenticated user. The BSN number is then used by the requesting SP, i.e., governmental organization, as a key in the citizen's database (GBA), in order to know who was authenticated and engage in an online transaction with her, if necessary. However, since in the context of DigiD-2, private companies are not allowed to use the BSN numbers, DigiD-2 needs to supply such organizations with identifiers that are different from normal BSN numbers. In the context of DigiD-2, such identifiers will be referred to as pseudonyms, and will be generated by DigiD-2. Each user will have a different pseudonym with each organization, and all of her pseudonyms will cryptographically be based on her BSN number. Note that none of the pseudonyms should leak any information about the underlying BSN number. Further technical requirements that the pseudonyms should satisfy are listed in Section 3.4.2, where also possible candidate methods for generating such pseudonyms are suggested.

5.1.4 One Time Linking of Pseudonyms with SP Accounts

Clearly, when a SP receives a user's pseudonym from DigiD-2, she can no longer use it as a key in the citizen's database in order to know who was authenticated. Therefore an SP needs a new means in order to relate the supplied pseudonym with the digital identity of the authenticated user. That is, the SP needs a method of verification, which allows her to relate (once) the received pseudonym, with the local account of the corresponding user. In order to solve this issue, we consider three scenarios. In the first scenario, a user has an existing local account and authentication credentials (e.g., username and password) at the SP, in the second scenario, a user has an account which contains some information that the SP can verify against user's identity (e.g., mobile phone number, home address), but does not have authentication credentials. In the final scenario, the user is completely unknown to the SP. Finally, it should be highlighted that the linking procedure is performed only on a one time basis. Since, in the future the SP will be able to directly identify the user's local account from the supplied pseudonym.

- **A User has an Existing Local Account, and Authentication Credentials at an SP.** Consider that a user already has a local account and authentication credentials (e.g., username/password) at some SP. Also, consider that this user was authenticated by DigiD-2, and that DigiD-2 had supplied the pseudonym of the authenticated user to such SP. After the SP receives the pseudonym of the authenticated user, she has to determine to which user's local account this pseudonym corresponds. For this, she can ask the user, to re-authenticate herself at the SP, using her local login information. After successful authentication, the SP will be able to link the local account of the authenticated user with the supplied pseudonym. Clearly, in the future, when the same organization receives such a pseudonym, she will directly be able to determine to which local user's account the supplied pseudonym corresponds to.

- **A User has an Existing Local Account, without Authentication Credentials at an SP.** Consider that a user has a local account (without authentication credentials) at some SP which contains information that the latter can verify against her identity (i.e., home address, or mobile phone number). Also, consider that this user was authenticated by DigiD-2, and that DigiD-2 had supplied the pseudonym of the authenticated user to such SP. After the SP receives the pseudonym of the authenticated user, she has to determine to which user's local account this pseudonym corresponds. For this, she first asks the user for her name. Next, she looks up for the account corresponding to the supplied name, and if an account with the verifiable information (e.g. mobile phone number, home address) exists, the organization sends out a verification code on such information (e.g. by SMS on users' mobile phone number, or by regular mail on user's existing home address). The user, correspondingly, will have to supply this verification code back to the SP during the account linking session. In case the user provides the organization with the correct verification information, the organization will be able to determine to which local account the supplied pseudonym corresponds to.

- **A User does not have a Local Account at an SP.** Consider that a user has neither local account, nor login information at some SP. In such case, a user can be forwarded to DigiD-2 for authentication, only if she initiates creation of a new local account at such SP. The SP, as a result, should be able to link the supplied pseudonym from DigiD-2, with the newly registered local account of the user. This can easily be performed, if the SP keeps track of the session where the user initiates creation of her account. After the user finishes filling in all her relevant details, she will be forwarded to DigiD-2 for authentication, as the final step of the account creation process. When DigiD-2 authenticates the user, she will send the corresponding pseudonym of the authenticated user to the SP. The SP, as a result, will be able to relate the received pseudonym with the local account that was being created before the user was redirected to DigiD-2. Note that such a linking procedure can be also performed before the start of the account creation session. That is, when a user requests to create an account at the SP, she will be first forwarded to DigiD-2, and as a result, the SP will receive the pseudonym of the authenticated user. Afterwards, the SP will ask the user to enter her new account details. After the account creation process is finalized, the SP will make a relation between the supplied pseudonym and newly created account. The benefit of this approach as compared to the previous one is that SPs do not need to temporarily store users' details, before account creation process is finalized.

5.1.5 A Typical DigiD-2 SSO Use Case

In this section we describe a typical DigiD-2 SSO use case. Note that a concrete specification of this use case, using SAML 2.0 Web SSO Profile using persistent pseudonyms and Artifact Binding will be given in Section 5.3.2. In our use case, which is similar to the use case described in Section 2.1.2.2, there is a user Joe, who has accounts at both DigiD-2, and some Organization A. At Organization A, user Joe, is registered as Joe1980, where he also has corresponding authentication credentials, in the form of a username and password. Organization A, has decided to use DigiD-2, for authenticating her users in a single sign on manner, for all her future transactions. Note that in this use case, we have used an assumption, that user Joe, has an existing local account, together with existing authentication credentials, at Organization A. Clearly, there can be scenarios, when Joe, neither has a local account, nor login information, at Organization A; or scenarios where Joe has an account at organization A, but does not have a login information. Based on the scenarios listed above, corresponding methods, as described in the previous section, can be used by Organization A, in order to link the pseudonym supplied by DigiD-2, with her user's local account.

1. User Joe decides to use DigiD-2 for his further interactions with Organization A. For this, he tries to access Organization A's web site, without having a valid authentication session. Since Organization A checks that user Joe does not have a valid authentication session, she forwards him (as a result of Joe's consent) to DigiD-2, for authentication.

2. When the user gets redirected to DigiD-2, DigiD-2 checks whether Joe has an active authentication session with her. If that is not the case, DigiD-2 authenticates Joe using specified authentication mechanism. However, if Joe has an authentication session with DigiD-2, then the authentication step is skipped and step 3 is evaluated.

3. After DigiD-2 authenticates Joe, she will first generate a pseudonym for him, which will be valid for Organization A. Afterwards, DigiD-2 will supply Organization A, with the result of the authentication, containing (together with other parameters) Joe's new pseudonym (azhqu900) generated for Organization A.

4. Organization A, after receiving the pseudonym (azhqu900) of Joe, will check if a user with the supplied pseudonym exists at such organization. If that is not the case, then organization A, will ask Joe to re-authenticate himself locally, in order to link the supplied pseudonym with his local account.

5. After successful authentication, Organization A, will be able to link the supplied pseudonym (azhqu900) with the account that corresponds to the locally authenticated user Joe1980. Cleary, in the future transactions, Organization A, will directly be able to link the supplied pseudonym (azhqu900) with the local account that corresponds to user Joe1980 (Section 5.1.4).

5.2 Functional and Non-Functional requirements of DigiD-2

Below, are the functional and non-functional (privacy and security) requirements of DigiD-2. Note that in the requirements listed below, DigiD-2 will be sometimes referred to as an identity provider (IdP), and the participating organizations will be referred to as service providers (SPs').

5.2.1 Functional Requirements

1. **Openness Requirements**
   - **a.** The new architecture of DigiD-2, should provide the widest support for: operating systems, programming languages, and network infrastructures.

2. **Single Sign On**[19]
   - **a.** A user, once authenticated at DigiD-2, should, during the lifetime of her authentication session with DigiD-2, be able to access desired services at all available SPs without the need for re-authentication.

3. **Attributes**
   - **a.** DigiD-2 should support the feature of transporting users' personal information (i.e., attributes), to the requesting SPs'.

4. **Dynamic Account Linking**
   - **a.** DigiD-2 should be able to provide a notice to the SPs' about the possible linking of users' local accounts at DigiD-2, with their accounts at corresponding SPs.
   - **b.** DigiD-2 should be able to provide a notice to the SPs' about the possible un-linking of users' local accounts at DigiD-2, with their accounts at corresponding SPs.

5. **Single Sign Off**[20]
   - **a.** DigiD-2 should allow its users to use a single sign off feature, as a result of which, users will be logged out from DigiD-2, and correspondingly, from all other SPs' where they had a valid authentication session.

5.2.2 Privacy Requirements

1. **Pseudonyms**
   - **a.** DigiD-2 should support mechanisms to create pseudonyms for her users.
   - **b.** Each user should have a different pseudonym for each SP.
   - **c.** A pseudonym of a user should be based on her BSN number[21].

2. **Attribute Release Privacy**
   - **a.** DigiD-2 should never release any user's attribute to a SP, without the consent of the corresponding user.

[19] Note that current DigiD does not satisfy this requirement (Section 4.2.3).
[20] This requirement can be satisfied using SAML 2.0 single logout protocol (Section 2.1.4.2)
[21] See Section 3.4.8.2 for a justification of this requirement.

5.2.3 Security Requirements

1. **Application Level Security**
   - **a.** The overall system should allow its users to use mechanisms (e.g., SSL/TLS), that allow interacting parties to authenticate each other, together with providing confidentiality and integrity properties of the transmitted data on the application layer.

2. **Message Level Security**
   - **a.** The overall system should allow its users to use mechanisms (e.g., XML encryption/signatures [51,52]), that allow interacting parties to authenticate each other, together with providing confidentiality and integrity properties of the transmitted data on the message layer.

3. **Multiple Levels of Authentication**
   - **a.** DigiD-2 should support the usage of one factor based authentication (e.g., username/password) for transactions with relatively low level of sensitivity.
   - **b.** DigiD-2 should support the usage of two factor (e.g., one-time passwords) based authentication for transactions with relatively average level of sensitivity.

5.3 SAML 2.0: A New Kernel for DigiD-2

5.3.1 Benefits of Using SAML 2.0 as a New Kernel for DigiD-2

Currently, DigiD uses A-Select [38] as its underlying framework. However, there are several motivations why using of SAML 2.0 as a new kernel of DigiD-2, should be considered more advantageous than the usage of A-Select. First of all, as it has been highlighted in Chapter 2, SAML 2.0 is an open standard, developed by several organizations and people, both from academia and industry. This means that it is more likely, that the resulting product is of adequate quality. Further, since SAML 2.0 is developed by several vendors, risks associated with facts that a certain vendor will not maintain or support it, are not applicable. Finally, since the standard is of open nature, it is open to security analysis, which reduces the risks associated with the insecure design of SAML 2.0; clearly since SAML 2.0 is an OASIS standard, its openness property has increased value as it attracts attention from wide audience. Also, note that there has been a recent publication [47] related to the security analysis of SAML 2.0 artifact based single sign on profile, which had highlighted that SAML 2.0 is "generally well written protocol", and "one of the most carefully designed browser-based protocols in federated identity management". Additionally, there are following technical benefits particularly related to the architecture of SAML 2.0 that make its usage reasonable as a new kernel for DigiD-2.

- **Platform Neutrality**. SAML 2.0 abstracts the security framework away from platform architectures and particular vendor implementations [5]. This is a good guarantee, that the openness requirement, (5.2.1-1.a), is met to the largest extent.

- **Web SSO Profiles**. SAML 2.0 supports Web SSO profiles (see Section 2.1.2) which were developed in order to allow for similar SSO use cases (Section 5.1.5) required for conducting typical DigiD-2 transactions. This fact can be exploited in order to straightforwardly implement DigiD-2 using such SAML 2.0 Web SSO Profiles.

- **Pseudonyms and Attributes**. SAML 2.0 supports elements and protocols for transporting users' pseudonyms and attributes respectively. Hence, these two new features can be embedded into the new deployment of DigiD-2 without any ad-hoc modifications to the actual SAML 2.0 protocols, or messages.

- **Message Layer Security.** Since, typical SAML 2.0 interactions are based on exchanging XML based user related authentication information between interacting parties, mechanisms like XML signature and encryption can easily be used, in order to satisfy the Message Level Security requirement (5.2.3-2.a).

- **Identity Federation.** SAML 2.0 supports existing mechanism that allow, both interacting parties, i.e., IdP and SP, to notify each other about the possible account linking procedures. This can be used, when DigiD-2 and SPs need to link pseudonyms of their users, with their corresponding local accounts at the SPs.

5.3.2 DigiD-2 SSO Use Case: SAML 2.0 Web SSO Profile and Artifact Binding

5.3.2.1 Motivations for Using the Artifact Binding[22]

As it has been highlighted in Chapter 2 of the thesis, SAML 2.0 has three major bindings that can be used for transporting SAML 2.0 messages on the communication channels, in order to allow interacting parties to communicate with each other. These bindings are: (1) POST binding – where SAML 2.0 messages are exchanged using the HTTP POST method, (2) GET binding – where SAML 2.0 messages are exchanged using the HTTP GET method, and (3) Artifact based binding – where SAML 2.0 messages are exchanged by back-end channels, using SOAP protocol and transported artifacts as references to these SAML 2.0 messages. For more details on these bindings an interested reader is referred to Chapter 2, Section 2.1.4.3. From these three bindings, the author of the thesis would like to highlight the Artifact based binding, and recommend it as a binding that should be used by DigiD-2 in order to conduct necessary transactions; clearly other bindings (POST, GET) can still be used if considered necessary. The reasons for such a recommendation are the following.

- **Limitations of Different Browsers on URL Lengths.** Different browsers put different limitations on the valid URL lengths. Since, GET binding, uses URL addresses for encoding SAML 2.0 messages inside it, this restriction can make such binding unusable for several browsers; especially, since the encoded SAML 2.0 messages typically are of a rather long length. With the artifact binding however, since instead of the actual messages, references to such messages are transported using URL's, the latter issue does not apply anymore, because such references typically have a rather small length. Note that this limitation can be avoided using POST binding as well, since in the POST binding, URL's are not used for transporting SAML 2.0 messages, rather SAML 2.0 messages are embedded in HTML forms. However, POST binding also suffers from certain limitations as elaborated below.

[22] Another major security related benefit is that a scientific paper from [47] had analyzed specifically Artifact Binding SSO Profile of SAML 1.0, and indicated that SAML "is one of the most carefully designed protocols in federated identity management". A corresponding security response to this paper from OASIS can be found here [65], which highlights that the attacks identified in the paper are not really dangerous, since if SAML were used according to its security specifications [4] these attacks can be avoided altogether.

- **Requirement for Active Content on the Browsers [48].** POST binding encodes SAML messages in the bodies of HTTP requests, and displays them to the user, using an HTML form. Users then have to manually press the “send” button in order to submit such messages to the receiving parties. However, this way, users have to perform additional (unnecessary) interactions with their browsers. In order to automate this process, i.e., automatically submit SAML 2.0 messages to receiving parties, one needs active content such as Java Script. Requirement for Java Script is also a limitation to the users, since several users might prefer to have it turned off in their browsers – due to various security reasons. By using the artifact based binding however, one does not face such an issue, since, the references to the actual messages, can be automatically exchanged between the interacting parties, using the GET binding, and the actual protocol messages, can be exchanged using the SOAP binding, without any additional user intervention.

5.3.2.2 Use Case Specification

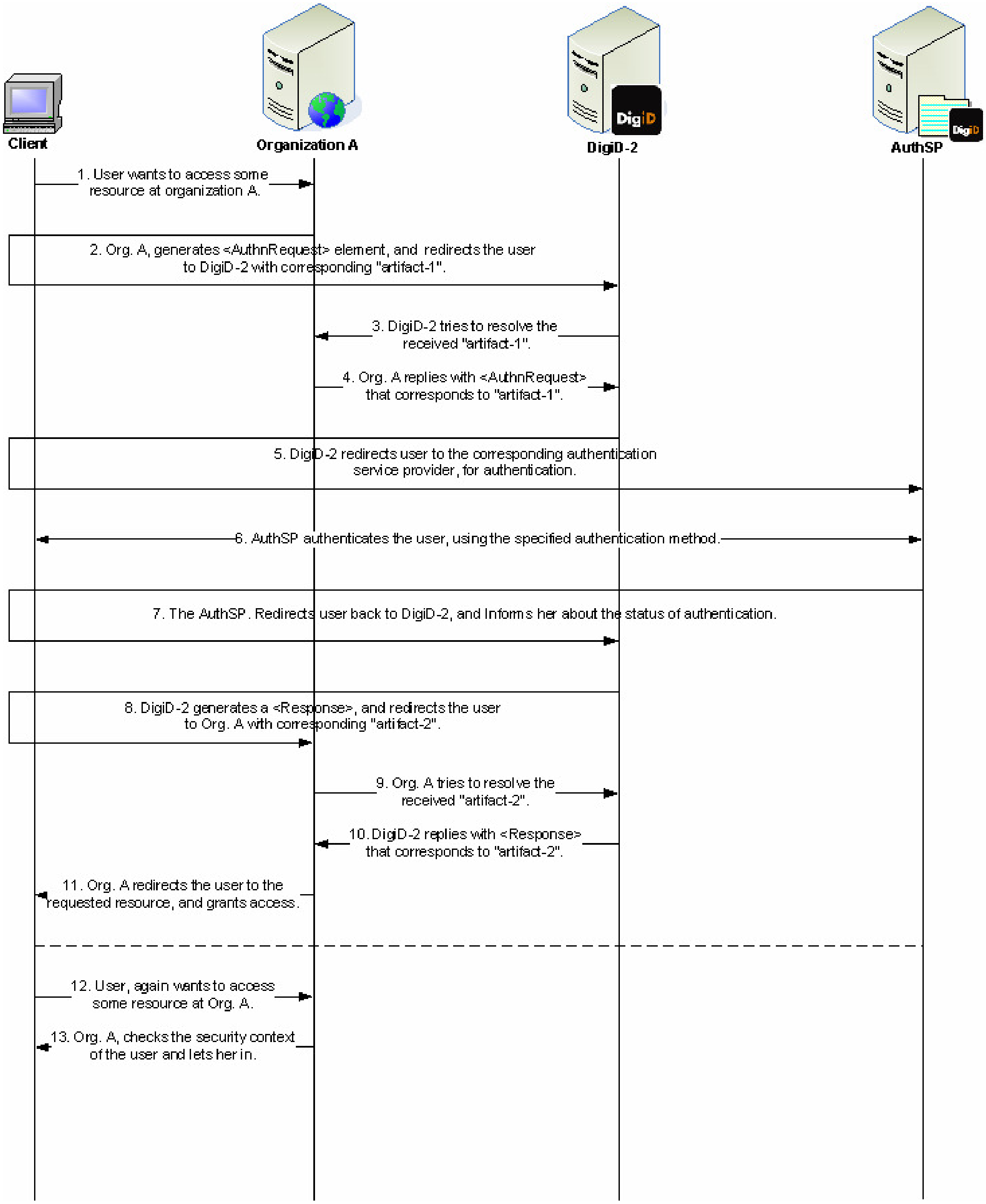
This section provides a concrete SAML 2.0 artifact binding based description of the single sign on use case described in Section 5.1.5. This use case is the same as the typical SAML 2.0 Web SSO profile [Section 2.1.4.4.1[23]] using Artifact Binding and persistent pseudonyms as user identifiers. For a full picture about such an SSO profile, i.e., which parameters should be included, and how they should be processed, an interested reader is referred to [3]. In our use case, the server of DigiD-2 and the server of Organization A, will be located on the following URLs’: http://www.DigiD-2.nl, http://www.Organization-A.nl. Two main SAML 2.0 protocols[24] that will be used in this profile, are the (1) Authentication Request Protocol: protocol used by the SP, in order to perform authentication of the user via DigiD-2, by exchanging the <AuthnRequest> and <Response> messages with her, and the (2) Artifact Resolution protocol: protocol used by Organization A and DigiD-2, for retrieving the actual SAML 2.0 messages to which exchanged artifacts correspond. This is done by transporting the <ArtifactResolve> and <ArtifactResponse> elements between each other correspondingly.

- **Step 1.** User Joe desires to access some resource, at Organization A, located on the following URL: http://www.Organization-A.com/myresource. Note that user Joe, does not have an existing authentication session with such an organization.

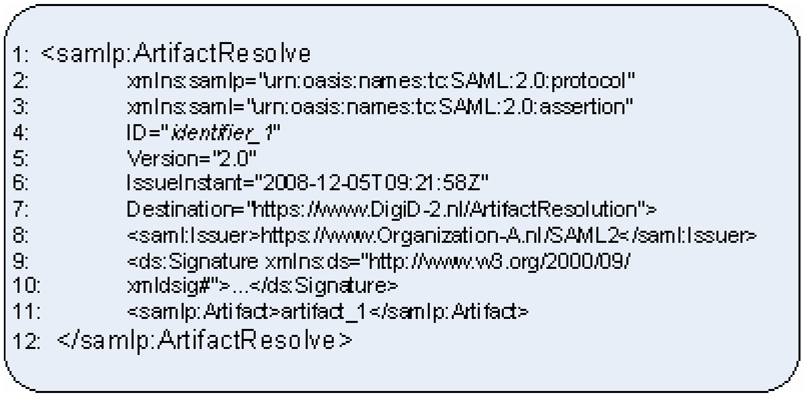
- **Step 2.** Organization A, checks that the incoming user does not have an existing authentication session, and based on this, decides to authenticate the user using DigiD-2. For this, first of all, she generates a <AuthnRequest> element, that is used to indicate to DigiD-2, to authenticate the forwarded user and generate a persistent pseudonym for her; this element is temporarily stored at Organization A. Afterwards, Organization A, generates an artifact (i.e., “artifact-1”), that will be used as a reference by DigiD-2, in order to obtain the generated <AuthnRequest> element using a back channel and the SOAP binding. Finally, Organization A, redirects the user to DigiD-2, together with the generated artifact attached to the redirection URL.
    - http://www.DigiD-2.nl/SAML2/Artifact?SAMLArt=artifact-1

[23] Note that the use case described in Section 2.1.4.4.1 is also similar to this use case, however, in DigiD-2 related use case, we use artifact binding as a communication channel, instead of redirect POST or GET bindings as it was in Section 2.1.4.4.1.

[24] Also, note that in this use case, the: name identifier management protocol will not be used. Since, we do not deal with the cases, when any of the user identifiers is required to be changed or declared as unusable dynamically.


Client
Organization A
DigiD-2
AuthSP
1. User wants to access some resource at organization A.
2. Org. A, generates <AuthnRequest> element, and redirects the user to DigiD-2 with corresponding "artifact-1".
3. DigiD-2 tries to resolve the received "artifact-1".
4. Org. A replies with <AuthnRequest> that corresponds to "artifact-1".
5. DigiD-2 redirects user to the corresponding authentication service provider, for authentication.
6. AuthSP authenticates the user, using the specified authentication method.
7. The AuthSP. Redirects user back to DigiD-2, and Informs her about the status of authentication.
8. DigiD-2 generates a <Response>, and redirects the user to Org. A with corresponding "artifact-2".
9. Org. A tries to resolve the received "artifact-2".
10. DigiD-2 replies with <Response> that corresponds to "artifact-2".
11. Org. A redirects the user to the requested resource, and grants access.
12. User, again wants to access some resource at Org. A.
13. Org. A, checks the security context of the user and lets her in.

[Figure 12]. A DigiD-2 SAML 2.0 artifact based single sign on use case, with successful authentication.



```
<samlp:ArtifactResolve
          xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
          xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
          ID="identifier_1"
          Version="2.0"
          IssueInstant="2008-12-05T09:21:58Z"
          Destination="https://www.DigiD-2.nl/ArtifactResolution">
          <saml:Issuer>https://www.Organization-A.nl/SAML2</saml:Issuer>
          <ds:Signature xmlns:ds="http://www.w3.org/2000/09/
          xmldsig#">...</ds:Signature>
          <samlp:Artifact>artifact_1</samlp:Artifact>
</samlp:ArtifactResolve>
```

[Figure 13]. A SAML 2.0 <ArtifactResolve> element.

- **Step 3.** When DigiD-2, receives “artifact-1”, she will try to request from Organization A, the actual message to which such an artifact corresponds to. For this, DigiD-2 sends the <ArtifactResolve> element, using a SOAP binding, to Organization A[25], with the received artifact (“artifact-1”) included inside. Such an <ArtifactResolve> message is represented in [Figure 13].

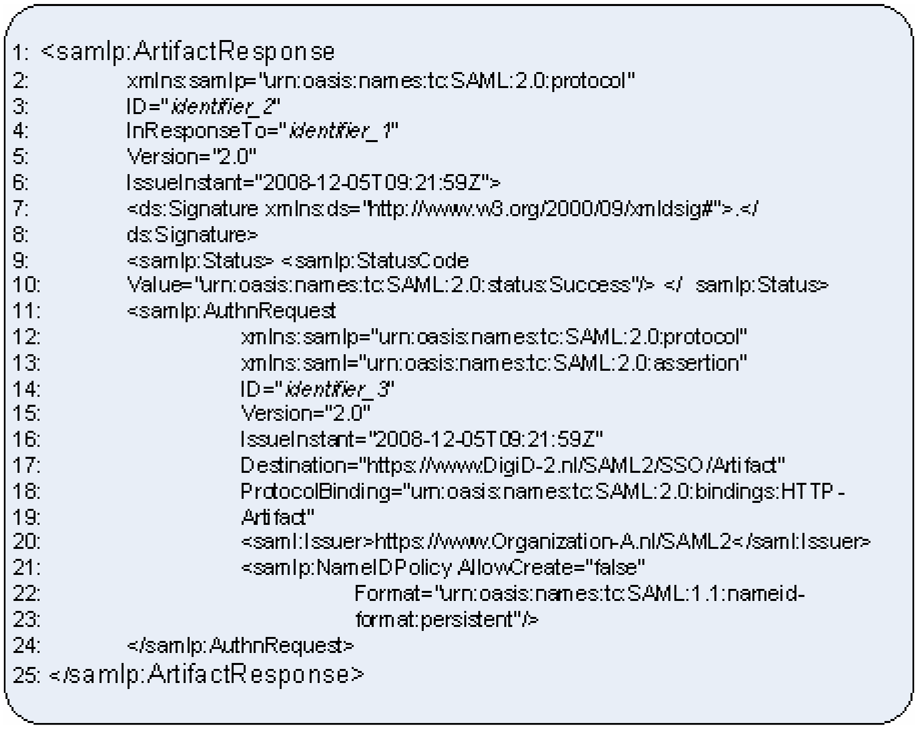
- **Step 4.** After Organization A will receive the <ArtifactResolve> element from DigiD-2, she will first obtain a message that corresponds to the requested artifact (“artifact-1”), which in our case, is the initially generated <AuthnRequest> element. Secondly, Organization A will embed the <AuthnRequest> element in a <ArtifactResponse> element, and forward it to DigiD-2, using direct SOAP binding. The <ArtifactResponse> element, together with the embedded <AuthnRequest> element, is represented on [Figure 14].

- **Step 5.** After DigiD-2 receives the requested <AuthnRequest> element, she will parse it correspondingly, and engage in an authentication procedure with the user. In our case, due to the type of generated <NemeIDPolicy> element, DigiD-2 knows that she has to generate a persistent pseudonym for the authenticated user, and insert it in the corresponding response message. DigiD-2 can forward the user to the authentication service provider for authentication then, but it can also be the case that the authentication service provider resides at the same server as DigiD-2; in our case however, we assume that there is a separate authentication provider[26].

- **Steps 6-7.** First the authentication service provider will check that the user does not have an active authentication session with her. If that is not the case, the authentication service provider will authenticate the user, and redirect her back to DigiD-2, together with the result of the authentication, embedded in the redirect message.

[25] Note that an artifact in SAML 2.0 contains the ID of the issuer of such an artifact as one of its components. Due to this, DigiD-2 knows to which service provider, she has to send the <ArtifactResolve> message to. For more details on the structure of the artifact, the interested reader is referred to [1, “artifact format”].

[26] Note that if there is a separate authentications service provider, the messages exchanged between her and DigiD-2 are out of scope of SAML 2.0. Clearly, if there is no separate authentication service provider, and DigiD-2 itself acts as the authentication service provider, exchange of such parameters is avoided at all.

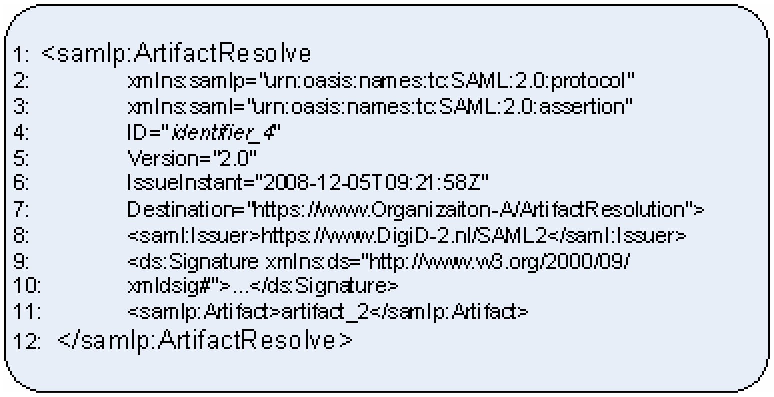


```
<samlp:ArtifactResponse
            xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
            ID="identifier_2"
            InResponseTo="identifier_1"
            Version="2.0"
            IssueInstant="2008-12-05T09:21:59Z">
            <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">.</
            ds:Signature>
            <samlp:Status> <samlp:StatusCode
            Value="urn:oasis:names:tc:SAML:2.0:status:Success"/> </  samlp:Status>
            <samlp:AuthnRequest
                        xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
                        xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
                        ID="identifier_3"
                        Version="2.0"
                        IssueInstant="2008-12-05T09:21:59Z"
                        Destination="https://wwwDigiD-2.nl/SAML2/SSO/Artifact"
                        ProtocolBinding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-
                        Artifact"
                        <saml:Issuer>https://www.Organization-A.nl/SAML2</saml:Issuer>
                        <samlp:NameIDPolicy AllowCreate="false"
                                    Format="urn:oasis:names:tc:SAML:1.1:nameid-
                                    format:persistent"/>
            </samlp:AuthnRequest>
</samlp:ArtifactResponse>
```

[Figure 14]. A SAML 2.0 <ArtifactResponse> element.

- **Step 8.** After DigiD-2 receives the authentication result of the user from the authentication service provider, she will generate a <Response> element for Organization A, which contains an <Assertion> indicating that user under pseudonym azhqu900 was successfully authenticated using the username and password authentication method. Such a <Response> element, similarly to step 2 is stored locally, and an artifact “artifact-2” is generated, that will be used by Organization A as a reference, in order to obtain the generated <Response> element from DigiD-2. The user is then redirected to organization A with the generated artifact attached to the redirection URL. Note that DigiD-2 will also create an authentication context (e.g. using cookies) for the user, in order not to require her to re-authenticate within the validity interval of the authentication session.
    - http://www.organizaion-A/SAML2/Artifact?SAMLArt=artifact-2

- **Step 9.** After Organization A receives “artifact-2”, she will try to request the actual message to which such artifact corresponds to. For this, Organization A sends the <ArtifactResolve> element to DigiD-2, with the received artifact (“artifact-2”) included inside. Such <ArtifactResolve> element is represented below.



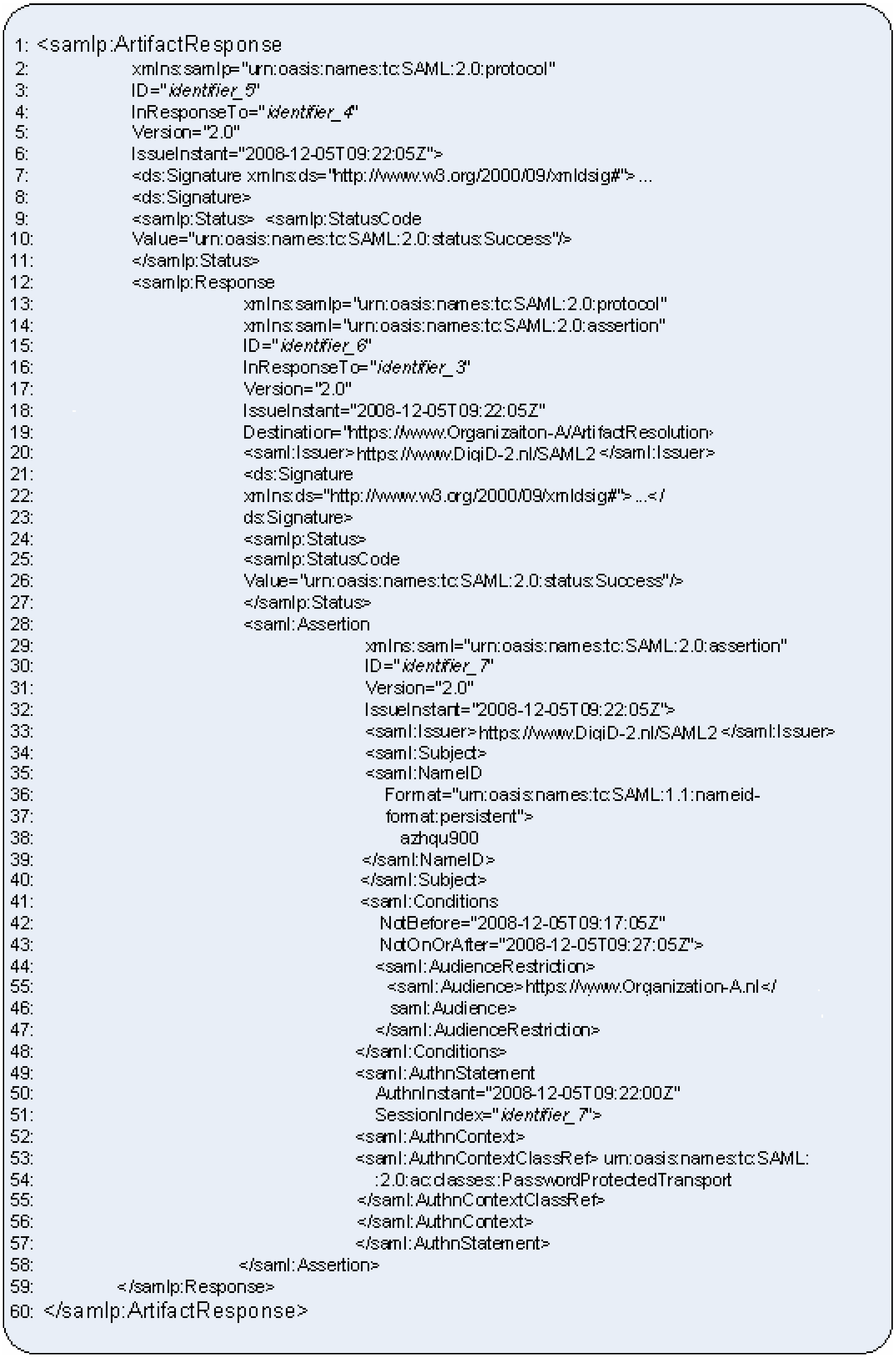
```
<samlp:ArtifactResolve
        xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
        xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
        ID="identifier_4"
        Version="2.0"
        IssueInstant="2008-12-05T09:21:58Z"
        Destination="https://www.Organizaiton-A/ArtifactResolution">
        <saml:Issuer>https://www.DigiD-2.nl/SAML2</saml:Issuer>
        <ds:Signature xmlns:ds="http://www.w3.org/2000/09/
        xmldsig#">...</ds:Signature>
        <samlp:Artifact>artifact_2</samlp:Artifact>
</samlp:ArtifactResolve>
```

[Figure 15]. A SAML 2.0 <ArtifactResolve> element.

- **Step 10.** After DigiD-2, receives the <ArtifactResolve> element from Organization A, she will first obtain a message that corresponds to the requested artifact (“artifact-2”), which in our case, is the <Response> element containing the authentication assertion from step 8. Secondly, DigiD-2 will embed such <Response> in the <ArtifactResponse> element, and forward it to Organization A, using direct SOAP binding. The <ArtifactResponse> element is represented in the way as shown in [Figure 16].

- **Step 11.** Organization A will parse the received authentication <Assertion>, redirect the user to the initially requested resource, and grant her access for this resource. Additionally, Organization A creates a security context, i.e., a cookie, which in the future, will help her to determine whether the user was recently authenticated or not.

- **Steps 12-13.** User Joe, wants to again access Organization A, however this time, he has an existing authentication session with such organization. Organization A, checks the security context of user Joe, and grants him access without the need to re-authenticate.

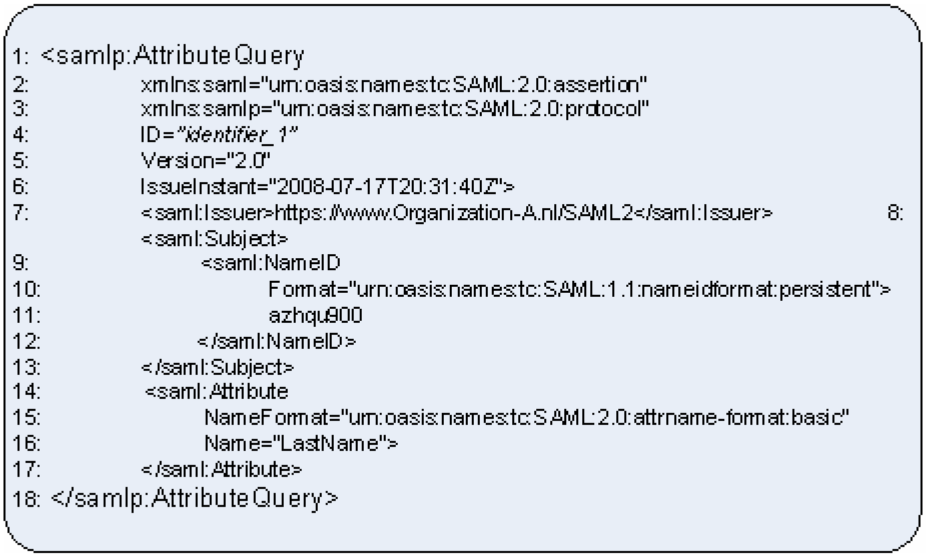


```
<samlp:ArtifactResponse
            xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
            ID="identifier_5"
            InResponseTo="identifier_4"
            Version="2.0"
            IssueInstant="2008-12-05T09:22:05Z">
            <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#"> ...
            <ds:Signature>
            <samlp:Status>  <samlp:StatusCode
            Value="urn:oasis:names:tc:SAML:2.0:status:Success"/>
            </samlp:Status>
            <samlp:Response
                        xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
                        xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
                        ID="identifier_6"
                        InResponseTo="identifier_3"
                        Version="2.0"
                        IssueInstant="2008-12-05T09:22:05Z"
                        Destination="https://www.Organizaiton-A/ArtifactResolution>
                        <saml:Issuer>https://www.DigiD-2.nl/SAML2 </saml:Issuer>
                        <ds:Signature
                        xmlns:ds="http://www.w3.org/2000/09/xmldsig#"> ...</
                        ds:Signature>
                        <samlp:Status>
                        <samlp:StatusCode
                        Value="urn:oasis:names:tc:SAML:2.0:status:Success"/>
                        </samlp:Status>
                        <saml:Assertion
                                    xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
                                    ID="identifier_7"
                                    Version="2.0"
                                    IssueInstant="2008-12-05T09:22:05Z">
                                    <saml:Issuer>https://www.DigiD-2.nl/SAML2 </saml:Issuer>
                                    <saml:Subject>
                                    <saml:NameID
                                      Format="urn:oasis:names:tc:SAML:1.1:nameid-
                                      format:persistent">
                                        azhqu900
                                    </saml:NameID>
                                    </saml:Subject>
                                    <saml:Conditions
                                      NotBefore="2008-12-05T09:17:05Z"
                                      NotOnOrAfter="2008-12-05T09:27:05Z">
                                      <saml:AudienceRestriction>
                                        <saml:Audience>https://www.Organization-A.nl</
                                        saml:Audience>
                                      </saml:AudienceRestriction>
                                    </saml:Conditions>
                                    <saml:AuthnStatement
                                      AuthnInstant="2008-12-05T09:22:00Z"
                                      SessionIndex="identifier_7">
                                    <saml:AuthnContext>
                                    <saml:AuthnContextClassRef> urn:oasis:names:tc:SAML:
                                      :2.0:ac:classes::PasswordProtectedTransport
                                    </saml:AuthnContextClassRef>
                                    </saml:AuthnContext>
                                    </saml:AuthnStatement>
                        </saml:Assertion>
          < /samlp:Response>
</samlp:ArtifactResponse>
```

[Figure 16]. A SAML 2.0 <ArtifactResponse> containing an authentication <Assertion>.

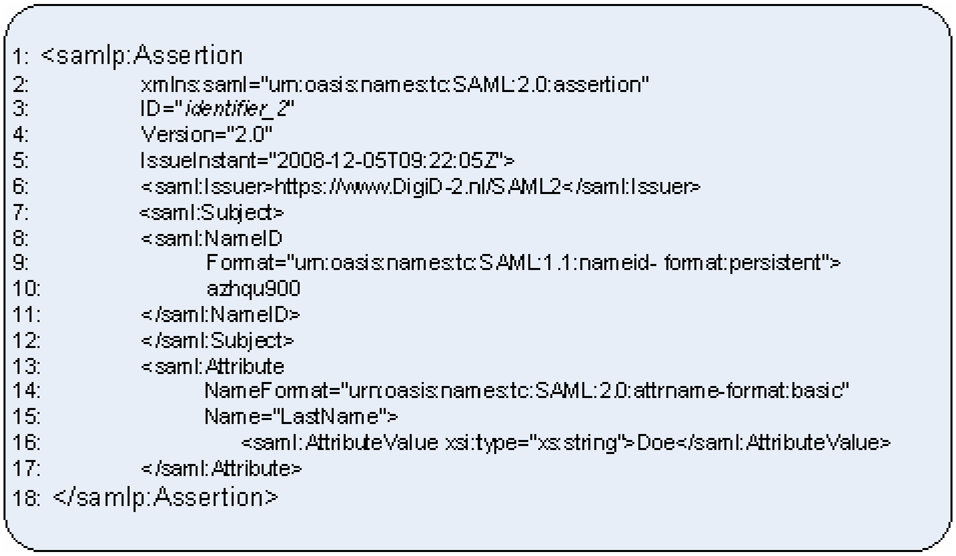
5.3.3 Requesting Attributes of a User

As it has been highlighted in Chapter 2 (Section 2.1.4.2), one of the protocols that SAML 2.0 supports is the assertion request and query protocol. Since, assertions can contain attribute information about the users, an SP can use such a protocol, in order to query the identity provider for various attributes about the authenticated user. Note that the request for the attributes, can be sent as a separate query (as shown below) or as a part of the authentication request protocol [8]. The type of attributes exchanged is determined by SAML 2.0 attribute profiles [3]. SAML 2.0 supports various attribute profiles, like: (1) basic attribute profiles: which allow for transportation of attributes with values as specified in the built-in XML Schema data types [49, Section 3] (e.g., string, boolean, decimal, float, etc.) (2) LDAP attribute profiles: which allow for transportation of attributes which can be used as parameters in LDAP directories (3) XACML attribute profiles: which allow for transportation of SAML 2.0 attributes that can be mapped automatically into XACML attributes and used as input to XACML authorization decisions, and etc; note that SAML 2.0 also allows for definition of third party attribute profiles, but from the interoperability perspective, this is not recommended. In the context of DigiD-2, we will limit ourselves only with basic attribute profiles. Below, is an example of an assertion query, where the SP asks DigiD-2 for the last name of the authenticated user (e.g., user that has pseudonym azhqu900) [Figure 17]. DigiD-2 replies with the corresponding response assertion [Figure 18], which contains the last name (Doe) of the user azhqu900, embedded in the <AttributeValue> element, as a string parameter.



```
<samlp:AttributeQuery
        xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
        xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
        ID="identifier_1"
        Version="2.0"
        IssueInstant="2008-07-17T20:31:40Z">
        <saml:Issuer>https://www.Organization-A.nl/SAML2</saml:Issuer>
        <saml:Subject>
            <saml:NameID
                Format="urn:oasis:names:tc:SAML:1.1:nameidformat:persistent">
                azhqu900
            </saml:NameID>
        </saml:Subject>
        <saml:Attribute
            NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:basic"
            Name="LastName">
        </saml:Attribute>
</samlp:AttributeQuery>
```

[Figure 17]. Querying DigiD-2 for the last name of user azhqu900.



```
<samlp:Assertion
            xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
            ID="identifier_2"
            Version="2.0"
            IssueInstant="2008-12-05T09:22:05Z">
            <saml:Issuer>https://www.DigiD-2.nl/SAML2</saml:Issuer>
            <saml:Subject>
            <saml:NameID
                    Format="urn:oasis:names:tc:SAML:1.1:nameid- format:persistent">
                    azhqu900
            </saml:NameID>
            </saml:Subject>
            <saml:Attribute
                    NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:basic"
                    Name="LastName">
                        <saml:AttributeValue xsi:type="xs:string">Doe</saml:AttributeValue>
            </saml:Attribute>
</samlp:Assertion>
```

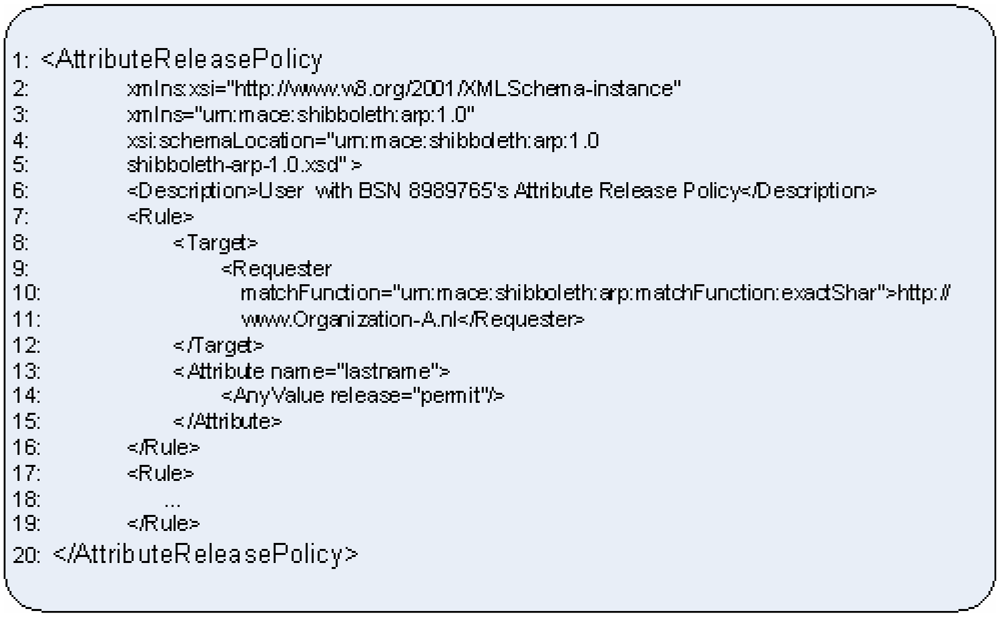
[Figure 18]. DigiD-2 replies with the requested attribute.

5.3.3.1 Controlling the Release of User's Attributes

As it has been highlighted in the requirements of DigiD-2, the release of users' attributes by DigiD-2 should be controlled by the users themselves, i.e., users should give consent on which attributes should be released to which SPs. This feature is rather important, since if this requirement is not fulfilled, most of the users of DigiD-2 will reject using DigiD-2 at all, due to the possible risk of unwanted exposure of their private data to various service providers.

One of the ways to solve this issue is to use the existing Attribute Release Policy (ARP) component used by Shibboleth [50]. ARPs are simple XML files, and are used to determine which attributes of the user should be released to which service providers; note that each user can have a separate ARP which applies to different SPs, and also each SP can have his own ARP which applies to all of her users – however, we will only use the first feature, since in the context of DigiD-2 only the users should be in control of their attributes. As an example of such an attribute release policy located at DigiD-2, consider [Figure 19], where a user's (with the specified BSN number 8989765) ARP indicates that she would like her last name only to be released to Organization A. As it can be seen from the figure, a single <Rule> element specifies which attributes which SP is allowed to see. Such a rule is being specified using two sub-elements: (1) <Target> element, which indicates to which SP this specific <Rule> element applies to, and (2) the <Attribute> element, which indicates which attribute the given SP is allowed to see. In our example, the service provider Organization A, is allowed to observe the last name of the user, no matter what value this attribute has – note that using ARP one could specify that release of some attribute (e.g., salary) is allowed, unless its value is equal to 1000[27]. However, in our scenario, we do not use this option. Finally, clearly, one may have several <Rule> elements within a single attribute release policy, which can specify which attributes a user is allowing to release to which service providers.

[27] E.g., this could be done by specifying <Value release="deny">1000</Value>", instead of <AnyValue release="permit"/>



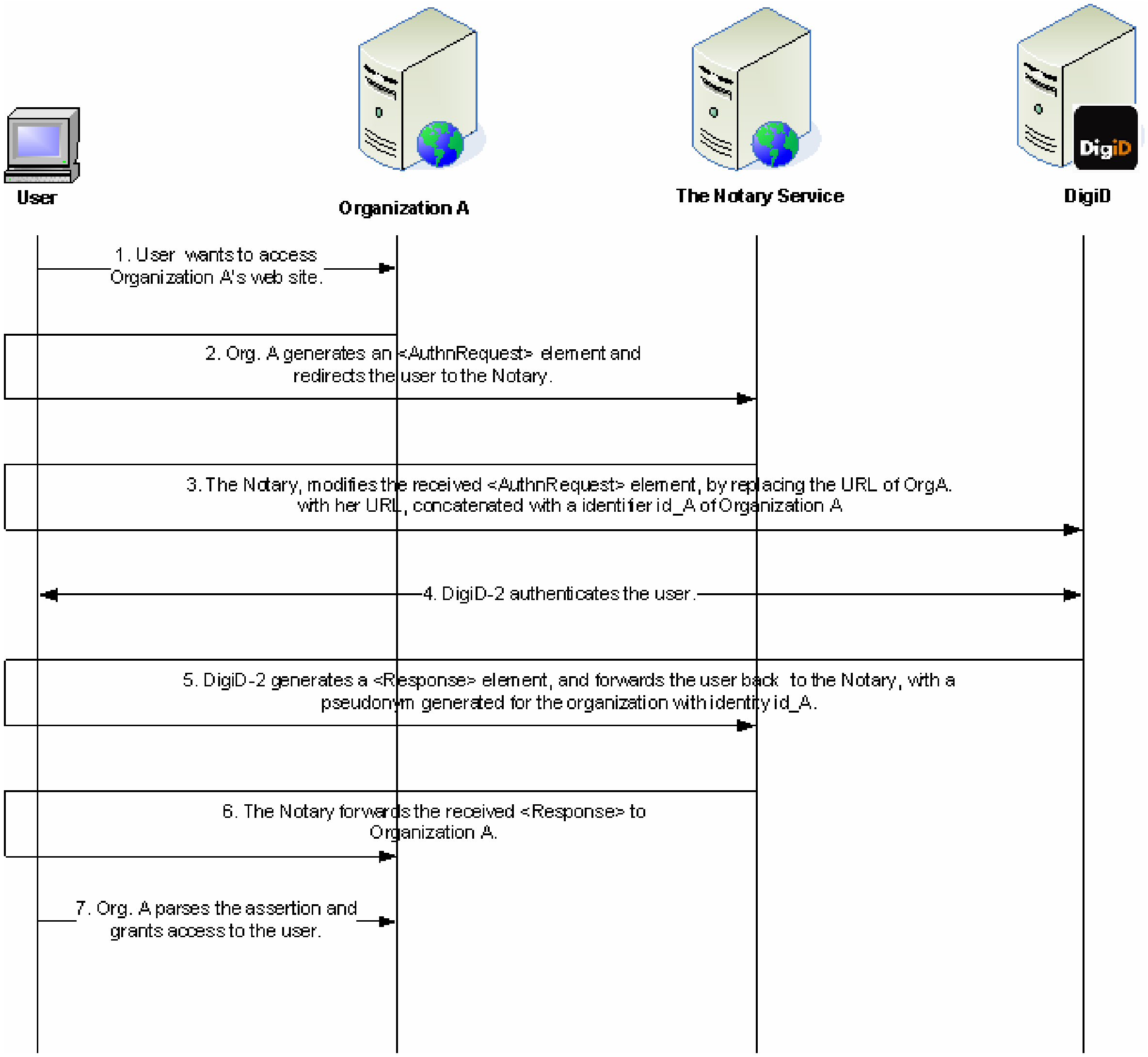
```
<AttributeReleasePolicy
        xmlns:xsi="http://www.w8.org/2001/XMLSchema-instance"
        xmlns="urn:mace:shibboleth:arp:1.0"
        xsi:schemaLocation="urn:mace:shibboleth:arp:1.0
        shibboleth-arp-1.0.xsd" >
        <Description>User  with BSN 8989765's Attribute Release Policy</Description>
        <Rule>
            <Target>
                <Requester
                  matchFunction="urn:mace:shibboleth:arp:matchFunction:exactShar">http://
                  www.Organization-A.nl</Requester>
            </Target>
            <Attribute name="lastname">
                <AnyValue release="permit"/>
            </Attribute>
        </Rule>
        <Rule>
              ...
        </Rule>
</AttributeReleasePolicy>
```

[Figure 19]. A attribute release policy of a user with BSN number 8989765,
who permits Organization-A to request his last name.

For more details, related on how this component can be used and integrated into an identity provider server can be viewed in [50]. Note that the setup of such policies, can be done beforehand, and also dynamically, that is, whenever a SP requests some attributes of the user from DigiD-2, DigiD-2 could display a modal dialog box to the user, asking her what action DigiD-2 should take with respect to the requested attributes. Afterwards, DigiD-2 would create or update a corresponding ARP.

5.3.4 Aside: A DigiD-2 SSO Use Case Using the Notary Service

As it has been highlighted in Chapter 3, all server centric single sign on systems, where an identity provider is used in order to authenticate users, forwarded to her from the SPs, suffer from the same important privacy issue. That is, the identity provider is able to trace, which SPs' the authenticated user is conducting transactions with. In this section, the author of the thesis suggests a way, which enables the users, to conduct necessary transactions, without the fear that they can be traced by the identity providers. This is achieved by introducing a trusted third party service in the system, which acts as a mediator between the interactions of the identity and service providers, and replaces the URL of the requesting SP with her URL, and attaches a pseudonymized identifier of the requesting SP to this URL; note that the identity provider does not know to which service provider the received pseudonymized identifier corresponds to, but she can still use such an identifier for generating SP specific pseudonyms of users. As a result, we receive an extra privacy protection for the users, where the Notary herself, does not know which user is being authenticated at DigiD-2, since DigiD-2 supplies the Notary with the pseudonym of the authenticated user, instead of her real world identity. In order to illustrate such a scenario, a corresponding single sign on use case is presented using SAML 2.0 POST binding; this binding is selected only for the purposes of ease of presentation. Also, since details of the exchanged messages are similar to typical SAML 2.0 single sign on profiles, details and examples of the exchanged assertions will not be provided and only a figure [Figure 20] illustrating the corresponding authentication flows will be given.


User
Organization A
The Notary Service
DigiD
DigiD
1. User wants to access Organization A's web site.
2. Org. A generates an <AuthnRequest> element and redirects the user to the Notary.
3. The Notary, modifies the received <AuthnRequest> element, by replacing the URL of OrgA. with her URL, concatenated with a identifier id_A of Organization A
4. DigiD-2 authenticates the user.
5. DigiD-2 generates a <Response> element, and forwards the user back to the Notary, with a pseudonym generated for the organization with identity id_A.
6. The Notary forwards the received <Response> to Organization A.
7. Org. A parses the assertion and grants access to the user.

[Figure 20]. A DigiD-2 SSO use case, with a Notary Service

5.4 Security Considerations of DigiD-2

5.4.1 Usage of SSL/TLS and XML Signature/Encryption

As it has been highlighted in Section 2.1.5, in order to provide effective level of security for SAML 2.0 transactions: (1) SSL/TLS with unilateral authentication – for transactions between the user and SAML 2.0 business entities (i.e., identity provider and service provider), and (2) SSL/TLS with bilateral authentication – for transactions between SAML 2.0 business entities has to be used. In order to additionally increase the security of SAML 2.0 transactions, and extend the security properties of the transmitted data from the application layer also to the message layer, one should use XML signature and encryption capabilities, as described in [51, 52]. The author would like to highlight that it is a good idea, if at least XML signatures are used next to SSL/TLS when an identity provider generates sensitive assertions. Since this way, the integrity and authenticity properties of the generated assertions are ensured throughout the lifecycle of such an assertion.

5.4.2 SAML 2.0 Based Ad-hoc Security Measures

There are several known scenarios, when usage of SSL/TLS or XML Encryption/Signature can not aid in protecting against certain threats in SAML 2.0. Such scenarios are mainly considered with the cases, when one of the legitimate parties (e.g., a service provider) misbehaves and tries to use his/her privileges, in a bad way. SAML 2.0 implementations advise to use the following optional SAML 2.0 mechanisms, in order to be secure against above mentioned scenarios.

- **<Destination>, <Audience> elements**: these SAML 2.0 assertion elements describe the entity (e.g., an identity provider, or a service provider), for which the corresponding assertion was generated. The purpose of the inclusion of this element in the SAML 2.0 messages is to avoid scenarios, where a legitimate but malicious party will be able to forward SAML 2.0 messages to the parties who were not the indented recipients of such a message. Clearly, when this element is included in the message, XML signature has to be applied in order to avoid the modification of this element by a malicious party.

- **Relay State mechanism**: as it has been highlighted in [47], A Service provider can only determine that a given assertion is generated for her organization. However, if such SP hosts multiple services, then one service can use the received assertion, in order to impersonate the user, at another service, which has a higher security level for instance. In order to avoid this issue, most of the SAML 2.0 bindings allow for transportation of Relay State data [1] which is typically used to identify the particular URL of the resource that the assertion is generated for.

- **<OneTimeUse>**: SAML 2.0 allows embedding one time use property elements into SAML 2.0 assertions [4]. Clearly, the receiving parties have to verify that they do not process SAML 2.0 assertions that have such a property twice. This method is effective against replay attacks. However, as indicated before, SSL/TLS could also protect against replay attacks on the application layer.

5.4.3 Security Considerations of the SAML 2.0 Artifact Binding

As indicated in Section 2.1.5, SAML 2.0 provides recommendations on the secure usage of each component of SAML 2.0 architecture [4]. Even though we have highlighted the most common secure usage guidelines so far, there are still some details related specifically to the SAML 2.0 Artifact Binding [“Section 6.5”, 4]. This is due to the fact, that in this binding first references (artifacts) to the actual messages are exchanged instead of directly exchanging the messages themselves.

- **One Time Use of Artifacts**. First of all, it should be indicated that all parties in the SAML 2.0 deployment should ensure that any artifact is used only once. This also effectively protects against replay attacks.

- **Querying for Non Existent Artifacts**. Since artifacts are random values, some party could try to guess certain values of artifacts. Even though guessing the artifacts is not possible due to their size, SAML 2.0 parties should try to raise alarm when they are frequently queried for non existing artifact values.

- **Protecting Against Impersonation Attacks**. Exchange of artifacts before exchanging the messages, gives us an extra protection, where a malicious SP can not forward an assertion to another SP. Since the legitimate SP could spot that the assertions are requested by another SP to which the artifacts were initially sent.

5.4.4 Discussion against Denial of Service Attacks

Implementations of SAML 2.0 are vulnerable to Denial of Service (DOS) attacks, whereby an attacker overloads the identity provider with several fake or useless SAML 2.0 messages (e.g., an <AuthnRequest> elements), resulting in the identity provider to spend substantial amount of time on processing these elements; this is especially a concern, since processing of SAML 2.0 elements is a very expensive operation [4]. In order to protect the IdPs from such scenarios, several different approaches can be taken.

- **Verifying XML signatures**. First of all, an IdP could verify XML signatures present on the SAML 2.0 message requests and combine it with access control, but since these procedures are rather expensive operations themselves, this method can itself result in a Denial of Service attack.

- **Shared Secrets**. Another method to protect against DOS could be, to let the IdP share different secrets with the service providers, such that the SPs would have to include such secrets in corresponding SAML 2.0 requests. Clearly, checking the validity of such a secret (i.e., just looking it up in a small database), would be more efficient against a signature verification.

In the context of SAML 2.0, other types of DOS attacks can be more dangerous than attacks initiated by malicious service providers. These types of attacks can occur if a set of attackers, use a legitimate set of service providers, in order to initiate authentication requests (e.g., automatically, maybe by using some scripts) with a single identity provider.

5.4.5 User Vulnerability Related Security Considerations

- **Phishing Attacks.** As it has been highlighted in Section 4.2.4.2, users can become vulnerable to phishing attacks, whereby an attacker creates a fake DigiD-2 web-site, and asks them to enter their authentication credentials inside. In order to protect against such scenarios, DigiD-2 could support the feature, which allows each user, to have a separate bitmap displayed on the web-site of DigiD-2 (i.e., sign on seals [45]). Users could then verify such bitmaps, anytime they are forwarded to DigiD-2, in order to ensure that they are visiting a legitimate DigiD-2 web site.

- **Selecting Weak Passwords.** The security of the overall system highly depends on the strength of the passwords (clearly if they are used as authentication credentials in certain situations) that the users will choose. Hence, it might be beneficial, to aid the users, in selecting strong passwords, or at least preventing them from choosing easy passwords. For instance, DigiD-2 could enforce following password selection rules: (1) Selected passwords should be at least 8 digits long, (2) Selected passwords should contain at least one numerical or punctuation character, (3) Selected passwords should contain at least one capital and lowercase letters, (4) Selected passwords should not contain any existing words that can be found in regular dictionaries.

Chapter 6

Conclusions

This chapter summarizes my thesis, and highlights important facts found out throughout the research. First of all, in Section 6.1, we mention why we were motivated to suggest extension of DigiD to the private sector, and describe problems and solutions related to such an extension. In Section 6.2, we highlight the major new features that we have suggested for DigiD-2. In Section 6.3, we provide a detailed explanation, on why we consider it important to extend DigiD as a browser based Federated Identity Management system, as compared to a private credential system, and also argue why SAML 2.0 is considered to be beneficial if used as a new kernel for DigiD-2. Further discussions are also provided on why maintaining the current kernel of DigiD, called A-Select, is considered to be a bad idea from the security perspective. Finally, in Sections 6.4 and 6.5, some of the open issues and recommendations for further research are provided.

6.1 Extending DigiD to the Private Sector (DigiD-2)

In this thesis, we have recommended to extend DigiD to the private sector, since we think that private companies, their users, and Dutch government, can obtain several benefits (Section 5.1.1) from such an extension. Next to the benefits, we have also identified the main issues related to such an extension: (1) since Dutch private companies can not use BSN numbers, DigiD-2 needs to supply them with alternate user identifiers (e.g., pseudonyms), and (2) since for Dutch companies, the supplied new identifiers appear as random numbers, they need a one time means in order to relate such identifiers with the local accounts of the corresponding users. Also, in Section 3.4.8.3 we have elaborated, why we consider it beneficial to cryptographically base such pseudonyms on the BSN numbers of the corresponding users. Further, in Section 3.4, we have suggested five pseudonym generation techniques (PGT). We have identified (Section 3.4.8.2) that PGT-1 is the most efficient both from the storage and implementation perspectives, as compared to other PGTs. However, PGT-1 does not satisfy[28] one of the pseudonym requirements listed in Section 3.4.2, namely the offline merge requirement. Other PGTs on the other hand are employed in order to satisfy all pseudonym requirements listed in Section 3.4.2. So, if achieving all of the pseudonym requirements is important for DigiD-2, then out of the remaining four PGTs, we have recommended the use of PGT-3, since it is more lightweight from the storage perspective as compared to other PGTs, and elegantly achieves all desired pseudonym requirements. Finally, as mentioned in the beginning of this section, when an organization receives a new pseudonym of the authenticated user from DigiD-2, she needs a mechanism in order to relate (once) such a pseudonym with the local account of the corresponding user. Several solutions to this issue have been suggested in Section 5.1.4, based on the information that such organization has about a certain user.

6.2 New Features for DigiD-2

- **Attribute Exchange**. We have suggested extending DigiD-2 with a feature which allows her to send available personal information of her users to the requesting service providers. However, we have indicated that in order to maintain the privacy of the users, DigiD-2 should employ mechanisms which allow users to control the release of their personal information to specified service providers; one of such control mechanisms has been discussed in Section 5.3.3.1.

- **Full Single Sign On**. We have also recommended (Section 5.2.1) that the users who authenticate with DigiD-2, should, during the validity of their authentication session, be able to access web-sites of any service provider without the need for re-authentication. It should be indicated that currently DigiD does not maintain this feature, since it does not even track the authentication status of the authenticated users (section 4.2.3).

- **Single Sign Off**. Finally, as a result of the above suggested feature, we have also recommended (Section 5.2.1) that DigiD-2 supports a single sign off feature, which allows users, to after logging out of DigiD-2, be automatically logged out from all service providers where they have active sessions; clearly this feature is not implemented in current DigiD.

[28] The offline merge property is not critical and can be achieved in an out of band manner as well. That is, to allow organizations merge, and ask DigiD-2 to reissue new pseudonyms for the users of the merged organizations.

6.3 Recommending a New Architecture for DigiD-2

6.3.1 Functional Motivations for Using SAML 2.0

First of all, due to the new functionalities of DigiD-2, we have decided to extend it to a Federated Management System. Next, as indicated in Chapter 1, currently there exist two types of FIM systems: browser based FIM systems and private credential systems. Browser based FIM systems are relatively easy to use as compared to private credential systems (Section 3.3.3), and can offer same security properties if properly deployed. Hence, since in the context of DigiD-2, both security and ease of use are rather important objectives, we have decided to extend DigiD-2 with a browser based FIM system. Consequently, out of the available browser based FIM systems, we have selected an OASIS standard SAML 2.0 which is a combination of the other browser based FIM systems, like SAML 1.1 and Liberty ID-FF 1.2. Another alternative to SAML 2.0 was a browser based FIM system, called WS-Federation. However, we have highlighted that it is not an OASIS standard yet, and also developers might consider SAML 2.0 to be easier to implement than WS-Federation (Section 2.4) since SAML 2.0 supports existing and well specified protocols and profiles which can be directly applied to different scenarios. As a result, one of the main benefits[29] of using SAML 2.0 as a new kernel for DigiD-2, is that it already supports web SSO profiles that exactly match the new context of DigiD-2, and hence, one of such profiles (Section 5.3.2), can be used in order to implement full functionality of DigiD-2.

6.3.2 Security Motivations for Using SAML 2.0

Next to the functional motivations, there are also security related motivations that have motivated us to recommend SAML 2.0 as a new kernel for DigiD-2. First of all, we consider that the current kernel of DigiD, called A-Select, has not undergone extensive security analysis and extensive security attention from a wider audience. The protocol that it relies on is not fully specified, and a security evaluation of the protocol is not present to the author's best knowledge. As a result, we consider (from a security perspective) it a bad idea[30], to rely such a sensitive application like DigiD, on a proprietary application like A-Select. On the other hand, since SAML 2.0 is an OASIS security related standard, it has gained strong attention from the wider audience, and even an academical publication [47] related to the security evaluation of one of its SSO profiles[31] is available; such a publication had indicated that SAML 2.0 is a "generally well written protocol" and "one of the most carefully designed browser-based protocols in federated identity management". Finally, the architecture of SAML 2.0 is fully specified, and guidelines on the secure usage of each of SAML 2.0 architecture components are available [4, 65] which allow the usage of SAML 2.0 in a manner that protects SAML 2.0 from all of the currently identified attacks.

[29] For further benefits that highlight why using SAML 2.0 is beneficial, the interested reader is referred to Section 5.3.1.

[30] As an example, in section 4.2.4.2 we have provided an attack, which could be performed if certain assumptions in DigiD hold. To the author's best knowledge, the existing documentation of DigiD does not make clear whether these assumptions are valid or not.

[31] It needs to be indicated that exactly this SSO profile has been recommended to be used, in order to implement new functionality of DigiD-2.

6.4 Open Issues

- **Denial of Service.** As indicated before, SAML 2.0 is vulnerable to Denial of Service Attacks. Consequently, several mechanisms have been suggested in Section 5.4.4, in order to protect against such attacks. However, a mechanism which can efficiently and permanently prevent these types of attacks has not been identified yet. Hence, together with the countermeasures discussed in Section 5.4.2, several other ways of protecting against DOS attacks can be further researched.

- **Phishing Attacks.** Even though we have suggested the usage of single sign on seals, in order to counter against phishing attacks, this problem still remains. Since, users need to be aware and motivated to check their selected bitmaps on the web-site of DigiD-2, in order to ensure the legitimacy of such a web-site. DigiD-2 correspondingly has to put strong efforts on educating and motivating users to do so. This can be achieved for instance, by informing users how likely it is that such an attack can occur, and warning them about the possible consequences of such an attack.

- **User Browsing Privacy Preserving FIM**. Since one of the main issues with browser based FIM systems, is that the identity provider is able to trace with which service providers the users are conducting transactions with, it is very likely that in our context, users are not satisfied with the fact that DigiD-2 can trace all their transactions. As a result, we have suggested a technical solution to this issue (Section 5.3.4), which allows users to perform transactions with service providers, such that DigiD-2 can not trace their behavior. However, it needs to be decided whether achieving such a privacy feature is worth, the performance and effort penalty that one will have to take in order to implement the proposed solution.

6.5 Future Research

- **Extending Attribute Exchange**. One could think to extend the attribute exchange feature of DigiD-2, such that one organization, can request attribute information about her users, from other organizations as well; instead of only DigiD-2. Technically, this is feasible, since the organizations could ask DigiD-2 for the requested attributes, and DigiD-2 could query other organizations for such attributes. One of the main reasons why all the queries should go through DigiD-2 is to enforce privacy of users, and make sure, that some organizations do not send out not necessary information to other organizations.

- **Identity Related Information Retrieval.** Users could also use DigiD-2, in order to query the participating private organizations (via DigiD-2) for the identity related information that such organizations have on their users. For this purposes, DigiD-2 could use the assertion query and request protocol of SAML 2.0 in order to retrieve all of the information about the users from different service providers, and then communicate the retrieved information to the users so that they are able to interpret such information using their browsers.

- **Usage of Transient Pseudonyms.** One could think of using transient (one time) pseudonyms of SAML 2.0 in situations where users want to conduct their transactions anonymously, and get access to some organizations only based on such pseudonyms or based on such pseudonyms and certain attributes of the users. The main difference of this approach as compared to current persistent pseudonym approach would be that since every time a new pseudonym is used, organizations do not have to link these pseudonyms with the local accounts of corresponding users, and thus users can maintain certain degree of anonymity in their transactions. Also in this case the generated pseudonyms can be just new random numbers, which do not even have to be based on the BSN numbers of the users. Finally, using such an approach an attribute based access control can be implemented.

- **Local Attribute Store.** As indicated, in browser based FIM systems, generated assertions have a short life time, since after their generation they are typically sent to the service providers in a short time period. On the other hand, in the private credential systems, the generated assertions have a long lifetime, and a separate software is needed in order to transfer these pseudonyms across organizations. It could be interesting to research ways, where users manage to store the generated assertions on their local machines, and make their transfer available using regular browsers, so that usage of local software is avoided at all and the privacy of the users' transactions would be respected since the identity provider would not be involved in the transfer of the assertions anymore.

References

1. Cantor, S., Hirsch, F., Kemp, J., Philpott, R., and E. Maler**, "Bindings for the OASIS Security Assertion Markup Language (SAML) V2.0"** OASIS Standard saml-bindings-2.0-os, March 2005.

2. Cantor, S., Kemp, J., Philpott, R., and E. Maler**, "Assertions and Protocol for the OASIS Security Assertion Markup Language (SAML) V2.0"** OASIS Standard saml-core-2.0-os, March 2005.

3. Hughes, J., Cantor, S., Hodges, J., Hirsch, F., Mishra, P., Philpott, R., and E. Maler, **"Profiles for the OASIS Security Assertion Markup Language (SAML) V2.0"**. OASIS Standard OASIS.saml-profiles-2.0-os, March 2005.

4. Hirsch, F., Philpott, R., and E. Maler, **"Security and Privacy Considerations for the OASIS Security Markup Language (SAML) V2.0"**. OASIS Standard saml-sec-consider-2.0-os, March 2005.

5. Madsen, P. and E. Maler, **"SAML V2.0 Executive Overview"**. OASIS SSTC Committee Draft sstc-saml-exec-overview-2.0-cd-01, April 2005.

6. E. Maler et al. **"Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) 1.1"**. OASIS, September 2003. OASIS Standard: oasis-sstc-saml-core-1.1.

7. E. Maler, et al. **"Technical Overview of the OASIS Security Assertion Markup Language (SAML) V1.1"**. OASIS, May 2004. OASIS Standard: sstc-saml-tech-overview-1.1-cd.

8. E. Maler et al. **"Bindings and Profiles for the OASIS Security Assertion Markup Language (SAML) 1.1"**. OASIS, September 2003. OASIS Standard: oasis-sstc-saml-bindings-1.1

9. Scott Cantor, et al. "**Liberty ID-FF Architecture Overview-1.2"**. Liberty Alliance, 2003. Document ID: liberty-idff-arch-overview-v1.2.pdf.

10. Robert Aarts, et al. "**Liberty ID-FF Protocols and Schema specification"**. Liberty Alliance, 2003. Document ID: draft-liberty-idff-protocols-schema-1.2-errata-v3.0.pdf

11. Robert Aarts, et al. **"Liberty ID-FF Bindings and Profiles specification"**. Liberty Alliance, 2003. Document ID: draft-liberty-idff-bindings-profiles-1.2-errata-v2.0.pdf

12. Hal Lockhart, et al. **"Web Services Federation Language (WS-Federation)-Version 1.1"**, December 2006. Available at: http://www.ibm.com/developerworks/library/specification/ws-fed/

13. IBM and Microsoft. **"Security in a Web Services World: A Proposed Architecture and Roadmap",** April 2002. Available at: http://www.verisign.com/wss/architectureRoadmap.pdf

14. **SOAP** (Simple Object Access Protocol) specification: http://www.w3.org/TR/soap/

15. **Hypertext Transfer Protocol** - HTTP/1.1, RFC document 2616. Available at: http://www.w3.org/Protocols/rfc2616/rfc2616.html

16. Ministry of Science, Technology an Education, Denmark. **“The Definition of open standards”**. Available at: http://www.projectliberty.org/liberty/about/open_standards_commitment

17. Lidong Chen, “**Access with Pseudonyms”.** In Cryptography: Policy and Algorithms, LNCS 1029, Springer-Verlag 1995, 232-243.

18. Birgit Pfitzmann and Michael Waidner. **“BBAE a general protocol for browser based_Attribute exchange”.** Research Report RZ 3455 (# 93800), IBM Research Sep. 2002.

19. Chaum, D., **“Security without identification: Transaction systems to make big brother obsolete”.** Communications of the ACM, 28(10):1030-1044, 1985.

20. Lysyanskaya, R. Rivest, A. Sahai, and S. Wolf. **“Pseudonym systems”.** In H. Heys and C. Adams, editors, Proceedings of SAC 1999, volume 1758 of LNCS, pages 184–99. Springer- Verlag, Aug. 1999.

21. Stefan Brands. **“Rethinking Public Key Infrastructure and Digital Certificates — Building in Privacy”**. PhD thesis, Eindhoven University of Technology, Netherlands, 1999.

22. B.Schoenmakers. **“Lecture notes on Cryptographic Protocols”**. February 8, 2008. Available at: http://www.win.tue.nl/~berry/2WC13/LectureNotes.pdf

23. Abhilasha Bhargav-Spantzel, Jan Camenisch, Thomas Gross, and Dieter Sommer. **“User centricity: a taxonomy and open issues”**. In DIM ’06: Proceedings of the second ACM workshop on Digital identity management”. Pages 1–10, New York, NY, USA, 2006. ACM Press.

24. Jan Camenisch, and Birgit Pfitzmann. **“Federated Identity Management”**. Chapter from the book: Security, Privacy, and Trust in Modern Data Management. ISBN 978-3-540-69860-9, June 12, 2007.

25. Pfitzmann and M. KÄohntopp. **“Anonymity, Unobservability and Pseudonymity A Proposal for Terminology”**. In Hannes Federath (Ed.), Designing Privacy Enhancing Technologies, Lecture Notes in Computer Science, LNCS 2009, pp. 19, Springer-Verlag, 2001.

26. S. Brands. **“A technical overview of digital credentials”**. Available online, February 20 2002. Available at: https://www.sics.se/~lra/thesis/library/brands02technical.pdf

27. S.Brands. **“Government Online”**. A Credentica whitepaper. Available at: http://www.credentica.com/whitepapers/GovOnline.pdf

28. D. Chaum and J.-H. Evertse. **"A secure and privacy-protecting protocol for transmitting personal information between organizations"**. In CRYPTO '86, vol. 263 of LNCS, pp. 118-167. Springer-Verlag, 1987.

29. Ivan Damgard**. "Payment systems and credential mechanism with provable security against abuse by individuals"**. In CRYPTO '88, vol. 403 of LNCS, pp. 328-335.

30. J. Camenisch and A. Lysyanskaya. **"An efficient system for nontransferable anonymous credentials with optional anonymity revocation."**. in Advances in Cryptology - EUROCRYPT 2001, ser. LNCS, vol. 2045. Springer, 2001, pp. 93–118.

31. Rivest, R., Shamir, A., and Adleman, L**. "A method for obtaining digital signatures and public-key cryptosystems."**. Comm. A CM 21, 2 (Feb. 1978), 120-126.

32. Alfred Menezes, Paul van Oorschot and Scott Vanstone, **"Handbook of Applied Cryptography."**. CRC Press, October 1996 – 5th reprinting, Aug. 2001.

33. L. Chen, I. Damgard and T. P. Pedersen**. "Parallel divertibility of proof of knowledge"**. In Advances in Cryptology - proceedings of EUROCRYPT 94, Lecture Notes in Computer Science pages ? - ?. Springer-Verlag, 1995.

34. J. Daemen and V. Rijmen. **"The Design of Rijndael: AES – The Advanced Encryption Standard."** Springer-Verlag 2002.

35. David Galindo, Eric R. Verheul. **"A practical pseudonym scheme segregating database unions and intersections."**

36. N. Koblitz. **"Elliptic curve cryptosystems"**. in Mathematics of Computation 48, 1987, pp. 203–209

37. J. C. Benaloh and M. de Mare. **"One-way accumulators: A decentralized alternative to digital signatures."** In T. Helleseth, editor, Proceedings of Eurocrypt 1993, volume 765 of Lecture Notes in Computer Science, pages 274–285. Springer-Verlag, May 1993.

38. A-Select. **"Operational Concept and System Description"**. Available at: http://a-select.surfnet.nl/doc/aselect1.4.1.ocd.html

39. A-Select. **"System Overview"**. Available at: http://a-select.surfnet.nl/aselect_overview.html

40. A-Select. **"Authentication Flows"**. Available at: http://a-select.surfnet.nl/functional_flows.html

41. A-Select. **"A-Select API"**. Available at: http://a-select.surfnet.nl/version/1.5/api/api.html

42. Handreiking. “**Aanwijzingen en specificaties DigiD**”. Version 1.6. Document number: 2005/6017.

43. Handreiking. “**Application Programming Interface. DigiD for burgers**”. Version 1.6. Document number: 2006/1124.

44. Vassiliki Koutsonikola and Athena Vakali. ”**LDAP: Framework, practices, and trends.** “. IEEE Internet Computing, 08(5):66–72, 2004

45. Rachna Dhajima, J.D. Tygar, “**The battle against phising: Dynamic security skins**”. Proceedings of the 2005 symposium on Usable privacy and security, Pittsburgh, Pennsylvania, 77-88, 2005

46. Birgit Pfitzmann, Michael Waidner. “**Federated Identity-Management Protocols — Where User Authentication Protocols May Go**”. 11Th Cambridge Workshop on Security Protocols, Cambridge (UK), 2003

47. T. Gross. “**Security analysis of the SAML single sign-on browser/artifact profile**”. Computer Security Applications Conference, 2003. Proceedings.19th Annual, pages 298-307, 2003.

48. J.Emigh, “**Single Sign On, Multiple Benefits**”, http://govtsecurity.com/mag/single_signon_multiple/

49. P. V. Biron et al. “**XML Schema Part 2: Datatypes**”. World Wide Web Consortium Recommendation, May/01. See http://www.w3.org/TR/xmlschema-2/

50. Scott Cantor. “**Attribute Release Policies**”. Jan 07/2008. https://spaces.internet2.edu/display/SHIB/IdPARPConfig

51. D. Eastlake et al. “**XML Encryption Syntax and Processing**”. World Wide Web Consortium. See http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/.

52. D. Eastlake et al. “**XML-Signature Syntax and Processing**”. World Wide Web Consortium, February 2002. See http://www.w3.org/TR/xmldsig-core/.

53. Darrel Hankerson, Alfred Menezes, Scott Vanstone. “**Guide to Elliptic Curve Cryptography**”. Springer, ISBN 0-387-95273-X. July 2003.

54. S. Goldwasser and M. Bellare, “**Lecture notes on Cryptography**”. Available at: http://www.cs.ucsd.edu/~mihir/papers/gb.html

55. J. Kohl, C. Neuman. “**The Kerberos Network Authentication Service (V5)**”, RFC 1510. Available at: http://www.faqs.org/rfcs/rfc1510.html

56. A. Pashalidis, C. J. Mitchell: “**A Taxonomy of Single Sign-On Systems**”. In (R. Safavi-Naini, J. Seberry Hrsg.): Information Security and Privacy – 8th Australasian Conference, ACISP 2003, Wollongong, Australia, 9.-11. Juli 2003, Proceedings, Springer-Verlag (LNCS 2727), Berlin, 2003, S. 249-264

57. Computer Associates (CA): Identity Manager. Available at: http://www.ca.com/us/identity-management.aspx

58. **IBM Tivoli Identity Manager**. Available at: http://www-306.ibm.com/software/tivoli/products/identity-mgr/

59. **Computer Associates (CA): eTrust Single Sign-On.** Available at: www3.ca.com/Solutions/Product.asp?ID=166

60. **Evidian**: AccessMaster. Available at: www.evidian.com/accessmaster

61. H.C.A. van Tilborg. **"Fundamentals of Cryptology: A Professional Reference and Interactive Tutorial".** Kluwer Academic Publishers , Boston etc., 2000. ISBN 0-7923-8675-2

62. **Microsoft Passport**. Available at: http://www.passport.net.

63. Federal Information Processing Standards Publications. **"Digital Signature Standard".** Jan 27, 2000.

64. Certicom Research. "**SEC 2: Recommended Elliptic Curve Domain Parameters.**" September 20, 2000. Version 1.0. Certicom Corp.

65. Scott Cantor. **"SSTC Response to "Security Analysis of the SAML Single Sign-on Browser/Artifact Profile"".** Document Identifier: sstc-gross-sec-analysis-response-cd-01. July 2005.

Appendix A

Mathematical Preliminaries

This appendix gives some background information that can be useful, for understanding several concepts described in Chapter 3. More precisely, the private credential systems, and suggested pseudonym generation techniques. In Section A.1, we give some relevant background information in Number Theory. In Sections A.2, A.3, and A.4 we cover several concepts from Group Theory, Cryptography, and Elliptic Curve cryptography respectively.

A.1 Number Theory [32]

1. Definition (Remainder and Quotient). If a and b are integers with $b \geq 1$, then ordinary division of a by b yields unique integers r (remainder) and q (quotient), such that relation (A.1) holds.

$$a=qb+r,\ 0 \leq r<b \tag{A.1}$$

2. Notation. The remainder (r) from the relation (A.1) is denoted as $a \bmod b$, and the quotient (q) is denoted as a *div* b.

3. Definition (Greatest Common Divisor). The largest positive integer that divides two given integers a, and b, is called the greatest common divisor of a and b, and is denoted as $gcd(a,b)$, or simply (a,b). In order to find the greatest common divisor of two given integers, one can use the Euclidean Algorithm [32, Chapter 2].

4. Algorithm (Extended Euclidean Algorithm). Extended Euclidean algorithm can be used in order to find the greatest common divisor of two integers a and b, and values x and y, such that relation (A.2) holds. This algorithm is described in (A.3).

$$ax+by=gcd(a,b) \tag{A.2}$$

Extended Euclidean Algorithm **(A.3)**

Input: Two nonnegative integers a and b. $a \geq b$.
Output: x,y,d, such that $ax+by=d$, where $d=gcd(a,b)$.

```
1. If ($b=0$) then set $d:=a,\ x:=1,\ y:=0$, and return ($d,x,y$).
2. Set $x_2:=1,\ x_1:=0,\ y_2:=0,\ y_1:=1$.
3. While ($b>0$) do:
    a. $q:=a$ div $b,\ r:=a-qb,\ x:=x_2-qx_1,\ y:=y_2-qy_1$.
    b. $a:=b,\ b:=r,\ x_2=x_1,\ x_1=x,\ y_2=y_1$, and $y_1=y$.
4. $d:=a,\ x:=x_2,\ y=y_2$, and return ($d,x,y$).
```

5. Definition (Congruence Relation: *a=b mod n*). Two integers a and b, are said to be congruent modulo an integer $n \neq 0$, if there exists an integer k, such that $a-b=kn$ (i.e., n divides $a-b$). The notation is given in (A.4). In other words, (A.4) means that if we divide a, by n, then the remainder is b.

$$a=b \bmod n \tag{A.4}$$

6. Algorithm (Finding Modular Multiplicative Inverse). Assume a and n are two integers such that $(a,n)=1$. In order to find an inverse a^{-1} to number a, modulo n, i.e., $aa^{-1}=1 \bmod n$, one has to apply Extended Euclidean Algorithm with input parameters (a,n), and take the result x as the inverse of element a, i.e., a^{-1}.

A.2 Group Theory [54, 22]

1. Definition (Group). A Group $(G,*)$[32], is a set with a binary operation $*:GxG\rightarrow G$, where the following properties hold.

- If $a,b\in G$, then $a*b\in G$.
- * is associative: $(a*b)*c=a*(b*c)$. $a,b,c\in G.$
- For all $a\in G$, there exists an identity element $I\in G$, such that: $a*I=I*a=a$.
- For all $a\in G,$ there exists an inverse element $a^{-1}\in G$, such that: $a*a^{-1}=a^{-1}*a=I\in G$.
- If additionally to the above properties, also $a*b=b*a$, then such group is called abelian. $a,b\in G.$

2. Definition (Subgroup). If G is a group, then S is called a subgroup of $G,$ if: (1) S is a subset of set G, and (2) S is a group in its own right with the same operation * as defined in G.

3. Definition (Group Order). The order of a group G, is represented by the number of its elements, and is denoted as $|G|$.

4. Fact. If S is a subgroup of G, then $|S|$ divides $|G|$.

5. Notation (Power of an element). If $a\in G,$ and m is some integer, then a^m represents a group element obtained by applying * operation on element a, m times.

6. Fact. For all $a\in G,$ if $|G|=m,$ then $a^m=I.$

7. Definition (Order of a Group Element). Order of an element $a\in G$, is the least positive integer m, such that $a^m=I$.

8. Example (Group Z_n). Consider the set with integers modulo integer n, $Z_n=\{0,1,\ldots,n-1\}$. This set is a group, with an operation * defined as addition modulo $n,$ i.e., a*b = $a+b \; mod \; n.$

9. Example (Group Z^*_n). Consider the set with integers less then n and which are relatively prime with $n,$ $Z^*_n=\{m: 1\leq m\leq n,$ and $gcd(m,n)=1\}$. This set is a group, with an operation * defined as multiplication modulo $n,$ i.e., $a*b = a*b \; mod \; n.$

10. Example (Group Z^*_p). Consider the same group Z^*_n, as defined in the previous example. However, assume that instead of integer n, one has a prime number p. Then clearly, due to the definition of Z^*_n, $Z^*_p=\{1,\ldots,p-1\}$. The operation * is clearly defined in the same way, i.e., multiplication modulo n.

11. Definition (Euler's Totient Function). Euler's totient function is denoted as φ, and equals the number of integers that are relatively prime to given integer n. i.e., $\varphi(n)=|Z^*_n|$.

12. Fact. Clearly, if p is some prime, then $\varphi(p)=|Z^*_p|=p-1.$

[32] For simplicity, all groups defined further on will have assumed that the operation defined in such groups is represented using *, hence the need from writing (G,*) will be removed.

13. Definition (Primitive Element). Consider a group G, and an element of this group a, with order m. Let's build the following subgroup of G, $<a>=\{a^0,a^1,a^2,a^3,....\}$. An element a, is called to be a primitive element (i.e., generator) of group G, if its powers generate group G, i.e., $<a>=G$.

14. Note. Since in the above definition, it is assumed that a has order m, i.e., $a^m=I$, then clearly $a^{m+1}=a$, which means that after $m+1$ steps, the elements of group $<a>$ start to repeat. This means that $<a>$ consists of m distinct elements, $<a>= \{a_0,\ldots,a_{m-1}\}$, and all other powers are repeats.

15. Definition (Cyclic group). A group G is called cyclic, if it has a primitive element.

16. Fact. Any group with a primitive order is cyclic.

17. Note (Computing Inverse of a Group Element). Consider there is a cyclic group $G=<a>$, with order m. If one wants to find an inverse a^{-i} of element $a^i\in G$, she has to take it as $a^{-i}=a^{m-i}$.

18. Definition (Discrete log). The discrete log of an element $y\in <g>$ with respect to g, is defined as the least nonnegative integer x satisfying $y=g^x$.

19. Assumption (Discrete Logarithm Assumption [22]). The Discrete Logarithm Assumption for group G states that it is hard[33] to compute x given a generator g and a random group element g^x.

20. Assumption (Computational Diffie-Helman Assumption [22]). The Computational Diffie-Helmman for group G, states that it is hard to compute g^{xy} given the generator g and random group elements g^x, g^y.

21. Assumption (Decisional Diffie Hellman Assumption [22]). The Decisional Diffie Hellman assumption for group G states that it is hard to distinguish g^{xy} from a random group element g^z given generator g and random group elements g^x, g^y.

[33] For the definition of term "hard", an interested reader is referred to [22, Section 1.2.3].

A.3 Cryptography [32, 22]

1. Definition (Hash Function [32]). A hash function *H*, is a function that satisfies following two properties.

- Compression: It maps input x of arbitrary finite bit length, into output *H(x)* with fixed bit length *n*.
- Ease of Computation: given *H*, and input *x*, it is easy to compute *H(x)*.

2. Definition (One Way Hash Function [32]). A one way hash function is a hash function, as defined above, and that additionally satisfies following properties.

- Preimage Resistance*:* It should be hard to find *x*, such that *H(x)=y*, given only the value of *y*.
- 2-nd Preimage Resistance*:* It should be hard to find *x'≠x*, such that *H(x')=H(x)*, given only the value of *x*.

3. Definition (**One Way Collision Resistant Hash Function [32]).** A one way collision resistant hash function is a hash function, as defined in the beginning of this section that satisfies the 2-nd preimage Resistance property, together with the following property.

- Collision Resistance*:* It should be hard to find input parameters *x,x'* with *x'≠x*, such that *H(x)=H(x')*.

4. Note. In practice, (cryptographic) hash functions, like SHA-256, satisfy all above stated requirements [22]. So, in the future, when we refer to cryptographic hash function, it means that we refer to a function that satisfies all above listed properties.

5. Definition (Commitment Scheme [22]). Let *commit:* $\{0,1\}^k x\{0,1\}^* \rightarrow \{0,1\}^*$ be a function with *k* as a security parameter. A non interactive commitment scheme *C*, consists of two protocols (see below) between a sender and a receiver. Literally, the idea behind the commitment schemes is that a player in a protocol is able to choose a value from some (finite) set and commit to his choice such that he can no longer change his mind [?].

- Commit Phase*:* A protocol in which the sender commits to a value $x \in \{0,1\}^*$ by computing *C=commit(r, x)*, where *r* is a random number chosen from $\{0,1\}^k$, and sending *C* to the receiver. The receiver stores *C* for later use.
- Reveal Phase*:* A protocol in which the sender opens commitment *C=commit(r, x)* by sending *r* and *x* to the receiver. The receiver computes *commit(r, x)* and verifies that it is equal to the previously received commitment.

Additionally, a commitment scheme must satisfy following two properties.

- Binding: That is, the probability of generating $r,r' \in \{0,1\}^k$, satisfying *commit(r, b)=commit(r', b')* should be negligible.
- Hiding, i.e., the distributions induced by *commit(r, b)* and *commit(r, b')* should be indistinguishable.

6. Definition (Zero Knowledge Proof). A zero-knowledge proof is an interactive method for one party (*e.g., prover*) to prove to another (*e.g., verifier*) that some statement is true, without revealing anything other than the condition of the statement. A zero-knowledge proof must satisfy three properties [22].

- Completeness*:* if the statement is true, the honest verifier will be convinced of this fact by an honest prover.
- Soundness: if the statement is false, no cheating prover can convince the honest verifier that it is true, except with some small probability.
- Zero-knowledge*:* if the statement is true, no cheating verifier learns anything other than this fact.

A.4 Elliptic Curve Cryptography [53, 36]

1. Definition (Elliptic Curve over Finite Field Z_p). An elliptic curve *E,* together with a special point *O* called infinity, over prime finite field Z_p, is defined by equation (A.4.1), where $a,b \in Z_p$, and equation (A.4.2) holds.

$$y^2 = x^3 + ax + b \tag{A.4.1}$$

$$4a^3 + 27b^2 \neq 0 \bmod p \tag{A.4.2}$$

2. Note. Note that elliptic curves can be defined over any finite field. However, for simplicity purposes, we will limit ourselves by describing elliptic curves defined over a finite prime field Z_p. For a more detailed treatment about elliptic curves an interested reader is referred to [36].

3. Definition (Additive Group Generated by an Elliptic Curve. $E(Z_p)$). The set $E(Z_p)$ that consists of: all points $(x,y) \in Z_p x Z_p$ that satisfy equation (A.4.1), an infinity point *O*, and an addition operation +, as defined below, forms an additive group of points defined over an elliptic curve *E* with *O* acting as a zero element.

- Zero Element*:* For all $P \in E(Z_p)$, $P+O=O+P=P$.
- Inverse Element: If $P=(x,y) \in E(Z_p)$, then $(x,y)+(x,-y)=$O. The point $(x,-y)$ is denoted as $-P$, and called the inverse element of *P*.
- Addition*:* If $P=(x_1,y_1) \in E(Z_p)$, and $Q=(x_2,y_2) \in E(Z_p)$, where $P \neq -Q$. Then $P+Q=(x_3,y_3)$, where x_3, and y_3, are defined by equation (A.4.3), see also [Figure 21].

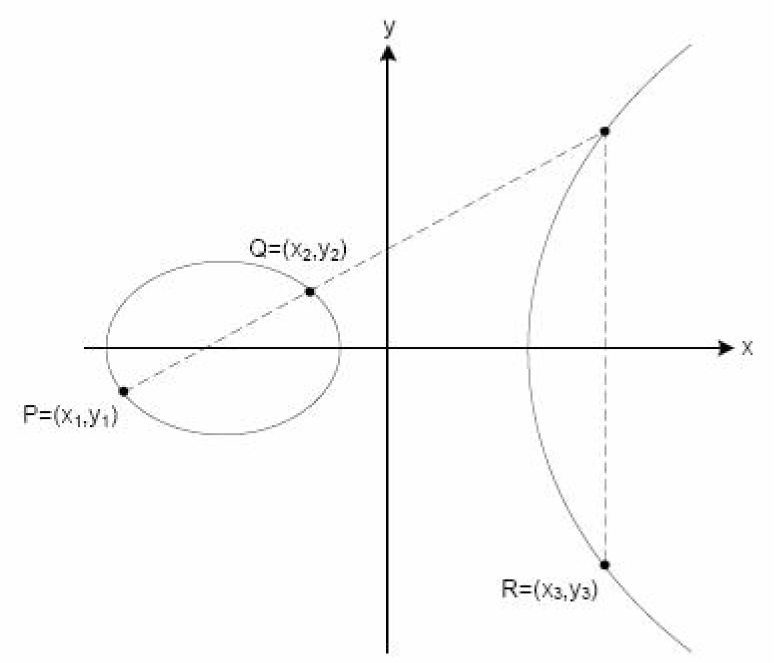
$$\begin{aligned} x_3 &= \lambda^2 - x_1 - x_2, \\ y_3 &= \lambda(x_1 - x_3) - y_{1,} \end{aligned} \quad and\ \lambda = \begin{cases} \dfrac{y_2 - y_1}{x_2 - x_1}, P \neq Q \\ \dfrac{3x_1^2 + a}{2y_1}, P = Q \end{cases} \tag{A.4.3}$$

4. Algorithm (Elliptic Curve Point Multiplication: *kP*). This algorithm describes how one can compute the value of *kP,* where *k>1* is an integer, and *P* is an elliptic curve point. First of all consider *k* is an *m* bit integer, and let $k=(k_{m-1}\ldots k_0)$ denote its binary coefficients. The algorithm is given below.

Elliptic Curve Multiplication: kP **(A.4.4)**

Input*:* integer $k=(k_{m-1}\ldots k_0),k>1$, and $P \in E(Z_p)$.
Output*:* $Q=kP$.

```
1. For ($i=m-1,\ldots,0$) do:
        a. $Q=Q+Q$ (note: initially, $Q$ is set as an infinity point).
        b. If ($k_i==1$) then $Q=Q+P$
2. return ($Q$).
```



[Figure 21]. A geometrical illustration of adding two elliptic curve points P and Q. R=P+Q.

5. Definition (Order of an Elliptic Curve). Let *E* be an elliptic curve defined over finite field Z_p. The number of points of such an elliptic curve, is called the order of elliptic curve *E*, and is usually denoted as $\#E(Z_p)$.

6. Definition (Order of an Elliptic Curve Point). Let *E* be an elliptic curve defined over finite field Z_p, and $P \in E(Z_p)$. The order of point *P*, is a smallest positive integer *n*, such that *nP=O*, where *O* is the point of infinity.

7. Definition (Cyclic Subgroup). Let *E* be an elliptic curve defined over finite field Z_p, and $P \in E(Z_p)$. Then, the set, $<P>=\{O,P+P,P+P+P,\ldots\}$ is called a cyclic subgroup of $E(Z_p)$ generated by *P*. Clearly, the number of distinguished elements in *<P>* is equal to the order of *P*.

8. Assumption (Elliptic Curve Discrete Logarithm Assumption ECDSA). Consider we have an elliptic curve *E*, as defined in the beginning of this section, and a corresponding additive group $E(Z_p)$ generated by the points of such an elliptic curve. Also, consider that *B* is any point that lies within this curve, i.e., $B \in E(Z_p)$, and the order of point *B* is *n.* Then, the discrete logarithm problem to the base *B* is the problem, given a point $P \in <P>$ to find (if it exists) an integer $k \in [0,n-1]$, such that *kB=P*.

9. Notes (Selecting Secure Elliptic Curves). The ECDSA assumption is an assumption on which the security of most elliptic curve cryptographic systems relies on. Thus, it is important, that in the group which is generated by a given elliptic curve, the ECDSA is not solvable in polynomial time. Several attacks which try to solve the ECDSA problem have been identified [53] to date, and below we give guidelines on selecting parameters of an elliptic curve, in order to effectively secure against such attacks.

- **Protecting Against Pohlig Hellman, and Pollard's rho attack [53]:** In order to protect against these attacks, the selected elliptic curve should be chosen such that the order of the generated group $E(Z_p)$ of elliptic curve points, i.e., $\#E(Z_p)$, is: (1) at least 160 bits long, and (2) is either a prime integer, or (3) almost prime integer, that is, a prime integer, multiplied by a small factor (i.e., 1,2,3,4).

- **Protecting Against Isomorphism Attacks [53]:** In order to protect against this attack, one should verify that $\#E(Z_p) \neq p$.

- **Protecting Against Weil and Tate Pairing Attacks [53]:** In order to protect against these attacks, one should ensure that $\#E(Z_p)$ does not divide p^k-1 for all $1 \leq k \leq 20$. Note that *20* suffices, if $\#E(Z_p) > 2^{160}$, else the way how the upper bound should be selected, an interested reader is referred to [53].

10. Algorithm (Embedding a plain text message into a point on an elliptic curve). In (A.4.5), an algorithm that describes how a plain text message *m,* with length *l* bits*,* can be embedded into an element of the $E(Z_p)$, where $p=3 \bmod 4$, is described. Note that this algorithm is probabilistic algorithm with a probability of failure$=0.5^k$.

Embedding a plain text message into a point on an elliptic curve (A.4.5)
(*p=3 mod4*)

Input*:* plain text message*: m.*
Output*:* an elliptic curve point *P,* which embeds *m* into the group element of $E(Z_p)$.

```
1. Let m' be an integer corresponding to the binary representation of m. $0 \leq m' \leq 2^l$.
2. Pick k, such that $0.5^k$ is small, and $m' < \frac{p-k}{k}$ .
3. For (j=0,1,…,k-1) do:
   a. $x_j = m'k + j$ modp;
   b. $w_j = x_j^3 + ax_j + b$ modp;
   c. $z_j = w_j^{\frac{p+1}{4}}$ modp;
   d. If ( $z_j^2 = w_j$ modp) then return $P=(x_j,z_j)$.
```

11. Note. If one wants to recover the plaintext message from the elliptic curve point $P=(x,y)$, she has to compute (5), where *k* is the same parameter as defined in the plaintext embedding algorithm.

$$m = x \text{ div } k \tag{5}$$

12. Note. Note that algorithm (A.4.5) works if for the prime number *p*, equation *p=3 mod 4* holds. However, there are other remaining alternatives[34], particularly, when *p=1 mod 8*, and *p=5 mod 8*. If *p=5 mod 8*, then the (A.4.5) algorithm still works, however, step (3.c) should be replaced by (6). If however, *p=1 mod 8*, then the interested reader is referred to [61, Section 9.5.2] for an alternative step which can be used instead of (3.c).

$$z_j = \begin{cases} c^{\frac{p+3}{8}} \bmod p : if, c^{\frac{p-1}{4}} = 1 \bmod p \\ 2c(4c)^{\frac{p-5}{8}} \bmod p : if, c^{\frac{p-1}{4}} = -1 \bmod p \end{cases} \tag{6}$$

13. Note (Selecting Elliptic Curves). There exist several documents which allow one to select existing elliptic curves with concrete domain parameters. These documents are: [63, 64], and the user can use our requirements mentioned above in order to select appropriate elliptic curves based on her needs.

[34] Since *p* is a prime number, then clearly, *p≠0 mod 4*, and *p≠2 mod 4* – else, *p* would be an even number. As a result, the only possibility remains that *p=3 mod 4*, which is equivalent to the conditions, *p=1 mod 8*, and *p= 5 mod 8*.