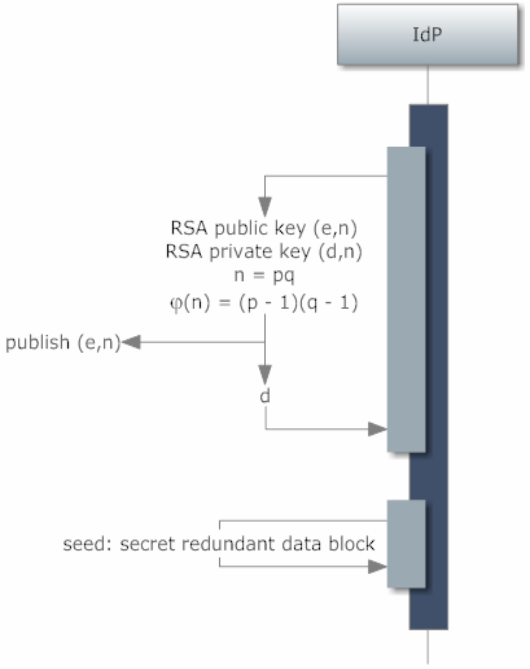
CHALMERS



Double SSO – A Prudent and Lightweight SSO Scheme

Master of Science Thesis in the Programme Secure and Dependable Computer Systems

SARI HAJ HUSSEIN

Chalmers University of Technology
University of Gothenburg
Department of Computer Science and Engineering
Göteborg, Sweden, November 2010

Double SSO – A Prudent and Lightweight SSO Scheme

SARI HAJ HUSSEIN

Examiner: DAVID SANDS

Chalmers University of Technology
University of Gothenburg
Department of Computer Science and Engineering
SE-412 96 Göteborg
Sweden
Telephone + 46 (0)31-772 1000

Cover:
Stage A of Double SSO, see Chapter 5

Department of Computer Science and Engineering
Göteborg, Sweden November 2010

Abstract

User authentication means the verification of a user identity in a computer system. In a typical scenario, users in an organization have access to several independent services, each of them requires separate credentials (e.g., user name and password) for user authentication. Users waste a considerable amount of time trying to recall their different credentials. The helpdesk workload caused by lost or forgotten credentials is also significant. Single Sign-On (SSO) was shown to be a successful authentication mechanism in networking environments where a large number of credentials would otherwise be required. SSO means that users authenticate only once and are granted access to the services they subsequently use without the need to re-authenticate. Obviously, SSO would increase users' productivity and satisfaction and reduce helpdesk calls. It also improves the usability of the system utilizing it. As a consequence, SSO has become an alluring feature called for by IT managers of organizations of various sizes.
In this thesis, we study the SSO technology from two analogous perspectives. In the first perspective, we view the technology from an industrial angle and introduce the knowledge necessary for an organization to determine its strategic SSO solution. We accomplish this by describing the taxonomies of SSO solutions and their qualities, in addition to presenting the architectures and operations of example SSO solutions in use today. In the second perspective, we move to (what we suppose) the next level and present our own SSO solution; namely Double SSO. Double SSO is a new SSO scheme designed to be lightweight, efficient and safe to implement in any wired or wireless networking infrastructure where SSO is needed, especially if the devices used in that infrastructure are resource constrained. This scheme appeals for a number of reasons. Of those reasons we mention; the minimum number of computations required and the minimum number of keys needed to accomplish the SSO experience, the ability to use digital identities of any type and to function in ubiquitous smart environments, and the immunity against known attacks.

Keywords: Single Sign-On, Single Sign-Out, Digital Identity, Credentials, Authentication, Uni-Factor Authentication, Multi-Factor Authentication, Identity Provider, Service Provider, Key to Kingdom Argument, Taxonomy, Authentication Authority, Authentication Server, Homogenous Environment, Heterogeneous Environment, Token, Public Key Infrastructure, Credential Synchronization, Credential Caching, Pseudo-SSO Component, Authentication Service Provider, Pseudo SSO, True SSO, Proxy, Kerberos, Web SSO, Enterprise SSO, Network SSO, Security Assertion Markup Language, Identity-Based Signature, Password-Based Identification Protocol, Challenge-Response Identification Protocol, Zero-Knowledge Identification Protocol, Identity Verification Scheme, Ubiquitous Smart Environment, Replay Attack, Man-in-the-Middle Attack, Weakest Link Attack, Forward Search Cryptanalytic Attack, DOS Attack, Repudiated Parties, Single Point of Failure, Certificate Revocation Problem, Implicit Certification, Private Key Escrow.

Acknowledgement

As an author of this thesis work, I would like to thank:

- My never-tiring supervisor and examiner Prof. David Sands for supporting me in different aspects, starting with encouragement, extending to fruitful discussions and effective support statements, and ending with containing my everlasting haste to get things done!
- Gunnar Åberg from Manodo AB (and from the nice north of Sweden!) for suggesting a proposal at short email notice, although he was looking for a full-time programmer!

As well, I would never forget:

- Lisbeth-Mariann Ekman (a.k.a. matte) for all the apple pies she has been baking for me for a year time now.
- Anyone else who walked with me once, even few footsteps (you may not know who you are).

Sari Haj Hussein
Gothenburg, 2010

Contents

Chapter 1

Introduction

As we progress in time, the dependence of our daily lives on computer-based information systems grows rapidly. As a consequence of this growth, the number of services we access on a daily basis is increasing. With a situation as such, authorities are demanding more precise control over this access. The answer to this demand is access control, which is a system that enables authorities to control our access to their services. Speaking about access to physical facilities, access control has become an everyday phenomenon. The simplest example is a lock on a car door. In the realm of computer security, access control comprises three processes; Authentication, Authorization and Audit (commonly abbreviated as AAA). In access control terminology, authorities protecting their services are called "objects", and we trying to access those services are called "subjects". Of course, both objects and subjects can be digital devices. Having this terminology, we can quickly define authentication, authorization and audit. Authentication specifies which subjects can log in to a computer system. Authorization specifies what those subjects can do after logging in. Audit specifies what those subjects did while logging in. This thesis work concentrates on the authentication process of access control.
To accomplish authentication, a subject should have a digital identity, which should be unique within its security domain. A digital identity (also called credentials) is simply a set of claims made by a subject about itself. If these claims are verified by the object, then the subject is authenticated successfully. The digital identity can have many forms. It can be something you know (for example, a password), something you have (for example, a smart card), something you are (for example, a fingerprint) or a location you are in (for example, inside or outside your company). Multi-factor authentication is a strong type of authentication that can be achieved by combining multiple different forms of digital identities, as opposed to uni-factor authentication which relies on a single form of digital identities. Achieving strong authentication is not the only concern of access control specialists though. As services proliferate in an organization, a user is required to maintain a set of digital identities to authenticate to each service. We can imagine the fatigue associated with maintaining this set and providing it over and over again for each service request.

1.1. Single Sign-On (SSO)

Single Sign-On (SSO) was suggested as a solution for this. It is a property of access control that enables a user to perform a single authentication to a service, and then get access to other protected services without the need to re-authenticate. With SSO, users' and administrators' lives become much easier as they will have to deal with a single digital identity for each user. A user will have to provide this digital identity only once per day. This will increase user's productivity. The maintenance of authentication data and enforcement of authentication policies become much easier with SSO, since authentications data will be centralized. Moreover, SSO reduces the chance that users will forget or lose their digital identities, therefore it reduces the risk of compromising a security system. The sad part of the story is that SSO has a disadvantage; namely the "Key to Kingdom argument". This argument means that if attackers manage to compromise an SSO authentication data store, they will gain access to all users' digital identities. Moreover, if attackers manage to obtain a single user's digital identity, they will gain access to all services protected by it. Despite the Key to Kingdom argument, SSO can be made extremely secure with careful planning,

implementation and administration. In fact, SSO is considered a valuable and indispensable security product in the overall IT security system nowadays.

1.2. Purpose, Contribution and Outline

SSO is the main topic of this thesis work. One purpose of our work is to give an objective and full picture of the SSO technology, and how it developed from a user convenient access control method into a crucial part of any IT infrastructure.

A large number of SSO solutions are available out-of-the-box today and choosing the right one is a challenging task that has many implications on the overall security strategy employed by an organization. Understanding this challenge, we start our thesis (in Chapter 2) with an explanation of the SSO taxonomies proposed so far by the academic security community. The taxonomies suggested by De Clercq [1][2] and Pashalidis-Mitchell [5] organize SSO solutions into generic categories, and thus expose (what otherwise could be overlooked) important differences in the security properties provided by those solutions. Having two different taxonomies does not suggest a contradiction at all. On the contrary, the two taxonomies harmonize and map perfectly to each other. This mapping is also shown in Chapter 2.

In agreement with what we present in Chapter 2, Chapter 3 includes an interesting taxonomy of qualities wanted in SSO solutions. This taxonomy enables security specialists to fine-tune the SSO choice they want to deploy in their organizations. Chapter 3 continues by listing the three salient categories of SSO solutions, and presenting example SSO solutions from each category. The author of this thesis work is no way biased towards any of the products that appear in Chapter 3, nor does he claim that they supplant or outperform other products in circulation. The author's aim in Chapter 3 is to give the reader a view of the architectures and operations of SSO solutions, while paying special attention to how these solutions obey the organization set forth by the SSO taxonomies presented in Chapter 2.

The other purpose of our thesis work is to deliver something new to the academic security community. Our new contribution is Double SSO, which is new SSO scheme designed to be lightweight, efficient and safe to implement in any wired or wireless networking infrastructure where SSO is needed, especially if the devices used in that infrastructure are resource constrained. Double SSO appeals for a number of reasons. Of those reasons we mention; the minimum number of computations required and the minimum number of keys needed to accomplish the SSO experience, the ability to use digital identities of any type and to function in ubiquitous smart environments, and the immunity against known attacks, such as the Forward Search Cryptanalytic Attack, the Replay Attack, the Man-in-the-Middle Attack and the Weakest Link Attack. Unlike most earlier SSO solutions, Double SSO can operate without the need to a Public Key Infrastructure (PKI) in place. You can imagine the cost reduction associated with this attractive property.

Chapters 4 and 5 of this thesis are entirely devoted to Double SSO. While Chapter 4 depicts the building blocks used in the design of the scheme, Chapter 5 includes its full description and security analysis. Unless the reader is familiar with the preliminaries of Double SSO that appear in Chapter 4, a reading of this Chapter is necessary to comprehend the textual description that appears in Chapter 5. The sequence diagrams included in Appendix A are useful whenever the reader finds it difficult to follow the message exchange in Chapter 5. While Chapters 2 and 3 are not absolutely necessary to understand what is said about Double SSO in Chapters 4 and 5, reading them teaches the reader where Double SSO stands under the wide umbrella of SSO. It further establishes the ground-truth of the appeal of this scheme compared to other SSO solutions. This thesis work is concluded in Chapter 7 where future line of research are also highlighted.

1.3. Milestone

This thesis work started with Manodo AB (http://www.manodo.se), a Swedish company that provides real estate companies with software solutions for metering heat, water and electricity consumption in building. Manodo's solutions are installed in over 10,000 properties in Sweden, Norway and China. The thesis proposal by Manodo has two versions. We obtained the first version on the 21st of July, 2010. An effective solution to the problem set forth by this version was conceived in a week time. A second and (more challenging) version was suggested on the 19th of August, 2010. The solution to this version came in two days time. In fact, both versions are related to SSO; however, the second version looks not only for an SSO solution, but also for a mechanism for transferring clients information from a remote site to Manodo's site. We started researching the literature relevant to SSO directly after obtaining the first version of the proposal, although a serious attempt to coin something new did not start until the end of August 2010, when we received valuable encouragement from our supervisor and examiner at Chalmers University of Technology, Prof. David Sands. A blog describing the development of our work should be accessible at http://ssospot.blogspot.com.

Chapter 2

Single Sign-On Taxonomies

Many SSO solutions have been proposed so far, and each one of them has its advantages and disadvantages. Two taxonomies were suggested to organize these solutions into generic categories; they are De Clercq's taxonomy and Pashalidis-Mitchell's taxonomy. The organization of SSO solutions into taxonomies is important, since it reveals important differences in the security properties supported by various SSO solutions, it creates a more structured context for the design of future SSO solutions and it enables the decision regarding suitable SSO solutions for a specific organization to be made wisely. In this chapter, we study the two taxonomies of SSO solutions. Then, we describe the relation between them, and finally we show how Double SSO is subsumed into each taxonomy.

2.1. De Clercq's Taxonomy of SSO

De Clercq [1][2] suggested the first taxonomy of SSO solutions. De Clercq's taxonomy is based on monitoring the complexity of SSO solutions. De Clercq determined this complexity using many factors. One factor is the nature of the environment; whether it is homogenous running the same platform and utilizing the same authentication protocol, or heterogeneous running different platforms and utilizing different authentication protocols. Another factor De Clercq used is the number of authentication authorities dealt with; single authority or multiple authorities. The last factor used in De Clercq's taxonomy is the number of credential sets the SSO solution handles; single set or multiple sets. De Clercq's taxonomy is described in details in the following sections.

2.1.1. Taxonomy-Related Terminology

To understand De Clercq's taxonomy, we need to define what an authentication server and an authentication authority are. An authentication server is a physical computing device whereas an authentication authority is a logical structure that comprises one or multiple authentication servers. Both an authentication server and an authentication authority are trusted enough by users to perform authentication functions.

2.1.2. Simple SSO Architectures

A simple SSO architecture runs in a homogeneous environment and deals with a single authentication authority. In simple SSO architectures, we can have a single authentication server with a single credentials database as shown in Figure 2.1.

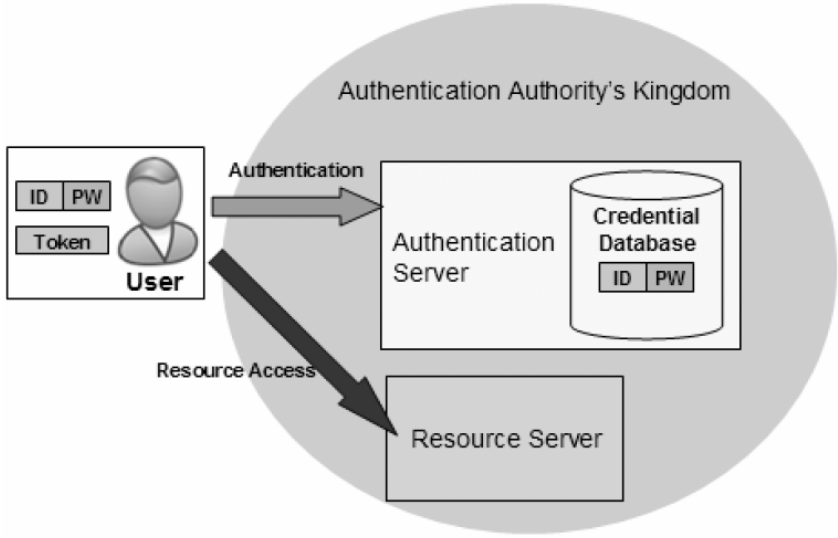


Figure 2.1. A simple SSO architecture in an environment with a single authentication server. Figure courtesy of [2]

Or we can have multiple authentication servers with multiple replicated credentials databases as shown in Figure 2.2.

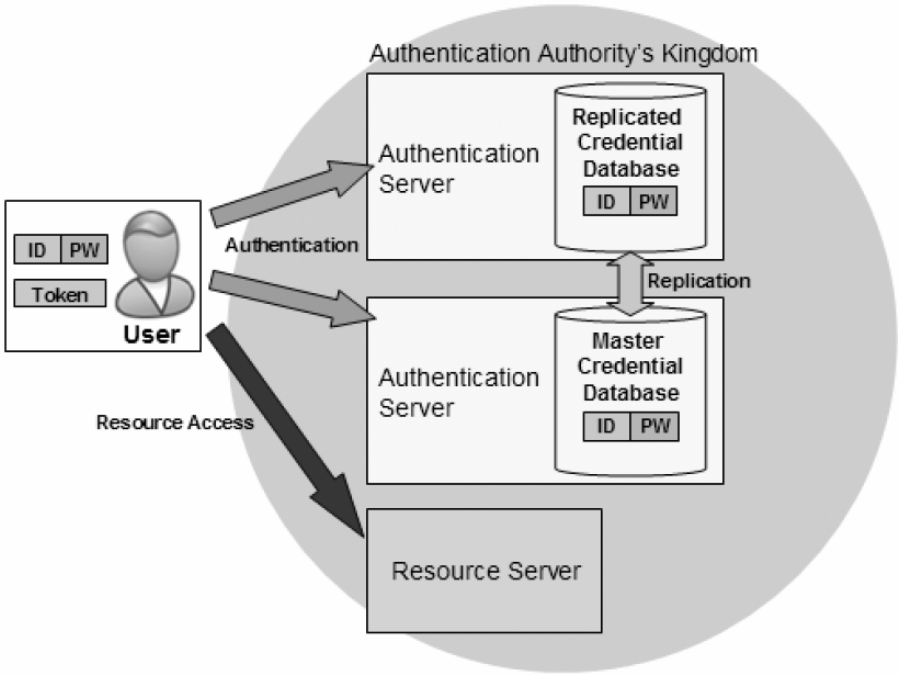


Figure 2.2. A simple SSO architecture in an environment with multiple authentication servers. Figure courtesy of [2]

Of course, the latter provides for better performance, scalability and availability. Note in both figures how a user has a single set of credentials and how she submits them to the authentication authority. The authentication authority will then validate the credentials using the data stored in the credentials database. If there is a match, the user's identity is considered authentic and she is granted access to the resource server. Later, the authentication authority can issue a token to the user. This token is basically an encrypted string that proves that the user was authenticated by the authentication authority. Simple SSO architectures can be easily implemented in networks where all computers are running the same operating system.

2.1.3. Complex SSO Architectures

A complex SSO architecture runs in a heterogeneous environment and deals with multiple authentication authorities running on different platforms and utilizing different authentication protocols. Figure 2.3 shows how authentication is done in a heterogeneous environment with multiple authentication authorities but without SSO. Note how the user maintains two different sets of credentials, one for each authentication authority.

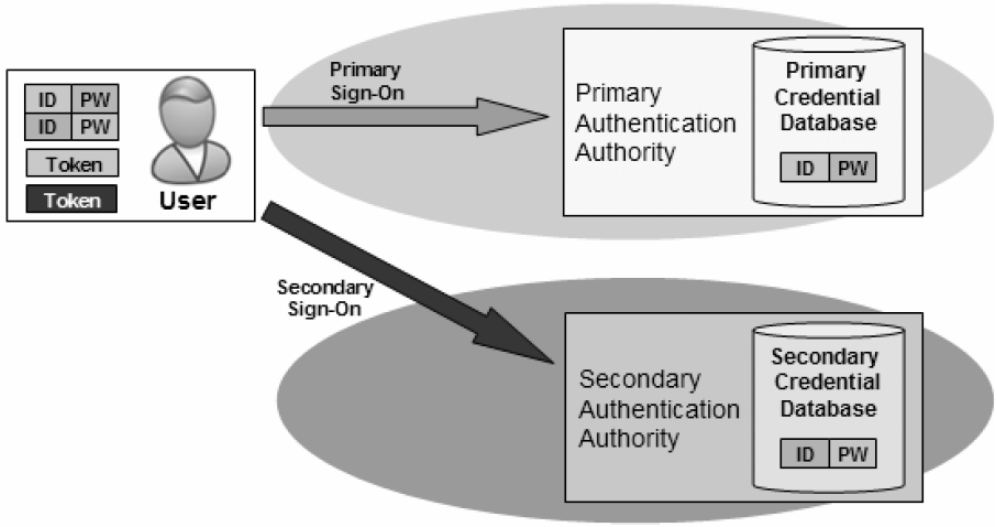


Figure 2.3. Authentication in an environment with multiple authentication authorities but without SSO. Figure courtesy of [2]

Complex SSO architectures can be grouped into those handling a single set of credentials and those handling different sets of credentials. We discuss each group separately.

2.1.3.1. Complex SSO Architectures Handling a Single Set of Credentials

These architectures use a single set of credentials recognized by multiple authentication authorities. There are two types of these architectures.

1. Token-Based SSO Architecture

In this architecture, a user gets a temporary token when she authenticates successfully to the primary authentication authority. This token can be later used to authenticate her to one or more secondary authentication authorities. Secondary authentication authorities validate the user's token using a symmetric cryptographic method that is based on secret key material shared between the primary and the secondary authentication authorities. In an HTTP environment, a token-based SSO architecture can be provided using HTTP cookies that store the tokens. The Kerberos protocol in which a Kerberos Key Distribution Center (KDC) plays the role of a primary authentication authority that issues what they call tickets (instead of tokens) is an example of this type of architecture. Authentication in a token-based SSO architecture is illustrated in Figure 2.4.

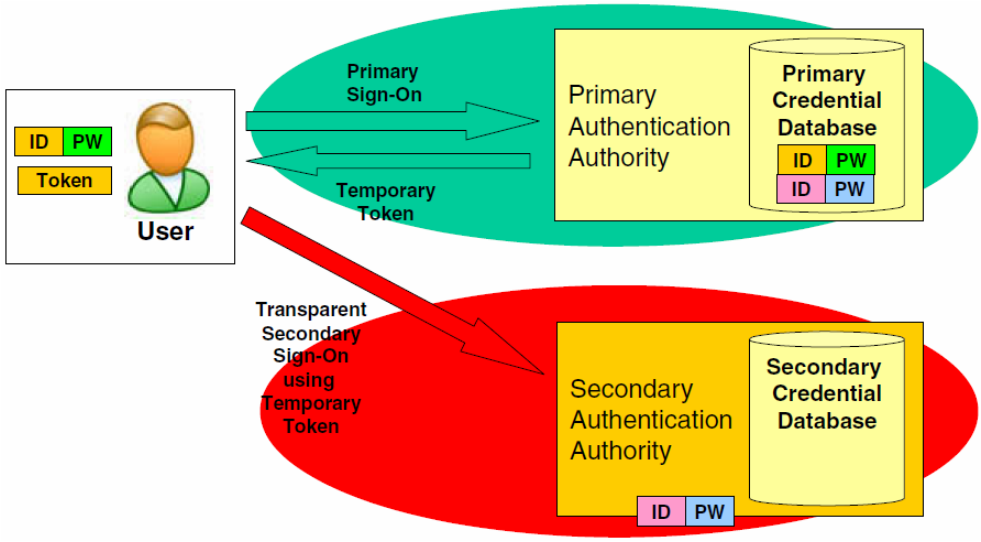


Figure 2.4. Authentication in a token-based SSO architecture (transparent means that the sign-on is executed by the SSO architecture without any user intervention). Figure courtesy of [2]

2. Public Key Infrastructure-Based SSO Architecture

In this architecture, a user first registers with a special authentication authority, called the Certification Authority (CA). When the CA receives user's credentials and public key, it generates a public key certificate and sends it back to the user. Later, the user's public key certificate and private key are used to generate a token the user uses to authenticate to one or more secondary authentication authorities. Secondary authentication authorities validate the user's token using an asymmetric cryptographic method that resembles a major difference between this architecture and token-based SSO architectures. Microsoft PKI in Microsoft Windows Server 2003 R2 is an example of this type of architecture. Authentication in a public key infrastructure-based SSO is illustrated in Figure 2.5.

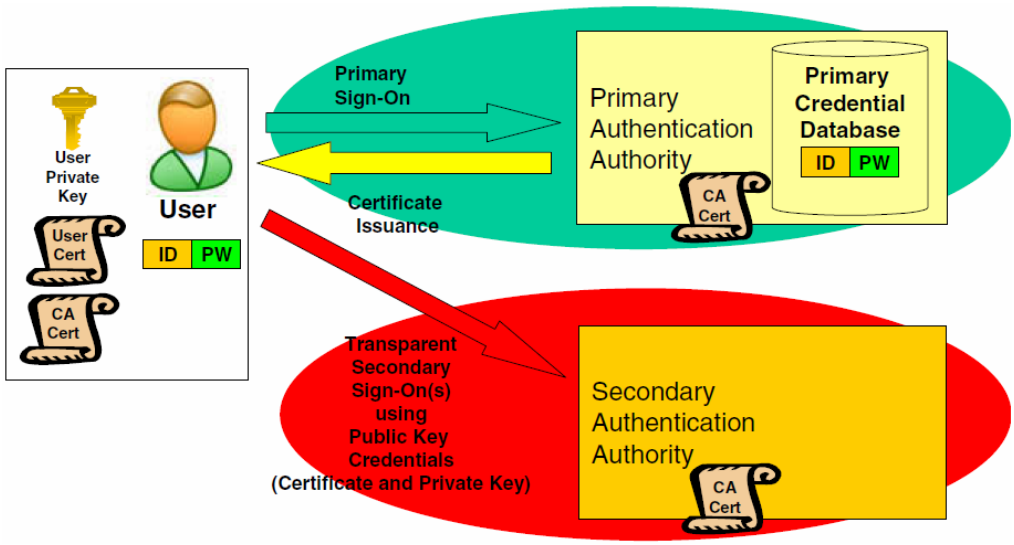


Figure 2.5. Authentication in a public key infrastructure-based SSO architecture (transparent means that the sign-on is executed by the SSO architecture without any user intervention). Figure courtesy of [2]

2.1.3.2. Complex SSO Architectures Handling Different Sets of Credentials

These architectures use multiple sets of credentials and support multiple authentication protocols. This enables them to effectively function in heterogeneous environments. There are three types of these architectures.

1. Credential Synchronization-Based SSO Architecture

Credential synchronization is a mechanism that keeps multiple credentials for each user identical by synchronizing them between databases of different authentication authorities. Credential synchronization-based SSO architectures are not true SSO solutions, because they require a user to provide her credentials on each access attempt to an authentication authority. These architectures are criticized also for being less secure, because if an attacker manages to compromise a database of an authentication authority, she will gain access to all users' credentials for all other authentication authorities (this is called the "Key to the Kingdom" argument). Moreover, synchronizing credentials between databases of different authentication authorities employing different password policies requires an administrator to find a common divisor of all policies. This usually coerces the administrator into mitigating the strictness of those policies. Courion PasswordCourier [3] is an example of this type of architecture. Authentication in a credential synchronization-based SSO architecture is illustrated in Figure 2.6.

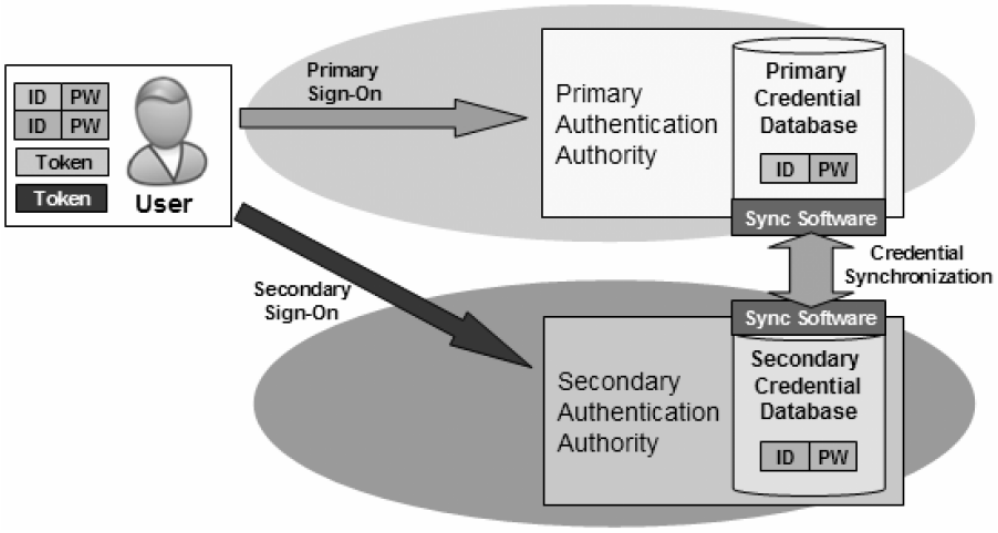


Figure 2.6. Authentication in a credential synchronization-based SSO architecture. Figure courtesy of [2]

2. Secure Client-Side Credential Caching-Based SSO Architecture

In this architecture, a user uses her primary credentials to unlock her credential cache stored on her machine. Later on, she uses credentials stored in that cache to authenticate to one or more secondary authentication authorities. Since the cache must be secured, the use of this architecture is not recommended on operating systems with poor security measures. The Credential Manager in Microsoft Windows XP Professional is an example of this type of architecture. Authentication in a secure client-side credential caching-based SSO architecture is illustrated in Figure 2.7.

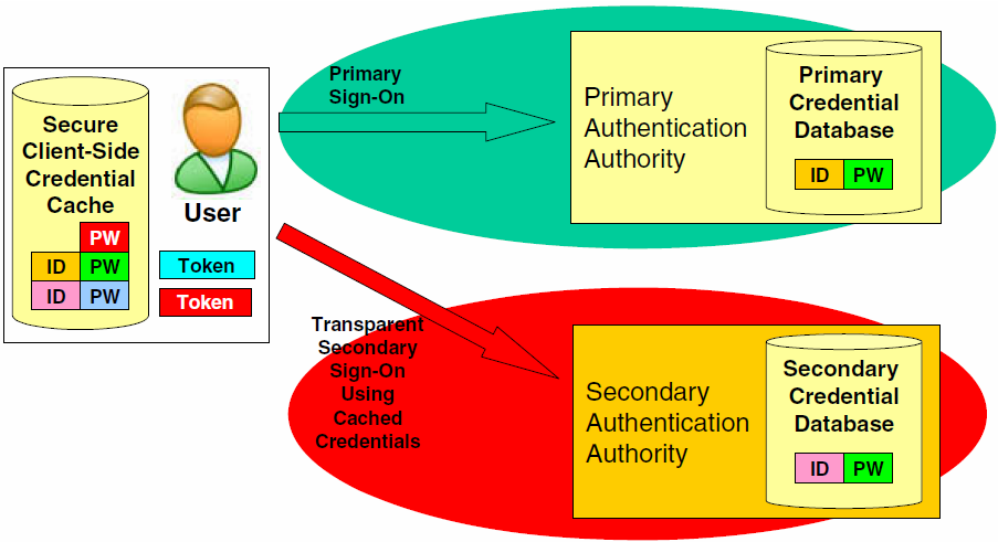


Figure 2.7. Authentication in a secure client-side credential caching-based SSO architecture (transparent means that the sign-on is executed by the SSO architecture without any user intervention). Figure courtesy of [2]

3. Secure Server-Side Credential Caching-Based SSO Architecture

In this architecture, credentials are stored in a cache on the sever, not on the client. A user uses her primary credentials to authenticate to the primary authentication authority which searches the cache for the credentials the user needs to authenticate to one or more secondary authentication authorities, and then transfers these credentials to the user in a secure wallet. Caching on the server side is more secure than caching on the client side, since a server machine is normally equipped with stronger security measures than a client machine. Moreover, the credentials are temporarily downloaded to the client machine and deleted when the sign-on session ends. A difficulty with this type of architecture is maintaining synchronization between credentials in the database of the primary authentication authority and the databases of secondary authentication authorities. Most SSO solutions use password synchronization for this purpose. Citrix Password Manager [4] is an example of this type of architecture. Authentication in a secure server-side credential caching-based SSO architecture is illustrated in Figure 2.8.

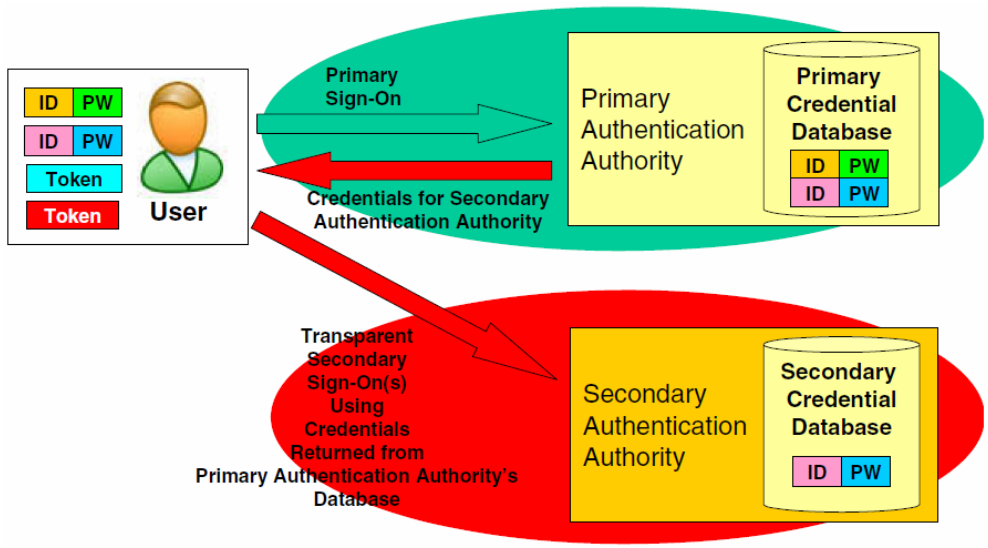


Figure 2.8. Authentication in a secure server-side credential caching-based SSO architecture (transparent means that the sign-on is executed by the SSO architecture without any user intervention). Figure courtesy of [2]

2.1.4. Summary of the Taxonomy

De Clercq summarized the advantages and disadvantages of the architectures included in his taxonomy in a table similar to Table 2.1.

Table 2.1. Summary of De Clercq's taxonomy

Architecture	Advantages	Disadvantages
Token-Based SSO	• A single set of credentials is easy to use and manage. • The software necessary are usually bundled with the OS.	• Can only deal with a single set of credentials. • Work only in a homogeneous environments. • Relies on symmetric cryptography.
PKI-Based SSO	• A single set of credentials is easy to use and manage. • The software necessary are usually bundled with the OS. • Relies on asymmetric cryptography.	• Can only deal with a single set of credentials. • Work only in a homogeneous environments. • Validating certificates requires a lot of processing on the client side.
Credential Synchronization-Based SSO	• Can deal with different sets of credentials. • Work in a heterogeneous environments.	• Different sets of credentials are hard to use and manage. • "Key to the kingdom" argument. • Not a true SSO solution. • Requires an administrator to find a common divisor of all password policies employed in all authentication authorities. • Requires additional software on the server side.
Secure Client-Side Credential Caching-Based SSO	• Can deal with different sets of credentials. • Work in a heterogeneous environments.	• Different sets of credentials are hard to use and manage. • Since the cache must be secured, the use of this architecture is not recommended on portable devices, such as PDAs.
Secure Server-Side Credential Caching-Based SSO	• Can deal with different sets of credentials. • Work only in a heterogeneous environment. • Caching on the server side is more secure than caching on the client side.	• Different sets of credentials are hard to use and manage. • Requires a credential synchronization mechanism. • Requires additional software on the server side.

2.2. Pashalidis-Mitchell's Taxonomy of SSO

Pashalidis and Mitchell [5] suggested another taxonomy of SSO solutions. Pashalidis-Mitchell's taxonomy is based on realizing two SSO devices that differ in functionality. The first device is a pseudo-SSO component which executes different authentication mechanisms on the behalf of the user, and the second device is an Authentication Service Provider (ASP) which has established relationships of trust with all service providers. The number of authentication operations involving a user, and the type of the relationship between user identities and service providers differ between these two devices. Pashalidis and Mitchell relied on the location of the two proposed devices to introduce a deeper classification into their taxonomy. Pashalidis-Mitchell's taxonomy is described in details in the following sections.

2.2.1. Pseudo-SSO Systems

In these systems, a user first performs a primary authentication through which she authenticates to a pseudo-SSO component. In turn, the pseudo-SSO component automatically executes different authentication mechanisms to different service providers (SPs) on the behalf of the user. Note in these systems that a separate authentication operation takes place whenever a user tries to access an SP. A user can typically have multiple identities for a single SP; however, any of those identities can be used with exactly one SP. This means that the relationship between SSO identities and SPs in pseudo-SSO systems is of type n:1. Authentication in a pseudo-SSO system is illustrated in Figure 2.9.

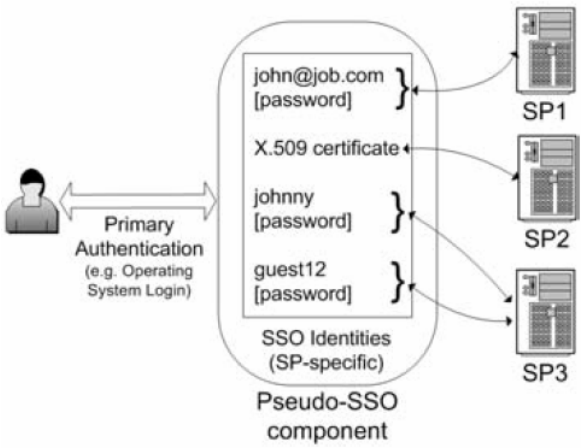


Figure 2.9. Authentication in a pseudo-SSO system. Figure courtesy of [5]

2.2.2. True SSO Systems

In these systems, a user first authenticates to an Authentication Service Provider (ASP) which has established relationships of trust with all SPs. Note in these systems that the only authentication operation involving the user is the one with the ASP. In later access attempts by the user, the ASP sends an authentication assertion to the SP. This assertion contains the user's identity and her authentication status with the ASP. The SP grants/denies access to the user by accepting/rejecting the assertion. Like in pseudo-SSO systems, a user can typically have multiple identities for a single SP; however and in contrast with pseudo-SSO systems, any of those identities can be used with multiple SPs. This means that the relationship between SSO identities and SPs in true SSO systems is of type n:m. Authentication in a true SSO system is illustrated in Figure 2.10.

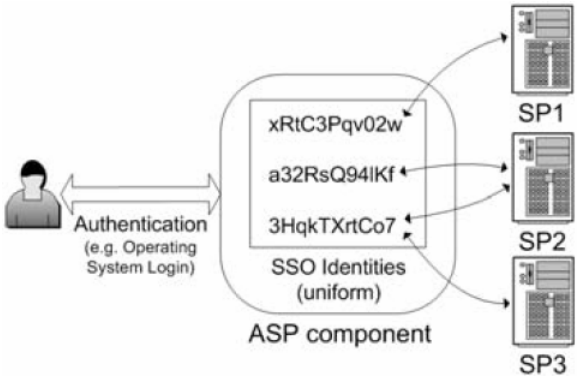


Figure 2.10. Authentication in a true SSO system. Figure courtesy of [5]

2.2.3. Information Flow in an SSO System

Information flow in an SSO system is illustrated in Figure 2.11. In step 1, the user authenticates to the ASP/pseudo-SSO component. Whenever a service is requested by the user in step 2, the ASP/pseudo-SSO component automatically authenticates the user to the SP in step 3. The mechanism of automatic authentication differs between pseudo- and true SSO systems as described above. If automatic authentication succeeds, the service is provisioned to the user in step 4.

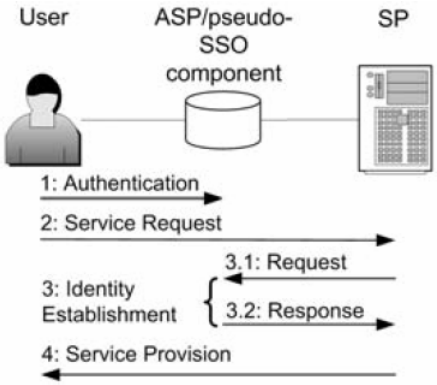


Figure 2.11. Information flow in an SSO system

2.2.4. A Deeper Classification

Based on the location of the ASP/pseudo-SSO component, Pashalidis-Mitchell's taxonomy can be broken down into four types.

1. Local Pseudo-SSO Systems

In these systems, the pseudo-SSO component is installed on the user machine. User authentication credentials for different SPs are stored (usually encrypted) inside this component. Note that the user machine will need a plaintext copy of user's credentials before encrypting them and placing them inside the pseudo-SSO component. Therefore, a user should trust her machine.

2. Proxy-Based Pseudo-SSO Systems

In these systems, the pseudo-SSO component is installed on an external proxy server. Likewise local pseudo-SSO systems, the proxy server will need a plaintext copy of user's credentials and should therefore be trusted by the user. Note that the user machine does not have access to user authentication credentials since authentication operations are done between the proxy server and the SPs.

3. Local True SSO Systems
In these systems, a trusted component inside the user machine plays the role of the ASP. Since the ASP is under user control, integrity of the ASP must be ensured through appropriate mechanisms.

4. Proxy-Based True SSO Systems
In these systems, an external proxy server plays the role of the ASP. Clearly, the ASP can impersonate any registered user at any SP. Therefore, users and SPs should have enough trust in the ASP.

2.2.5. Summary of the Taxonomy
Pashalidis and Mitchell summarized the properties of SSO systems included in their taxonomy in a table similar to Table 2.2.

Table 2.2. Summary of Pashalidis-Mitchell's taxonomy

	Local Pseudo-SSO	Proxy-Based Pseudo-SSO	Local True SSO	Proxy-Based True SSO
Pseudonymity and unlinkability of SSO identities	Cannot be guaranteed	Cannot be guaranteed	Can be guaranteed	Can be guaranteed
Anonymous network access	Needs additional services	Can be integrated	Needs additional services	Can be integrated
Support for user mobility	Needs additional services	Under suitable authentication methods	Needs additional services	Under suitable authentication methods
Use in untrusted environments	Not supported	Under suitable authentication methods	Not supported	Under suitable authentication methods
Deployment cost	Low	Low	High	High
Maintenance cost	Potentially high	Potentially high	Low	Low
Running cost	Low	High	Low	High
Trust relationships	Dynamically changing	Dynamically changing	Consistent	Consistent

2.3. Mapping Between the Two Taxonomies
Careful observation reveals that the complex SSO architectures De Clercq described in his taxonomy directly maps to the SSO systems Pashalidis and Mitchell described in their taxonomy. This mapping is illustrated in Figure 2.12.

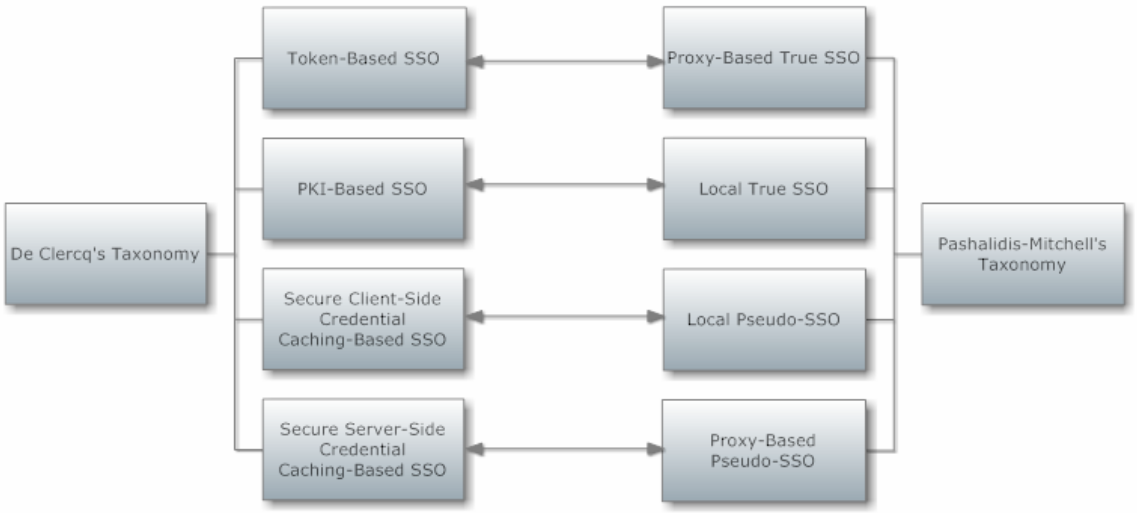


Figure 2.12. The mapping between De Clercq's and Pashalidis-Mitchell's taxonomies

2.4. Double SSO in the Two Taxonomies

In De Clercq's taxonomy, Double SSO is a secure server-side caching-based SSO architecture and in Pashalidis-Mitchell's taxonomy, Double SSO is a proxy-based pseudo-SSO system.

Chapter 3

A Selection of Single Sign-On Solutions

In this chapter, we describe how SSO solutions developed from accessories into crucial parts of any IT security system. Then, we present an interesting taxonomy of qualities wanted in SSO solutions. Afterwards, we list SSO categories and introduce example SSO solutions from each category discussing their architectures and operations in details. The solutions we have chosen are not the only ones available in the market today; there are other solutions that might be more powerful. However, our purpose here is to give the reader an insight into the design of such solutions before starting the discussion of our own SSO solution in Chapters 4 and 5.

3.1. The Evolution of SSO Solutions

At the beginning [6], SSO was a feature added to the overall IT security system, and was regarded as a user convenient access control method, the importance and value of such was a subject of debate. Figure 3.1 shows how SSO was incorporated into the traditional IT security system.

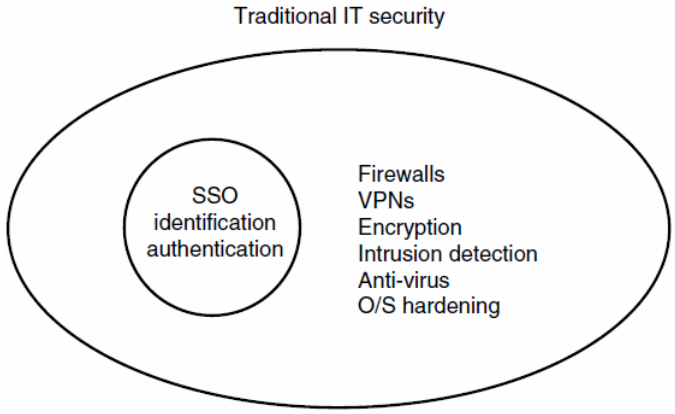


Figure 3.1. SSO at the beginning. Figure courtesy of [6]

SSO image soon improved, and it is now regarded as an important and valuable security product in the overall IT security system. Many factors have contributed to the upturn in the SSO concept, of which we mention:

1. The number of access points in an organization is in continuous growth, therefore there is a need to improve on security controls.
2. There is a need to minimize helpdesk calls in organizations with large number of workstations.
3. The number of workstations an employee has to pass through before reaching a service is increasing.

Figure 3.2 shows how SSO currently fits within the over all IT security system.

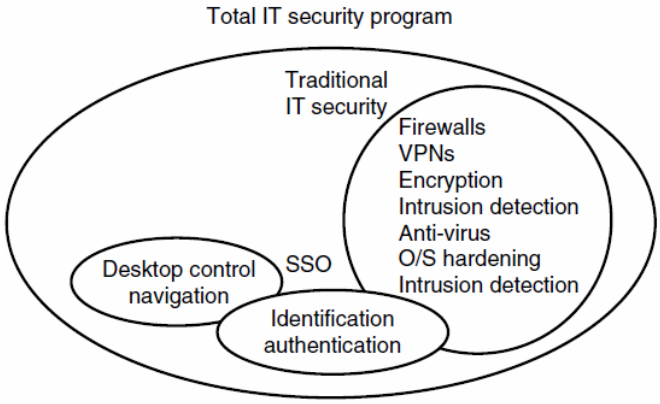


Figure 3.2. SSO at the moment. Figure courtesy of [6]

3.2. SSO Mechanisms

Many mechanisms for implementing SSO have been suggested over time [6]. We try to summarize them below at the very high level:

1. A mechanism that uses Kerberos authentication protocol that is based on a ticketing system to authenticate users and grant them access to resources.
2. The shell-and-script mechanism where the user first authenticates to the shell, and then the shell executes other scripts required to gain access to different systems. Implementing this mechanism is always easier than implementing the first one; however, it has the drawback of requiring a set up of user accounts on each system.
3. A mechanism that requires a user to carry out one sign-on, and then she is connected to all resources until sign-out. One drawback of this mechanism is resource exhaustion and degradation in system performance, because over the SSO operation, many directory accesses are performed and many sessions are opened. Another drawback is establishing unnecessary sessions in which a user might not be active.
4. A mechanism that requires a user to sign on to the network service only, while access to different systems is initiated only upon user request. This mechanism reduces resource exhaustion and does not open unnecessary sessions.

3.3. Quality Checklist for SSO Solutions

In Table 3.1, we list the general characteristics for determining the quality of SSO solutions [6]. For ease of retrieval, we organize the qualities into ten categories; portability, service access, credentials management, user's desktop control, roles and policies management, auditing, application handling, security and access control, security database, and administration and customization.

Table 3.1. Quality checklist for SSO solutions

Category	Quality
Portability	Manage access to the largest number possible of applications without requiring a change in the application. Run on the largest number possible of workstations. Run on the largest number possible of operating systems. Support a single set of control commands across all supported workstations and operating systems. The authorization server is portable to any operating system.
Service access	Support access by dial-up users. Support Internet access. Support remote users independently of their location.

Credentials management	Enforce the least number possible of IDs and passwords for the applications used. Support the largest number possible of physical identification technologies. Support automatic capture of passwords on their first usage by a user. Support revocation of IDs and passwords. Prohibit two IDs from being the same.
User's desktop control	Support the replacement and control of user's desktop with an SSO-managed desktop. Support automatic desktop lock when there is no activity for a period of time.
Roles and policies management	Support administering roles and assigning role-based privileges to users. Support grouping of users. Can enforce common rules on passwords e.g., password length and aging.
Auditing	Support recording individual application usage. Support application monitoring. Record all sign-on attempts.
Application handling	Support application positioning, which allows the customization of the start up script responsible for launching the application. Support application fusing, which allows application to run alongside in a way that makes the user feels that she is working with a single session. Support application controls, which limit user access to only specific parts of the application.
Security and access control	Support session encryption to protect the information traveling between the server and the user. Encryption is based on an acceptable standard. Support user authentication, whether this authentication is done at the server or at the workstation. Authentication is based on an acceptable standard. Reject future sign-on attempts after failure in a specific number of previous attempts. Support tracing access to a system. Protect all enterprise resources. Prohibit the storage of passwords in plaintext. Prohibit the transfer of passwords in plaintext across the network.
Security database	Support synchronizing security data across all systems. Support interfacing with a single security database or with several distributed security databases. Support generating SQL reports of security nature. Support backing up and restoring of the security database.
Administration and customization	Support the administration of endpoints. Normally, there are two methods for administering an endpoint: • API-based Agents: They update an endpoint system that does not support a specific API. • Session Animation Agents: They update an endpoint system that supports an API. Have a single point of administration. Support customization, applying patches and changes in real-time. One solution, not a collection of related solutions. Backward compatible. Include a test system to enable administrators to test security changes before

	applying them. Provide the administrator with a GUI. Provide a single point of authorization. Allow the administrator to customize error message in accordance with the enterprise. Allow the administrator to transfer some of her system privileges to others.

3.4. SSO Categories

There are three categories of SSO solutions:

1. Web SSO: These solutions are for users who access applications using a web interface.
2. Enterprise SSO: These solutions are much broader than web SSO in that they provide SSO to almost all kinds of applications, not only to web-enabled applications.
3. Network SSO: These solutions are for users who access applications in a corporate network domain either through a LAN, or wirelessly, or through a VPN connection.

In the following sections, we present example SSO solutions from each category.

3.5. Example Web SSO Solutions

3.5.1. Google SSO Solution

Google [7] offers a web SSO solution for Google Apps based on the Security Assertion Markup Language (SAML). In this solution, Google plays the role of a service provider by providing services such as Gmail or Google Calendar, while other enterprises play the role of identity providers by controlling user names and passwords needed to authenticate and authorize users to Google web applications. Figure 3.3 illustrates how a user can log in to a Google Apps application such as Gmail or Google Calendar through a SAML-based SSO service hosted by a partner enterprise.

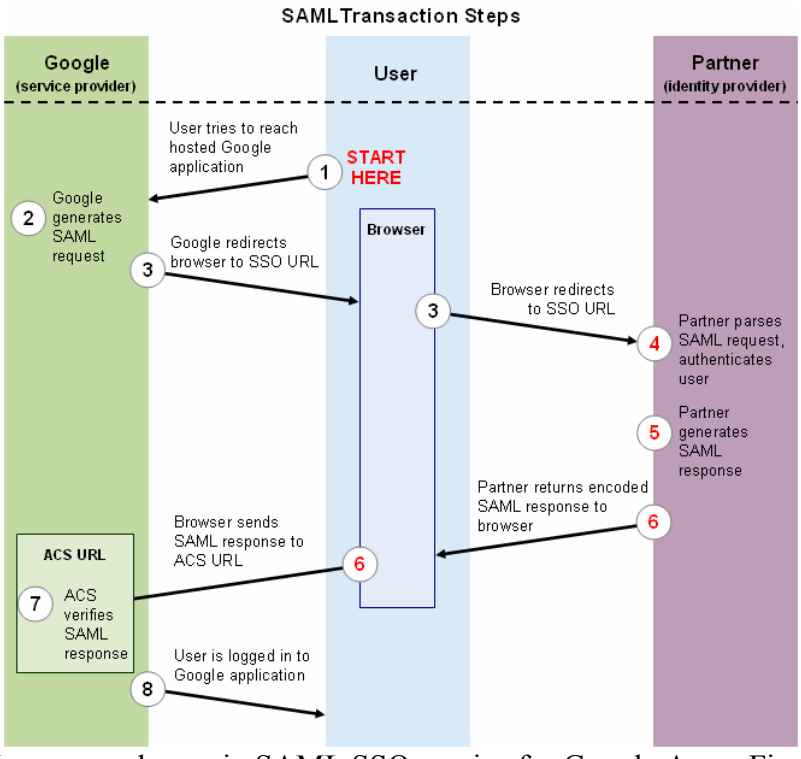


Figure 3.3. Message exchange in SAML SSO service for Google Apps. Figure courtesy of [7]

The following steps take place:
1. The user tries to access a Google Apps application.
2. Google generates a SAML authentication request, encodes it and includes it in the URL of the partner's SSO service.
3. Google sends a redirect to the user's browser which redirects the user to the SSO URL.
4. The partner's SSO service decodes Google's SAML authentication request and authenticates the user.
5. The partner's SSO service generates a SAML response that contains the authenticated user name. This response is signed with the partner's DSA private key.
6. The partner's SSO service encodes the SAML response and sends it to the user's browser which submits it to Google Assertion Consumer Service (ACS).
7. Google ACS verifies the signature on the SAML response using the partner's public key (which should be available). If the verification succeeds, Google redirects the user's browser to the required URL.
8. User is now logged in to the Google App application.

A nice demonstration of SAML SSO service for Google Apps can be found at [8].

3.5.2. Windows Live ID (Formerly Microsoft Passport)

Windows Live ID service [9] is an Internet-based software service that delivers rich experience to individuals and organizations. It can be seen as a federation identity provider that makes claims to other providers and accepts claims made by them. Windows Live ID is a set of claims that the Windows Live ID service makes. Of these claims, we mention:

- User's email address.
- Type of entity (individual or organization).
- Relationships among subjects, such as parent-child relationship.

Windows Live ID serves as a web SSO. It provides access to a range of Microsoft online services, and can be authenticated using the traditional methods (user name/password, smart cards and RADIUS), or using information cards created with Microsoft Windows CardSpace [10]. Windows Live ID service is also capable of mapping federated IDs into a form compatible with Microsoft using standardized protocols like WS-Trust, WS-Security and WS-Federation. Microsoft has made two software development kits, Windows Live ID Server SDK and Windows Live ID Client SDK, publicly available for developers who would like to program for the Windows Live ID service. Another feature related to Windows Live ID introduced by Microsoft is the Windows Live Delegated Authentication [11] that enables Windows Live ID users to permit selected web sites to access their personal information, but with a very precise control over this access and for a period of time specified by the user. The sign in page of Windows Live ID is shown in Figure 3.4.

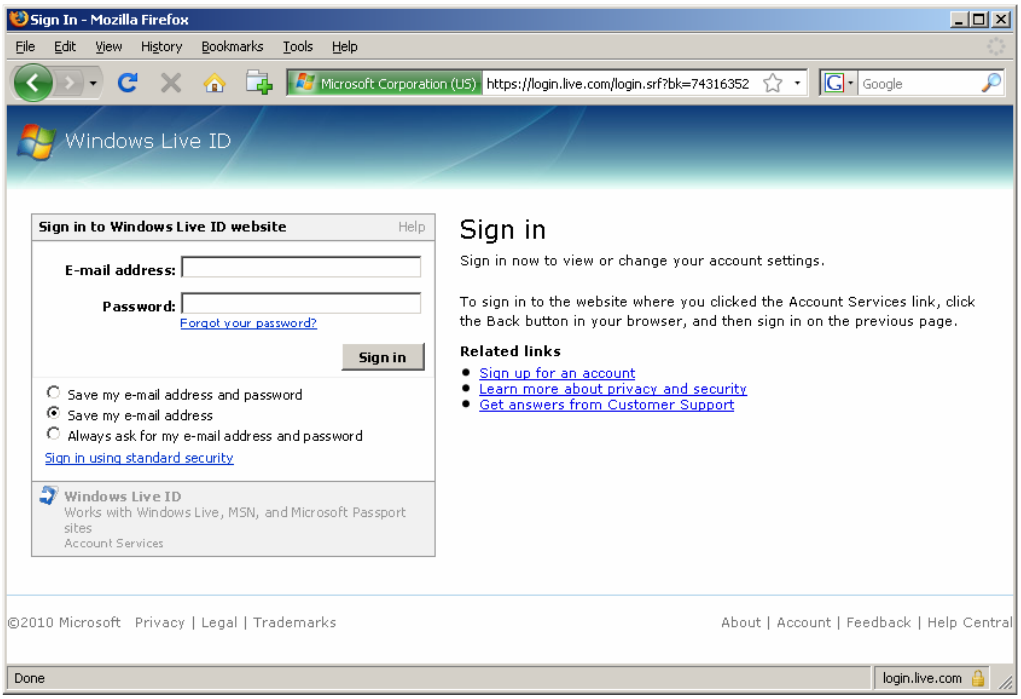


Figure 3.4. The sign in page of Windows Live ID

3.5.2.1. Architecture of Windows Live ID

Figure 3.5 shows the architecture of the Windows Live ID in the world of Microsoft online services.

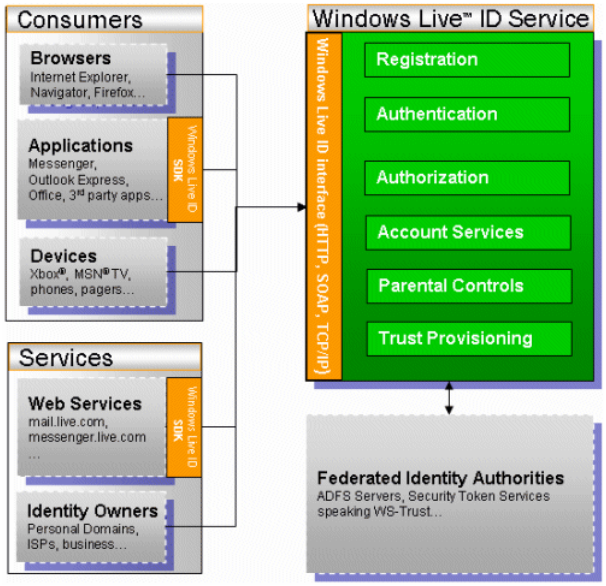


Figure 3.5. Architecture of Windows Live ID in the world of Microsoft online services. Figure courtesy of [9]

3.5.2.2. Operation of Windows Live ID

In Windows Live ID [12], three entities interact with each other; the user, the service provider and the Microsoft passport server. The service provider registers with the passport server and provides it with information about the services offered. The passport server, in turn, assigns a unique ID to the service provider and transmits this ID along with a secret key to the service provider. This key is used for later encrypted communications between the two. The user has to register with the passport server too. In this registration, the user supplies the passport

server with a valid email address and a password. Afterwards, a confirmation email is sent to the user to finalize the registration process. Subsequently, the passport server assigns a Passport Unique Identifier (PUID) to the user. This PUID is never revealed to the user and the purpose of it is to guarantee pseudonymity and unlinkability of SSO identities. The SSO protocol in Windows Live ID is illustrated in Figure 3.6.

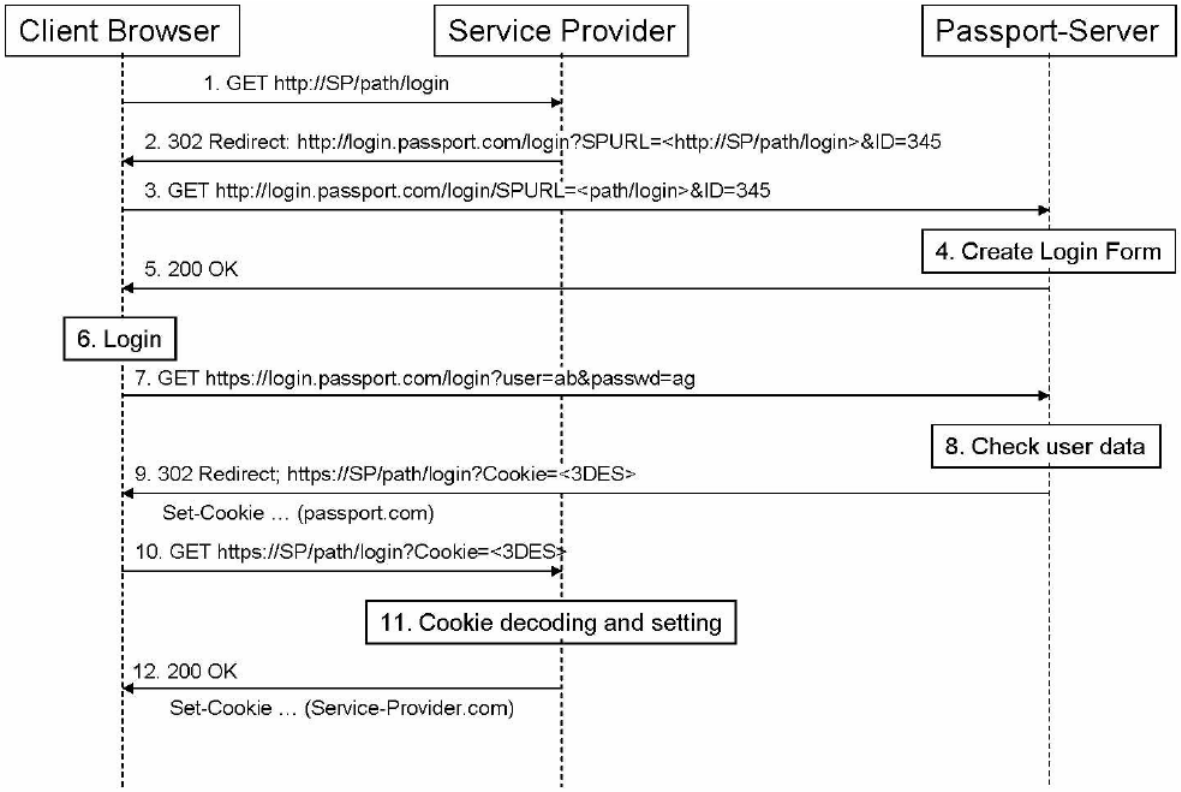


Figure 3.6. The SSO protocol in Windows Live ID. Figure courtesy of [12]

The following steps take place:
1) The user initiates a login on a webpage on the service provider.
2) The service provider redirects the user to the passport server.
3-8) The user performs the actual login through the login form created on the passport server. The user data are transmitted through a secure SSL connection.
9) If login to the passport server is successful, the server redirects the user to the service provider and saves the PUID of the user in a cookie on the user machine encrypted under 3DES. The presence of this cookie on the user machine affirms the authenticity of the user's identity to the service provider.
10-12) The user is granted access to the service on the service provider.
In later attempts by the user to access passport-enables service providers, the stored cookie is sent to the passport server that checks its validity. If the cookie is valid, the flow continues with step 10 above, thus achieving an SSO experience.

3.5.3. Microsoft Office SharePoint Server (MOSS)

Microsoft Office SharePoint Server (MOSS) includes a web SSO solution that allows SharePoint-based web applications to retrieve information from back-end applications that require credentials different from those used to authenticate to the SharePoint Server [2].

3.5.3.1. Architecture of MOSS SSO

MOSS SSO is an example of a secure server-side credential caching-based SSO architecture (see Chapter 2) that stores the credentials encrypted in a Microsoft SQL Server database. MOSS SSO does not support password synchronization, moreover, configuring it requires some customized coding. MOSS SSO credential mapping can happen in one of two ways:

1. MOSS individual mapping: It is a one-to-one mapping between MOSS and the back-end application; meaning that an MOSS user uses a dedicated back-end account to access the back-end application.
2. MOSS group mapping: It is a many-to-one mapping between MOSS and the back-end application; meaning that all MOSS users use a single back-end account to access the back-end application.

You can configure MOSS SSO using the MOSS web administration interface shown in Figure 3.7.

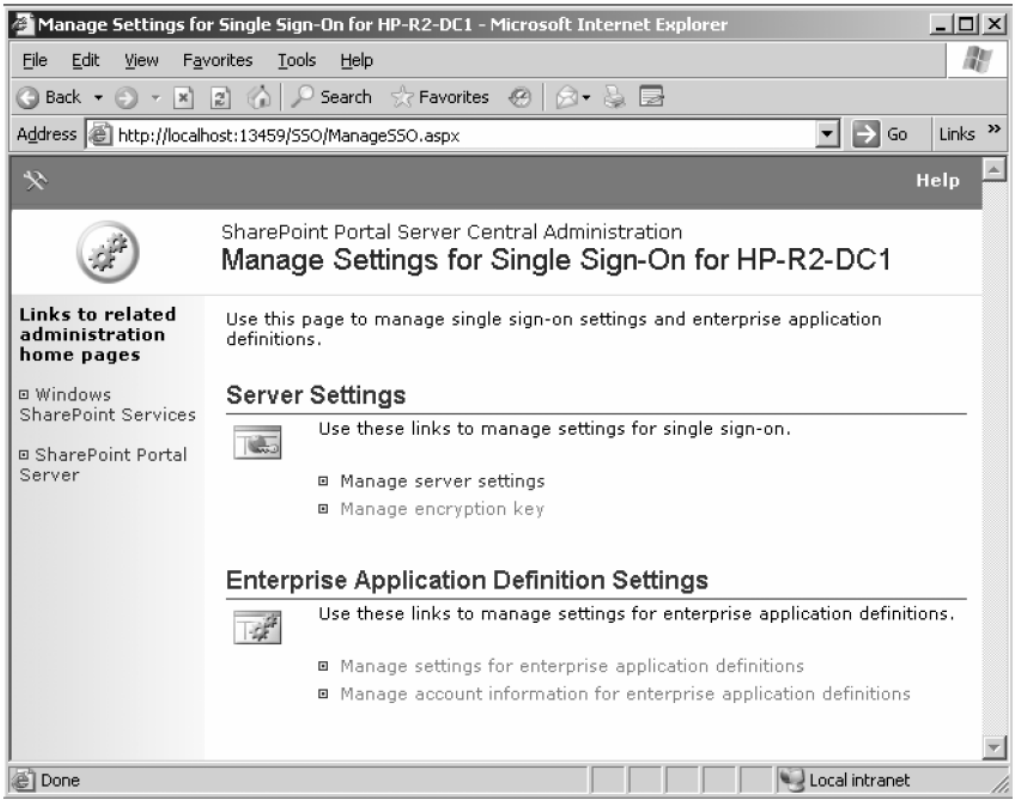


Figure 3.7. The MOSS SSO web administration interface. Figure courtesy of [2]

3.5.3.2. Operation of MOSS SSO

Figure 3.8 shows message exchange in MOSS SSO.

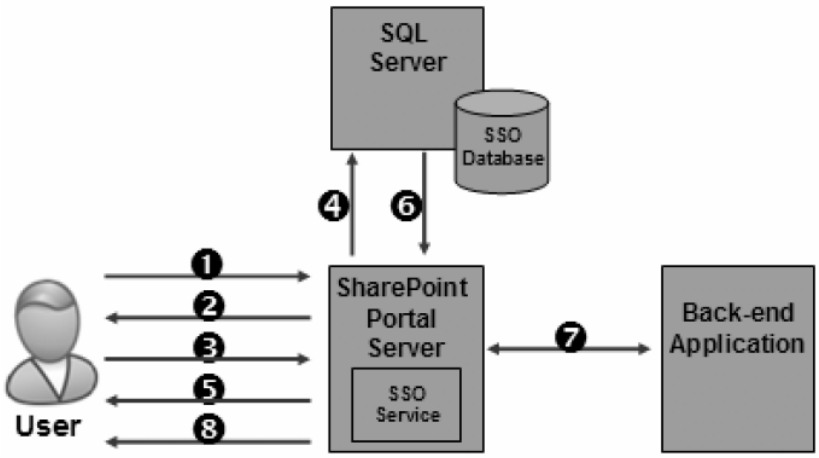


Figure 3.8. Message exchange in MOSS SSO. Figure courtesy of [2]

The user in Figure 3.8 is accessing a SharePoint-based web application that requests data from the back-end application. The following steps take place:

1. The MOSS SSO web logic checks whether the credentials required to access the back-end application are stored in the MOSS SSO database. If they are there, message exchange continues from step 6.
2. If the credentials are not found, the user's browser is redirected to the logon page of the back-end application.
3. The user enters the appropriate credentials in the logon page.
4. MOSS SSO performs a mapping between the credentials entered and the user's Windows account, and stores the mapping in the MOSS SSO database.
5. The user's browser is redirected back to the original web application.
6. The web application retrieves the credentials from the MOSS SSO database.
7. The web application pushes the credentials to the back-end application and retrieves necessary data.
8. Data are displayed in the web application.

3.5.4. Active Directory Federation Service (ADFS)

Active Directory Federation Service (ADFS) was introduced by Microsoft in Microsoft Windows Server 2003 R2 [2]. It is an identity federation and web SSO solution that allows enterprises to extend the scope of their Active Directory databases to other enterprises. The goal of federation is to make it easier for enterprises to share data with authorized external users.

3.5.4.1. Architecture of ADFS

As shown in Figure 3.9, ADFS consists of three major components; the Federation Service (FS), the Web Agent and the Active Directory.

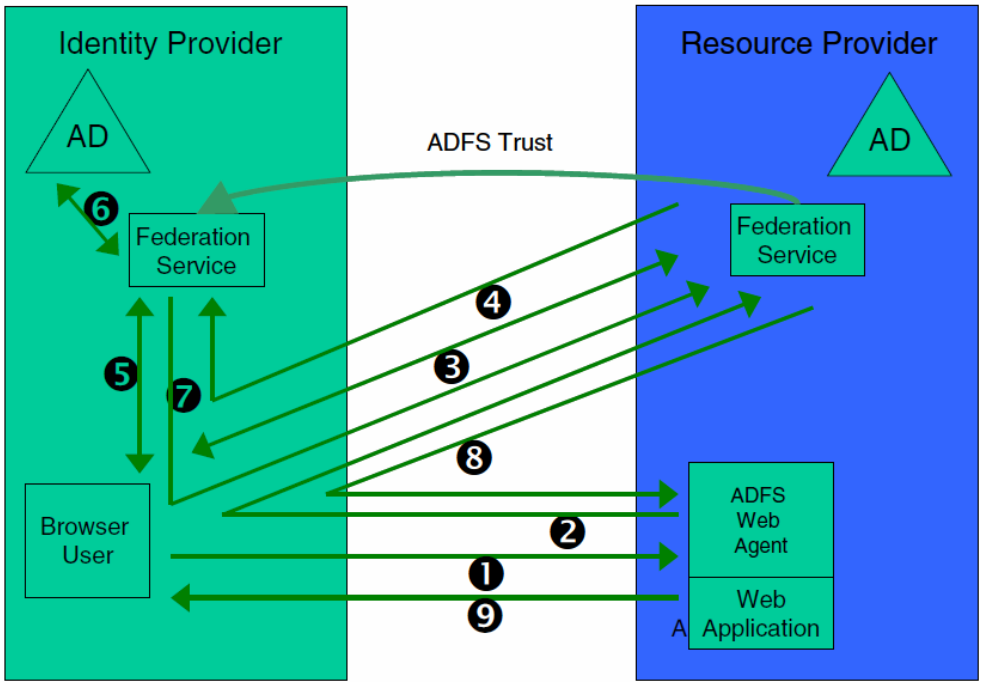


Figure 3.9. Architecture of ADFS. Figure courtesy of [2]

Note in this figure how a user's browser in one enterprise (the identity provider) is granted access to a web application in another enterprise (the resource provider) after ADFS established a one-way trust from the resource provider to the identity provider. Establishing trust is done using X.509 certificates.

3.5.4.2. Operation of ADFS

The message exchange in Figure 3.9 includes the following steps:

1. A user's browser on the identity provider attempts to access a web application on the resource provider.
2. Because the user was not authenticated before, the ADFS Web Agent redirects the user to the resource provider's FS.
3. This step is called "home realm discovery" through which the user provides the resource provider's FS with information about her home domain.
4. Based on the information provided in step 3, the resource provider's FS redirects the user to the correct identity provider's FS.
5. The user authenticates to the identity provider's FS using her Active Directory credentials.
6. The identity provider's FS verifies user's credentials against the Active Directory. If verification succeeds, the identity provider's FS generates an authentication cookie and a security token, it further signs the security token to protect it against tampering.
7. The identity provider's FS redirects the user along with the authentication cookie and the security token to the resource provider's FS.
8. This step is called "claim transformation" through which the resource provider's FS verifies the correctness of the security token signature, and then transforms the security token into a form understandable by the web application. Afterwards, the resource provider's FS generates a new authentication cookie and redirects, the user, the new authentication cookie and the transformed security token to the ADFS Web Agent. The ADFS Web Agent validates the security claims and sends them to the web application.
9. The web application returns the required content to the user.

3.5.5. Liberty SSO Solution

The Liberty Alliance [13] was created to set up a web SSO standard [12] that supports all known operating systems, programming languages and network structures. One advantage of the specifications set forth by Liberty is the coupling of a user's identities without revealing those identities to service providers. Another advantage is the ability to expand a sign-on with one service provider to other service providers thus achieving SSO experience. Yet another advantage is the possibility of single sign-out of all service providers. The SSO protocol in Liberty is illustrated in Figure 3.10.

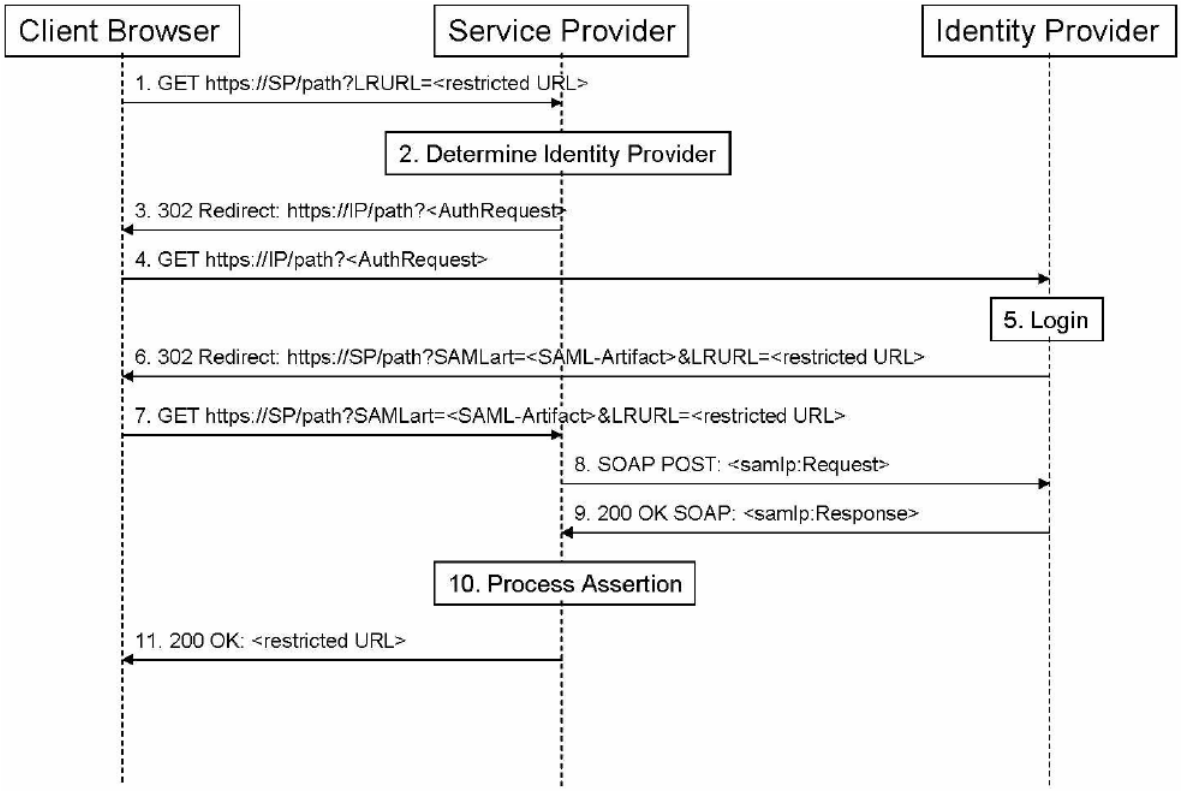


Figure 3.10. The SSO protocol in Liberty. Figure courtesy of [12]

The following steps take place:
1) The user navigates to a service provider that suggests a number of identity providers to the user.
2) The user determines the identity provider to use.
3) The service provider redirects the user to the chosen identity provider. This redirect includes an AuthRequest that controls the behavior of the identity provider.
4-5) If the user is not logged in to the identity provider, she has to login first.
6-7) If login to the identity provider is successful, the identity provider redirects the user to the service provider. This redirect includes a SAML artifact the service provider uses to access information about the user.
8) The service provider decodes this artifact and then asks the identity provider for the user information by sending a SOAP Request.
9) The identity provider sends those information to the service provider in a SOAP Response that includes an assertion about the user's identity.
10-11) The service provider processes the received assertion and grants access to the user.

3.6. Example Enterprise SSO Solutions

3.6.1. Microsoft BizTalk Server and Microsoft Host Integration Server (HIS)

Microsoft BizTalk Server and Microsoft Host Integration Server (HIS) include an enterprise SSO solution called the Enterprise Single Sign-On (ENTSSO) service [2]. ENTSSO functions not only in a Windows environment, but also in other environments (e.g., Linux) and platforms (e.g., SAP). ENTSSO is also capable of password synchronization.

3.6.1.1. Architecture of ENTSSO

ENTSSO is an example of a secure server-side credential caching-based SSO architecture (see Chapter 2). The architecture of ENTSSO is shown in Figure 3.11.

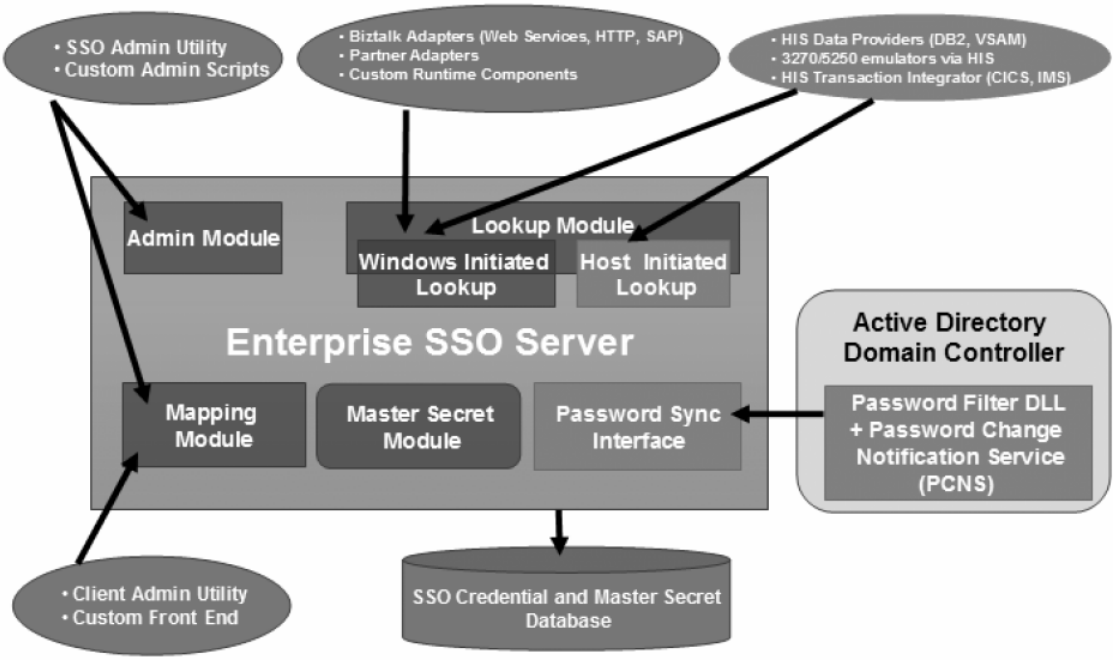


Figure 3.11. Architecture Of ENSSO. Figure courtesy of [2]

ENTSSO includes a credential mapping between Windows user accounts and non-Windows user accounts. Whenever a BizTalk-rooted or an HIS-rooted application needs to access a non-Windows application or platform, ENTSSO looks up in its credential mapping for the necessary credentials. The mapping in ENTSSO can be configured using one of two ways:

1. The ssomanage.exe command line tool on the server side and the ssoclient.exe command line tool on the client side (see Figure 3.12).
2. The ENTSSO server-side and client-side GUI configuration utilities (see Figures 3.13 and 3.14).

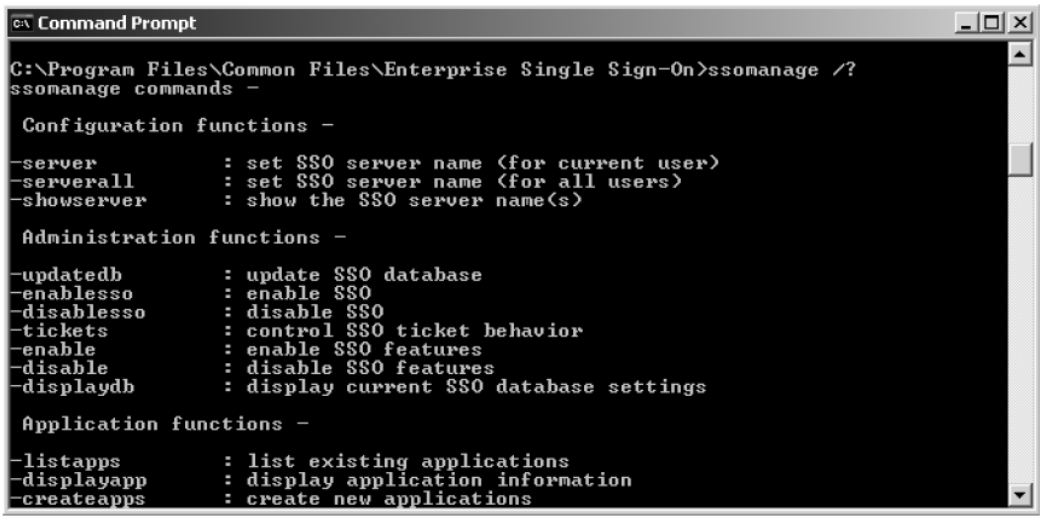


Figure 3.12. ENTSSO command line administration. Figure courtesy of [2]

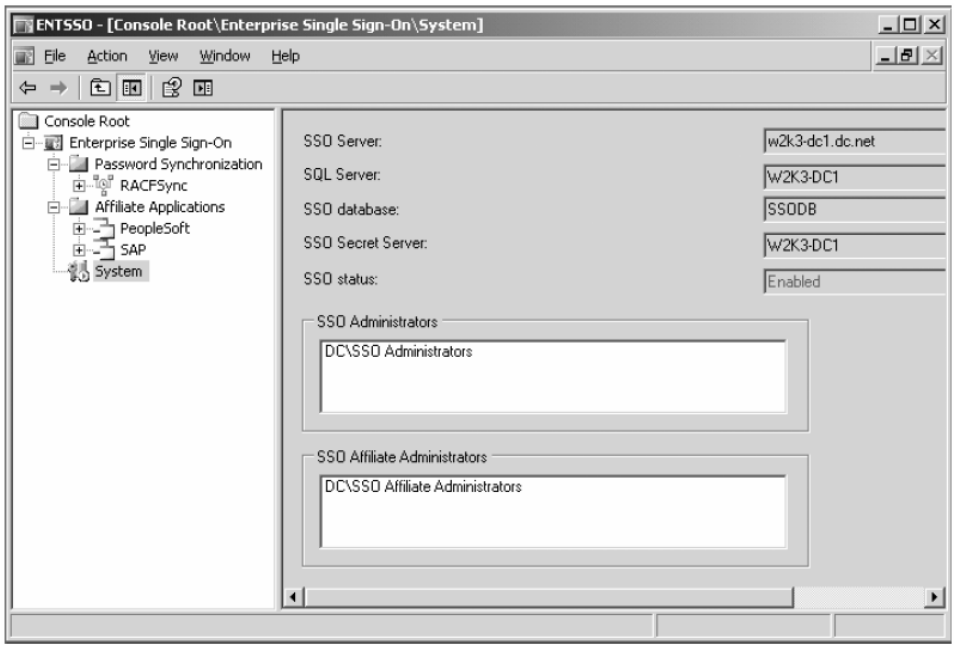


Figure 3.13. ENTSSO server-side GUI administration. Figure courtesy of [2]

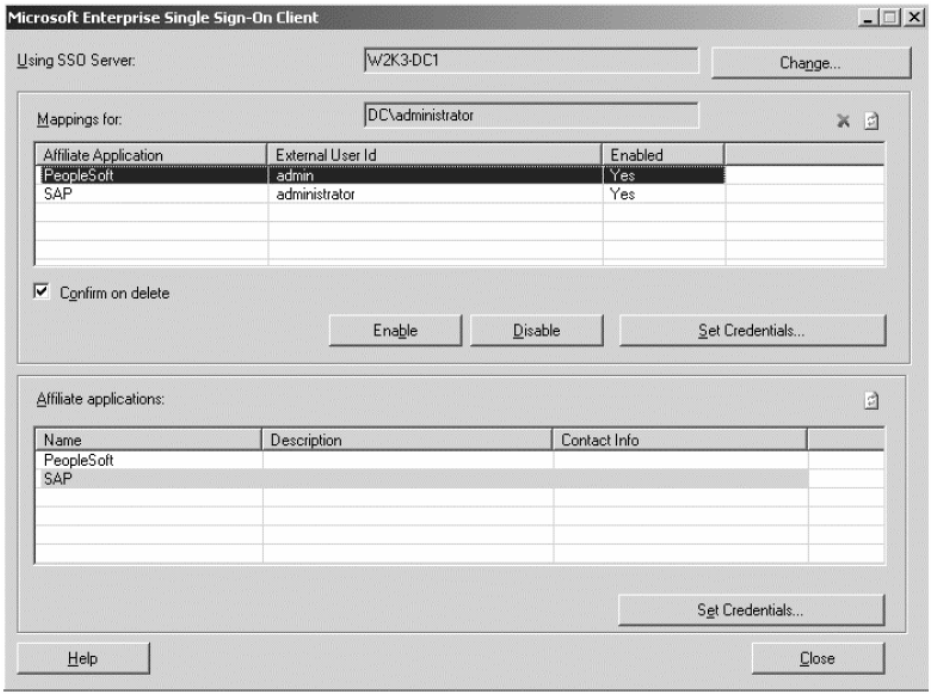


Figure 3.14. ENTSSO client-side GUI administration. Figure courtesy of [2]

Note in Figure 3.11 that when a BizTalk adapter triggers the ENTSOO credential mapping, a Windows initiated lookup takes place, whereas when an HIS data provider triggers the mapping, a Windows initiated or a host initiated look up takes place. The difference between Windows initiated and host initiated lookup is that the former means that a user who is logged on to a Windows environment is using ENTSSO to access a non-Windows environment, whereas the latter means that a user who is logged on to a non-Windows environment is using ENTSSO to access a Windows environment. ENTSSO credential mapping can happen in one of four ways:

1. Windows individual mapping: It is a one-to-one mapping between Windows and non-Windows accounts; meaning that a Windows user uses a dedicated non-Windows account to access the target system.
2. Windows group mapping: It is a many-to-one mapping between Windows and non-Windows accounts; meaning that all Windows users use a single non-Windows account to access the target system.
3. Host individual mapping (only available in HIS): It is a one-to-one mapping between non-Windows and Windows accounts; meaning that a non-Windows user uses a dedicated Windows account to access the target system.
4. Host group mapping (only available in HIS): It is a many-to-one mapping between non-Windows and Windows accounts; meaning that all non-Windows users use a single Windows account to access the target system.

The ENTSSO credential mapping is stored encrypted in a Microsoft SQL Server database named SSODB. Encryption is symmetric and is done using a 128 bit key called the master secret. The master secret is usually stored on a dedicated server. A corporate environment implementing ENTSSO normally consists of:

1. Multiple ENTSSO servers for BizTalk adapters or HIS data providers.
2. One server with Microsoft SQL Server installed to store ENTSSO database.
3. One server to store the master secret.

3.6.1.2. Operation of ENTSSO

Figure 3.15 shows a Windows initiated lookup using ENTSSO (here we assume working with Microsoft BizTalk Server).

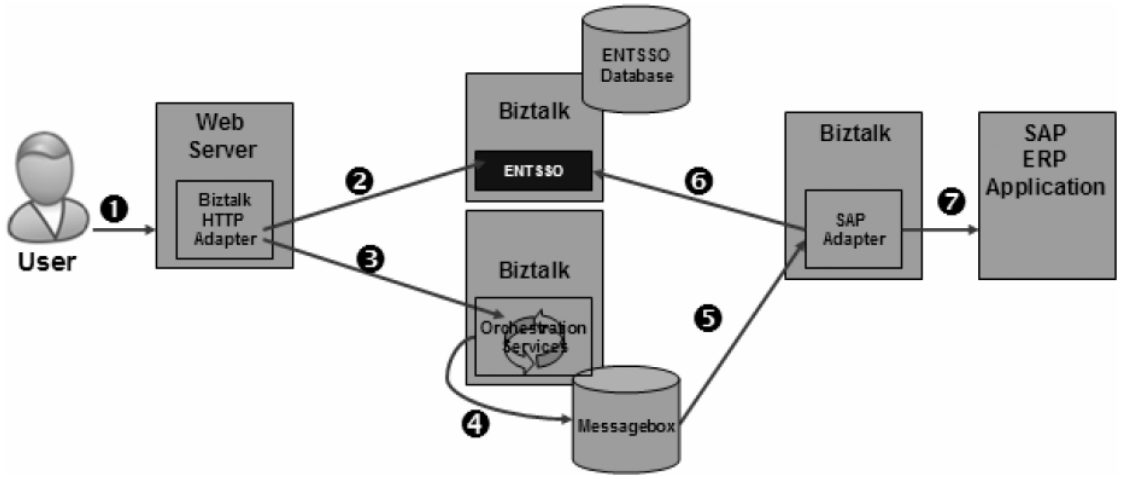


Figure 3.15. Windows initiated lookup using ENTSSO in Microsoft BizTalk Server. Figure courtesy of [2]

The user in Figure 3.15 is logged on to a Windows environment and is running a web application. The goal of this user is to access the data of a SAP ERP application running in a non-Windows environment. The following steps take place:

1. The web application is rooted into a BizTalk HTTP adapter and uses it to drop a message into the BizTalk Message Box.
2. The BizTalk HTTP adapter impersonates the user and requests an ENTSSO ticket from the ENTSSO server, then it drops this ticket into the BizTalk Message Box.
3. BizTalk orchestration services convert the message in the Message Box into a SAP message.
4. The BizTalk SAP adapter retrieves the SAP message and the ENTSSO ticket from the BizTalk Message Box.
5. The BizTalk SAP adapter uses the ENTSSO ticket to request user's SAP credentials from the ENTSSO server.
6. The ENTSSO server retrieves the encrypted user's SAP credentials from the ENTSSO database server, and retrieves the master secret from the master secret server securely through Remote Procedure Call (RPC). Finally, the ENTSSO server decrypts the user's SAP credentials.
7. The ENTSSO server sends the user's SAP credentials to the BizTalk SAP adapter which uses them to access the SAP ERP application.

Figure 3.16 shows a host initiated lookup using ENTSSO (here we assume working with Microsoft HIS).

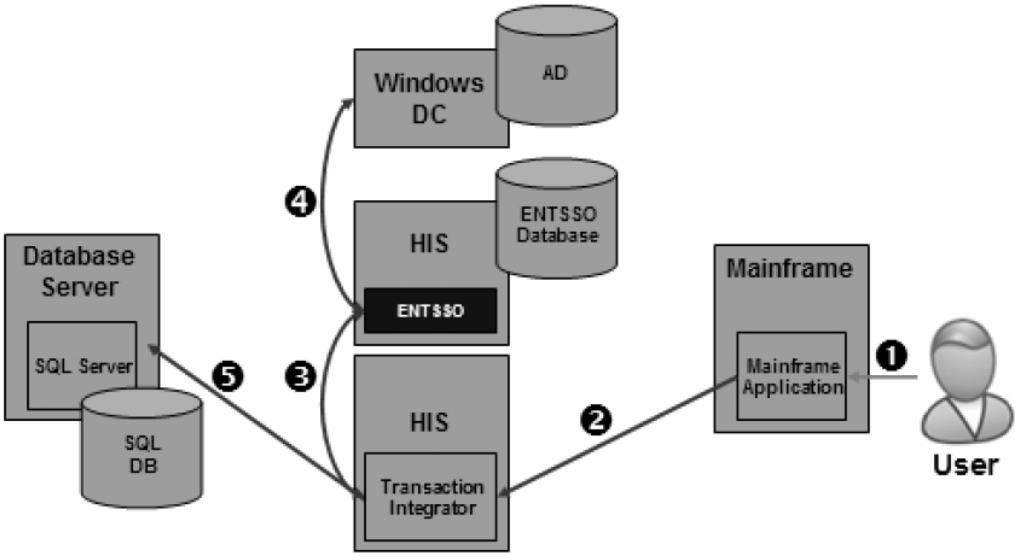


Figure 3.16. Host initiated lookup using ENTSSO in Microsoft HIS. Figure courtesy of [2]

The user in Figure 3.16 is logged on to a non-Windows mainframe and is running a mainframe application. The goal of this user is to access the data in a Microsoft SQL Server database running in a Windows environment. The following steps take place:

1. The mainframe application is rooted into an HIS data provider and uses it to call on the HIS Transaction Integrator.
2. The HIS Transaction Integrator impersonates the user and requests the user's Windows account name from the ENTSSO server.
3. The ENTSSO server requests an access token (Kerberos-based) from Windows domain controller.
4. The ENTSSO server sends the access token to the HIS Transaction Integrator.
5. The HIS Transaction Integrator uses the access token to access the Microsoft SQL Server database.

ENTSSO includes a password synchronization solution called the Password Change Notification Service (PCNS). PCNS must be installed on each domain controller and its configuration data are stored in an Active Directory database. PCNS includes three pieces of software:

1. The password filter dynamic link library: It is responsible for capturing a new password or a changed password from the domain controller.
2. The PCNS service: It is responsible for receiving notification of password changes from the password filter, and forwarding them to the target system.
3. The PCNS configuration utility: It is used to update the PCNS configuration data stored in the Active Directory database.

Using PCNS, password synchronization in ENTSSO can be configured in one of three ways:

1. Windows to non-Windows full synchronization: In this setting, password changes in the Active Directory database are captured and synchronized to non-Windows systems and to the ENTSSO database.
2. Non-Windows to Windows partial synchronization: In this setting, password changes in non-Windows systems are captured and synchronized to the ENTSSO database.
3. Non-Windows to Windows full synchronization: In this setting, password changes in non-Windows systems are captured and synchronized to the ENTSSO database and the Active Directory database.

3.7. Example Network SSO Solutions

3.7.1. Microsoft Internet Authentication Service (IAS)

Microsoft Internet Authentication Service (IAS) is an implementation of the Remote Authentication Dial In User Service (RADIUS) protocol [2]. It is a network SSO solution that comes with Microsoft Windows Server 2003 and later versions. IAS provides Authentication, Authorization and Accounting (AAA) services, and it can handle dial-up users, wireless users and users connecting through a VPN connection as shown in Figure 3.17.

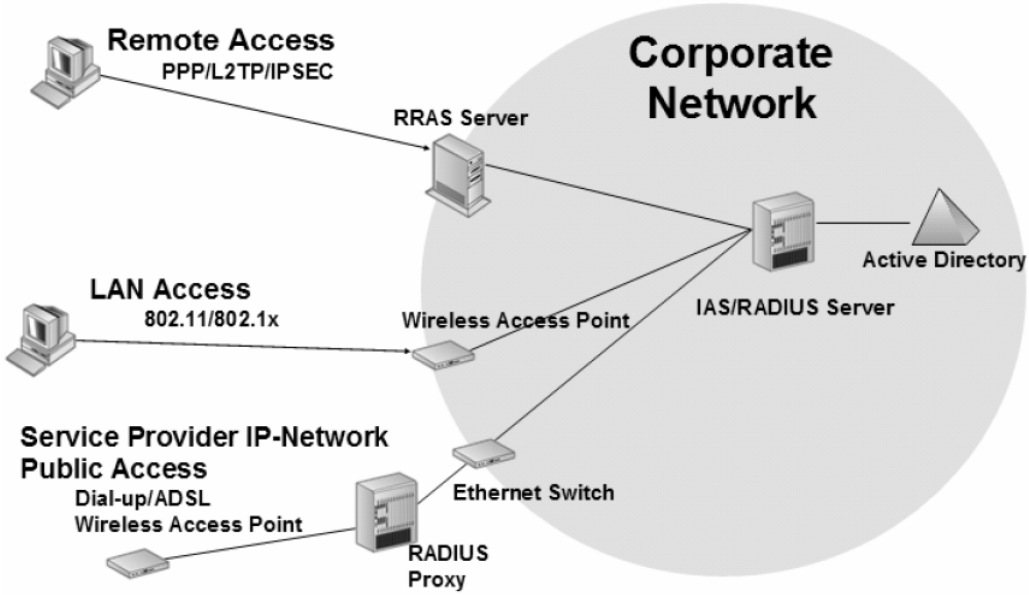


Figure 3.17. IAS network SSO. Figure courtesy of [2]

IAS can be integrated with Active Directory to authenticate users against an Active Directory database. IAS supports different authentication methods of which we mention, Password Authentication Protocol (PAP), Shiva PAP, Challenge Handshake Authentication Protocol (CHAP), Microsoft CHAP (MS-CHAP), Microsoft CHAP v2 (MS-CHAP v2), Extensible Authentication Protocol (EAP) and Protected Extensible Authentication Protocol (PEAP).

3.8. Example Both Web and Enterprise SSO Solutions

3.8.1. The Credential Manager

The purpose of the Credential Manager [2] is to rid users from having to enter the same credentials whenever they access resources on the same server. It serves as both; a web and enterprise SSO, and comes with Microsoft Windows XP Professional, Microsoft Windows XP Home (with a limited functionality) and Microsoft Windows Server 2003. The Credential Manager uses a mechanism that is similar to the secure client-side credential caching-based SSO Architecture (see Chapter 2).

3.8.1.1. Architecture of the Credential Manager

The Credential Manager is composed of three major components:

1. The credential store: As the name implies, the purpose of the credential store is to store user's credentials. Access to this store is protected using Microsoft Data Protection API (DPAPI), and it is unlocked when the user signs on to the machine or to the domain. In

addition to user's credentials, the credential store contains credential-target maps that relate user's credentials to target domains the credentials are meant to access.

2. The key ring: The key ring is used to manage the credential-target maps in the credential store. The "Stored User Names and Passwords" dialog box (see Figure 3.18) provides an interface to deal with this key ring.

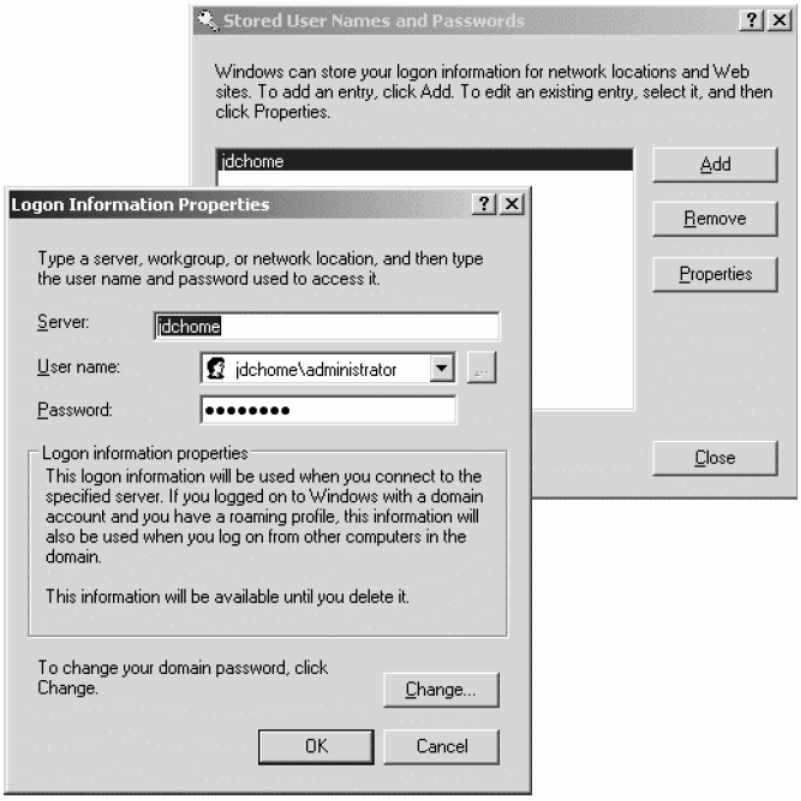


Figure 3.18. The "Stored User Names and Passwords" dialog box used for interfacing with the key ring. Figure courtesy of [2]

3. The credential collection component: When the Credential Managers fails at accessing a target using the primary credentials, its credential collection component presents the user with the "Connect to" dialog box shown in Figure 3.19. If the user decides to remember her password, the Credential Manager caches it in the credential store.

Figure 3.19. The "Connect to" dialog box used for interfacing with the credential collection component. Figure courtesy of [2]

3.8.1.2. Operation of the Credential Manager

To illustrate the operation of the Credential Manager, we will assume that a user named "Bob" working on a PC named "bobws" wants to access a resource named "share" located on a server named "devserv". The following steps take place (see Figure 3.20):

1. Bob uses his primary credentials to log on to her bobws, then she uses a front-end to access the resource \\devserv\share.
2. The front-end asks the Local Security Authority (LSA) to authenticate Bob to bobws.
3. The LSA asks the authentication package in the operating system.
4. The authentication package asks the Credential Manager for suitable credentials to access devserv. Of course, the Credential Manager does not find anything for devserv, therefore it returns Bob's primary credentials to the authentication package.
5. The authentication package tries with Bob's primary credentials but fails to access devserv.
6. The authentication packages passes this failure over to the LSA.
7. The LSA passes the same failure over to the front-end.
8. The front-end reacts by invoking the credential collection component that shows the "Connect to" dialog box to Bob.
9. Bob enters the credentials necessary to access devserv in the "Connect to" dialog box. These credentials are passed over to the front-end, then to the LSA, and finally to the authentication package.
10. The authentication package uses the supplied credentials to authenticate Bob to devserv and give her access to the resource \\devserv\share.

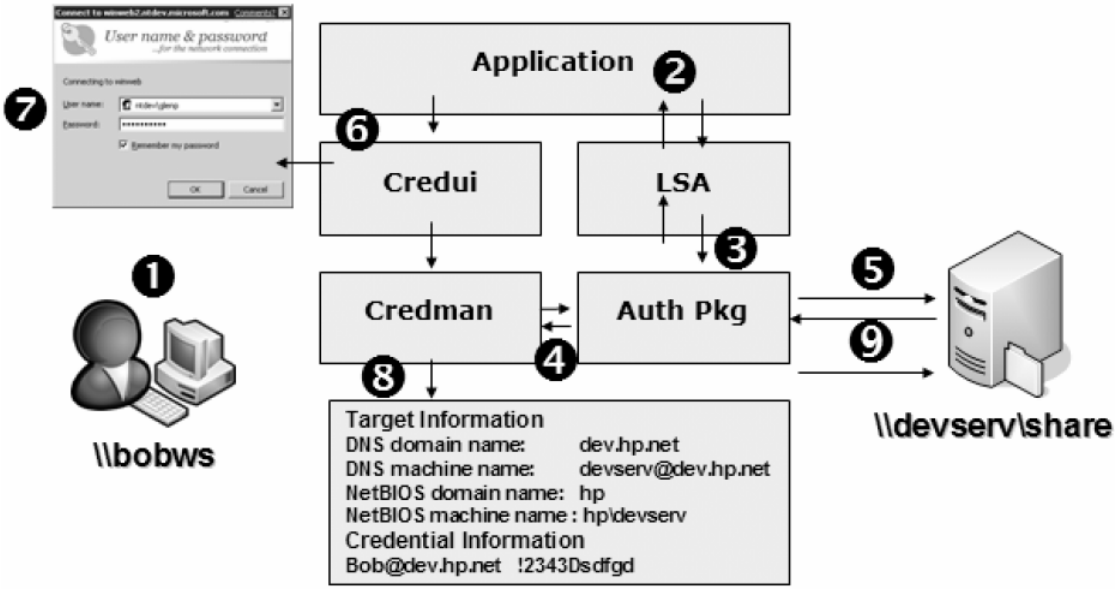


Figure 3.20. Operation of the credential manager. Figure courtesy of [2]

One drawback of the Credential Manager is that password changes on the server are not synchronized to the credential store. In addition, the security of storing credentials in the credential store on the client-side is questionable, although it can be disabled. Microsoft Windows Server 2003 includes a command line tool called cmdkey that allows you to create, delete and list stored credentials in the credential store (see Figure 3.21).

Figure 3.21. cmdkey tool in Microsoft Windows Server 2003. Figure courtesy of [2]

Chapter 4

Double SSO Preliminaries

Any large-scale protocol is a combination of building blocks and is subject to various kinds of attacks. Understanding the technicalities of these blocks and attacks is important to understand the flow, the strength and the weakness of the protocol. It is also useful in comparing the protocol with its counterparts. Double SSO, the central part of this thesis work, is one such large-scale protocol. It provides corporate network users with an efficient web SSO solution that frees them from the burden of maintaining a large number of usernames and passwords for every service with which they are registered. In this chapter, we present the essential building blocks of Double SSO; they are Shamir's Identity-Based Signature Scheme, Zero-Knowledge Interactive Proofs and Simmons' Impersonation-Proof Identity Verification Scheme. After describing each building block, we show how it is utilized in the design of Double SSO. We finally discuss one distinctive attack that can only threaten SSO protocols, of which Double SSO is.

4.1. Shamir's Identity-Based Signature Scheme

Shamir's Identity-Based Signature Scheme [14] is the first scheme of its kind that allows any two parties to sign and verify each other's signature without exchanging their public keys and without relying on a trusted public key directory to keep track of those keys. Shamir's scheme differs from traditional public key signature schemes in one way: Instead of randomly generating a pair of public and private keys for each user, the user uses her identity as a public key and asks a trusted Key Generation Center (KGC) to generate the corresponding private key. Only the trusted KGC can generate the private key since the generation relies on some secret information that is available only to it. The trusted KGC can be shut down after generating the private keys of all users and delivering them on smart cards, for example. User's identity can be anything that uniquely identifies the user, such as any combination of the name, email, social security number, address or phone number provided that this identity is available to other parties with which the user is going to communicate. Alice signs the message using her private key (on the smart card) and sends it to Bob. Bob verifies Alice's signature on the message using her identity. Figures 4.1 and 4.2 illustrate the differences between public key signature schemes and identity-based ones.

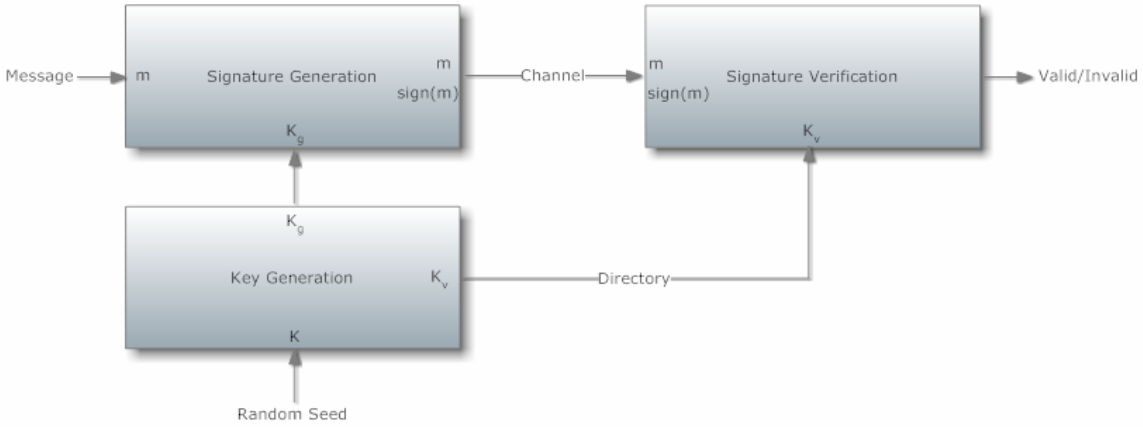


Figure 4.1. Illustration of the operation of public-key signature schemes

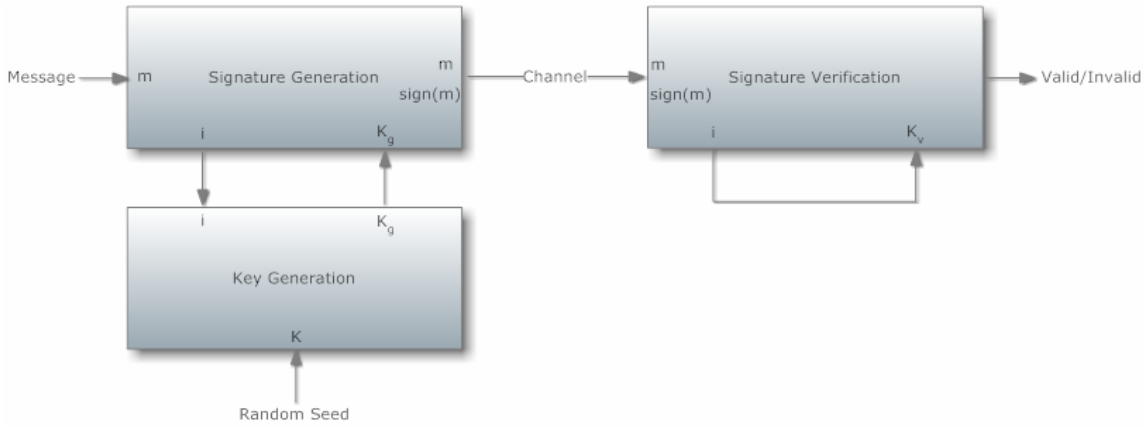


Figure 4.2. Illustration of the operation of identity-based signature schemes

Note in Figure 4.1 that a separate key channel (a directory) that preserves the authenticity of the public key is needed to transfer the signer's public key to the verifier. On the contrary, note in Figure 4.2 that the verification key is derived from the signer's identity which is already known to the verifier, thus the separate key channel between users is completely eliminated.

Now, we describe the implementation of Shamir's scheme. The KGC generates RSA public (e,n) and private (d,n) keys, where $n = pq$, $\varphi(n) = (p-1)(q-1)$ and p and p are secret large prime numbers. The KGC can then generate the private key g corresponding to a user's identity i using:

$$g^e = i \pmod{n}$$

Note that generating this private key requires the extraction of the e-th roots $\bmod n$. If the RSA problem is hard, this extraction can only be done by the KGC.

To sign a message m, the sender chooses a random number r and computes the first half of the signature using:

$$t = r^e \pmod{n}$$

And the second half using:

$$s = g.r^{f(t,m)} \pmod{n}$$

Where f is a one-way hash function.

The signature on message m becomes:

$$sign(m) = (t,s)$$

The receiver verifies the correctness of the signature $sign(m)$ on the message m using:

$$s^e = i.t^{f(t,m)} \pmod{n}$$

The verification condition holds since:

$$s^e = i.t^{f(t,m)} \pmod{n} \Rightarrow s^e = g^e.r^{e.f(t,m)} \pmod{n}$$

Because e is relatively prime to $\varphi(n)$, we can eliminate it from the exponent on both sides to get:

$$s = g.r^{f(t,m)} \pmod{n}$$

Which is effectively the second part of the signature generated by the sender.

To improve the security of the scheme, Shamir advised on the following:

1. Expand the user's identity string i into a long pseudorandom string by planting a salt and hashing the result using a one-way hash function similar to f. The purpose of this expansion is to prevent attacks based on multiplicative relationships between users' identities.

2. Ensure that e is a sufficiently large prime number, and that f is a strong one-way hash function so that $\gcd(f, e)$ is very unlikely to be different than 1. This is important since if $\gcd(f, e)$ is $c \neq 1$, an attacker can rely on the verification condition to extract the c-th roots mod n of i, which will threaten the secrecy of g.
3. The random number r used on each signing process should never be reused, since it protects the secrecy of g in the second half of the signature.

In Double SSO, the generation of distinctive user identities and the signing of those identities is done using Shamir's scheme.

4.2. Zero-Knowledge Identification Protocols

Identification [15] is a process by which one party is assured of the identity of a second party involved in a protocol, and that this second party has really participated. An identification protocol is a protocol used to achieve identification. It engages two parties; a Prover (P) and a Verifier (V), and it has the following objectives after its completion:

1. V accepts P's identity.
2. V cannot reuse the information exchanged with P to impersonate her to a third party T.
3. There is a negligible probability that a third party T can impersonate P to V.
4. The previous objectives should remain satisfied regardless of the number of protocol runs.

Password-based identification protocols do not meet objectives 2 and 3 above, and therefore provide weak identification. Challenge-response identification protocols were suggested to provide stronger identification. In these protocols, P proves her identity to V by demonstrating the knowledge of a secret known only to P, and without revealing that secret to V. Message exchange in a general challenge-response identification protocol is illustrated in Figure 4.3.

As seen in the figure, V challenges P with a time-variant challenge, which can be a random number, a sequence number or a timestamp. P uses the challenge and her secret to compute the response that she sends to V. V uses the response and her challenge to decide whether the response is correct. If the response is correct, V understands that P is in possession of her secret and therefore accepts her identity. Otherwise, V rejects P's identity. The identification shown in Figure 4.3 is one-way, but it can be developed into two-way identification by adding one more message exchange from V to P.

Challenge-response identification protocols can be built using symmetric or asymmetric key techniques. Although these protocols provide strong identification, they are subject to chosen-plaintext/chosen-ciphertext attacks in which an attacker adaptively selects challenges and obtains responses on them that reveal partial information about P's secret.

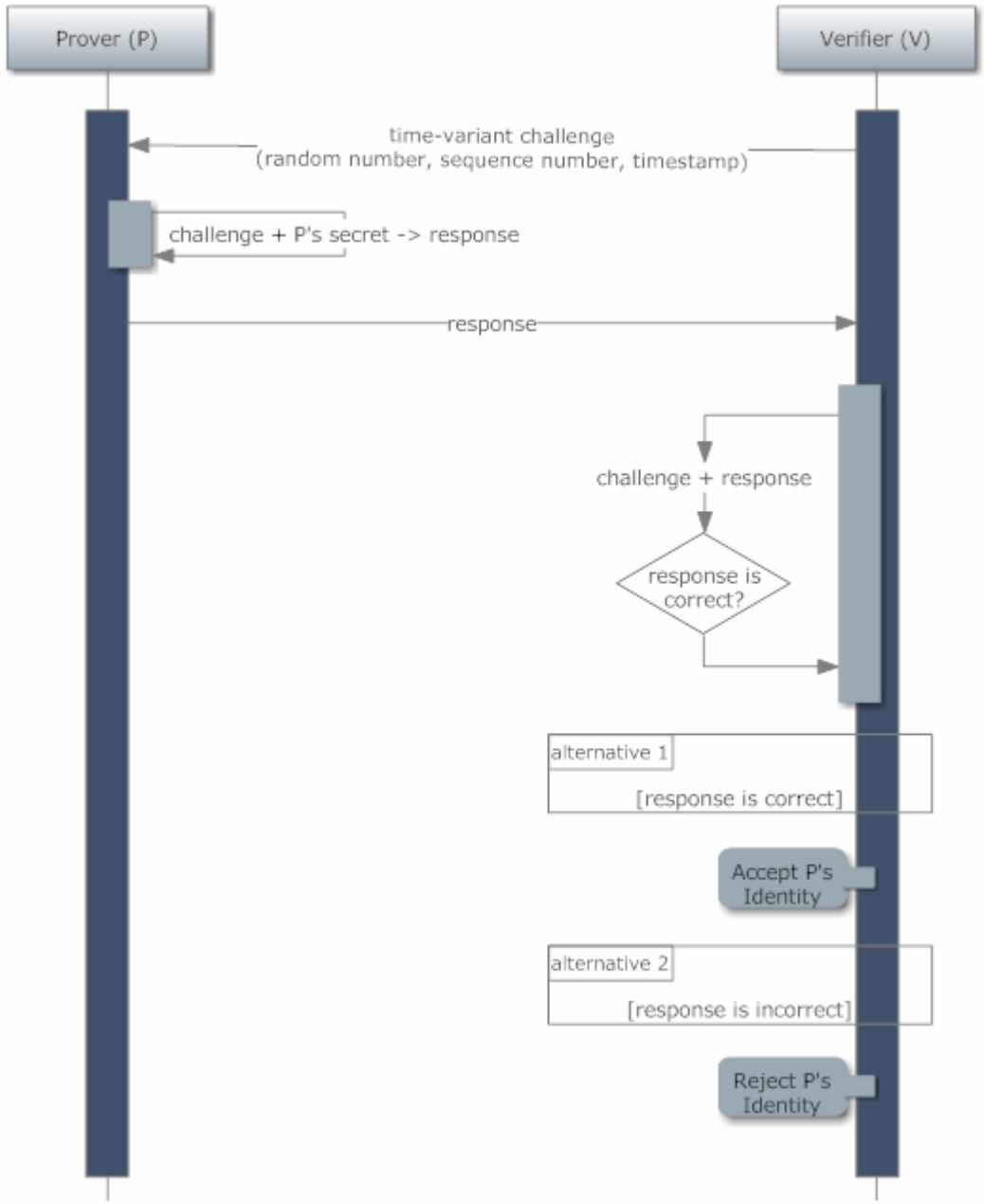


Figure 4.3. Message exchange in a general challenge-response identification protocol

Zero-knowledge identification protocols were designed to circumvent these attacks. In these protocols, P proves her identity to V by demonstrating the knowledge of a secret known only to P, and without revealing any information (not only the secret) whatsoever to V. Message exchange in a general zero-knowledge identification protocol is illustrated in Figure 4.4.

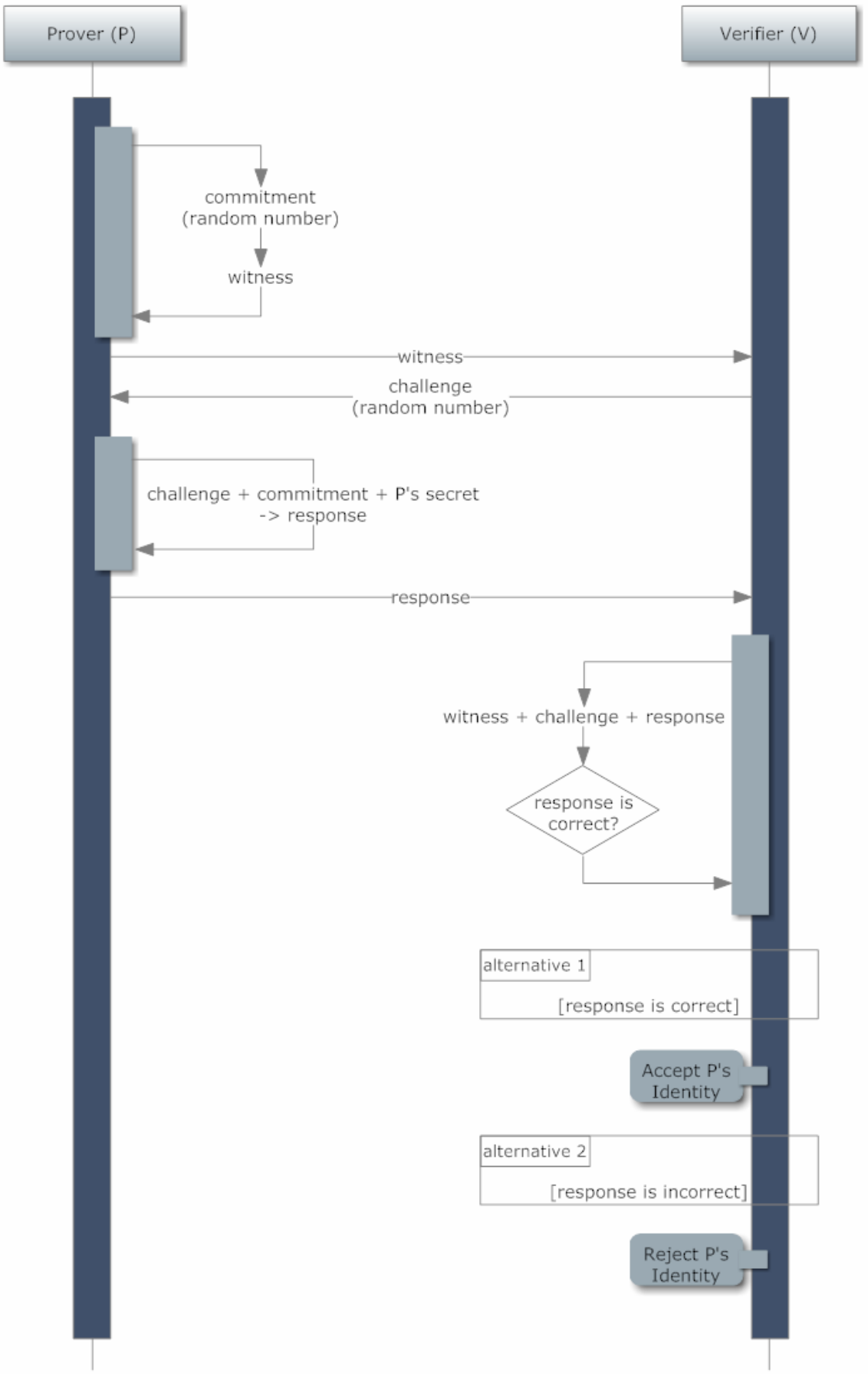


Figure 4.4. Message exchange in a general zero-knowledge identification protocol

As seen in the figure, P generates a commitment (random number) and based on this commitment she computes a witness that she sends to V. V challenges P with a random number. P uses the challenge, her commitment and her secret to compute the response that she sends to V. V uses the witness, the response and her challenge to decide whether the response is correct. If the response is correct, V understands that P is in possession of her secret and therefore accepts her identity. Otherwise, V rejects P's identity.
Note that the major difference in message exchange between zero-knowledge and challenge-response identification protocols is in P starting by generating a random commitment that is used in later steps of the protocol. This commitment makes zero-knowledge identification protocols resistant to chosen-plaintext/chosen-ciphertext attacks. There are multiple zero-knowledge identification protocols in use of which we mention, Feige-Fiat-Shamir identification protocol [16], Schnorr identification protocol [17] and Guillou-Quisquater identification protocol [18]. Double SSO uses a variant zero-knowledge identification protocol to authenticate users to service providers.

4.3. Simmons' Impersonation-Proof Identity Verification Scheme

Simmons [19] proposed an identity verification scheme based on Shamir's Identity-Based Signature Scheme and Feige-Fiat-Shamir identification protocol. Simmons' scheme relies on an issuer's public authentication channel to validate a private authentication channel belonging to a user who wants to prove her identity. These two channels can be independent and based on two different authentication algorithms. The scheme assumes a trusted issuer (similar to the KGC in Shamir's Identity-Based Signature Scheme) whose responsibility is to validate identification credentials of each user.
The issuer generates RSA public (e,n) and private (d,n) keys, where $n=pq$, $\varphi(n)=(p-1)(q-1)$ and p and p are secret large prime numbers. The issuer also chooses a suitable hash function h. The parameters e and n, and the function h are made public. d is held secret by the issuer. In addition, each user generates RSA public (e_i,n_i) and private (d_i,n_i) keys. e_i and n_i are made public, while d_i is held secret by the user. The user can then obtain her certificate from the issuer by following the protocol shown in Figure 4.5.

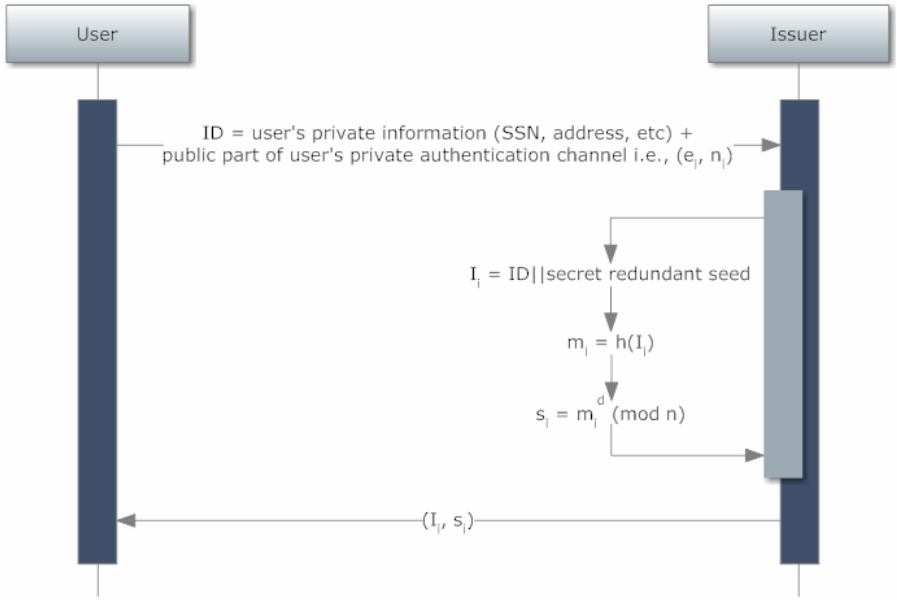


Figure 4.5. Obtaining user's certificate from the issuer in Simmons' Impersonation-Proof Identity Verification Scheme

After obtaining her certificate from the issuer, the user can prove her identity to any party by engaging in the protocol shown in Figure 4.6.

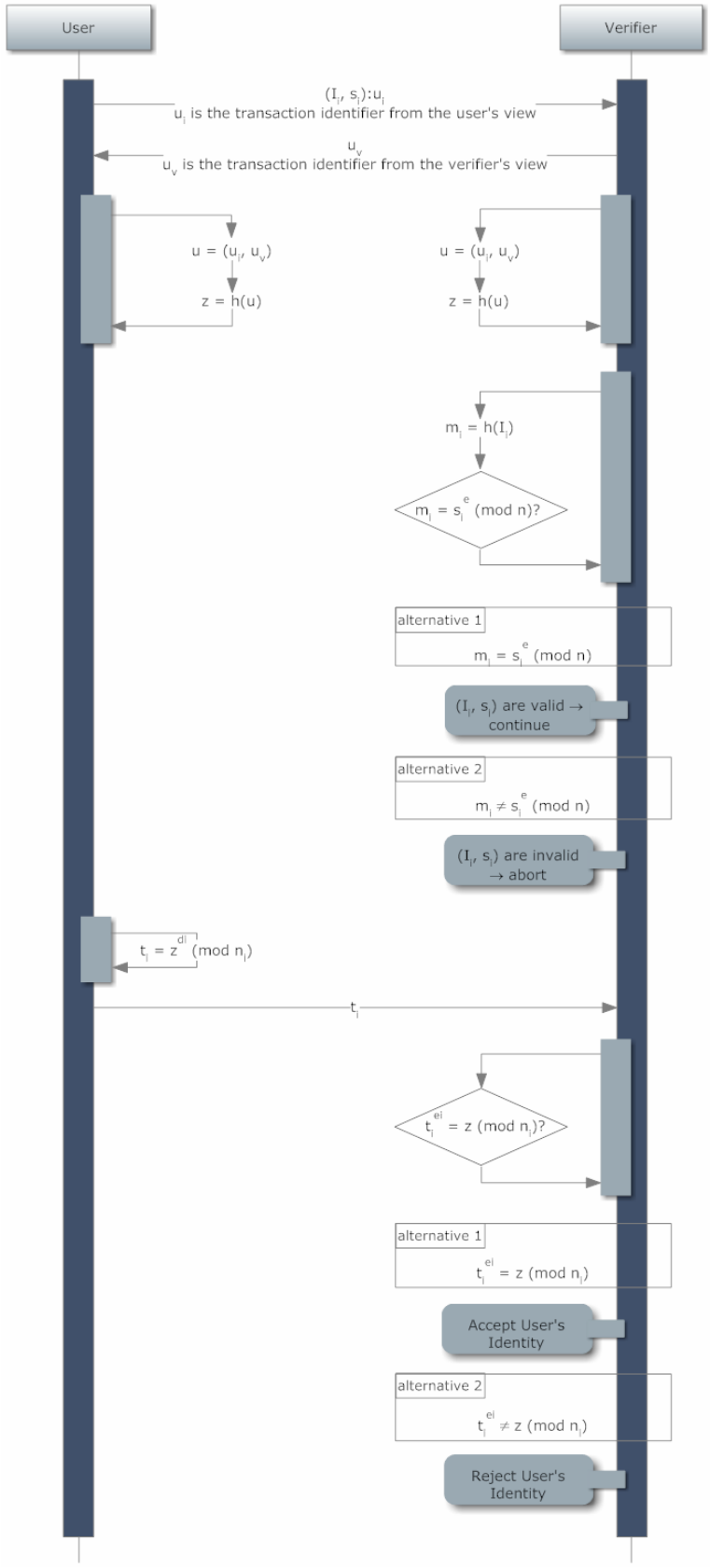


Figure 4.6. Proving user's identity in Simmons' Impersonation-Proof Identity Verification Scheme

Proving user's identity to the verifier is Simmons' scheme is somewhat similar to proving user's identity to the identity provider in Double SSO. One difference between the two is in the structure of users identities; in Double SSO, the structure is more complex. Another difference is in constructing the signature on users identities. In Simmons', it is done using RSA, whereas in Double SSO, it is done using Shamir's scheme we previously described in section 4.1.

4.4. The Weakest Link Attack

The Weakest Link Attack [20] is a parallel session attack that works only on SSO systems. In this attack, the adversary sends two service requests to two different service providers that employ two different authentication levels. The adversary uses the successful authentication response she gets from one identity provider to replace the one that failed. Doing so, the adversary succeeds at both service requests. The Weakest Link Attack cannot succeed unless all of the following conditions are met:

1. There are two or more service providers that rely on a single identity provider for user authentication.
2. The service providers use multilevel authentication.
3. One of the following conditions or both of them hold:
   - Integrity of the response message is not guaranteed.
   - The authentication level and/or the target service provider are/is not stated in the response message.
4. Redirection on the user side is active.

To illustrate how the Weakest Link Attack is launched, we consider the generic SSO solution in Figure 4.7.

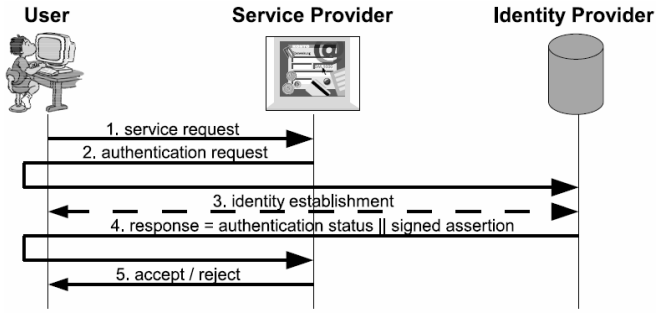


Figure 4.7. A generic SSO solution. Figure courtesy of [20]

The attack proceeds as shown in Figure 4.8.

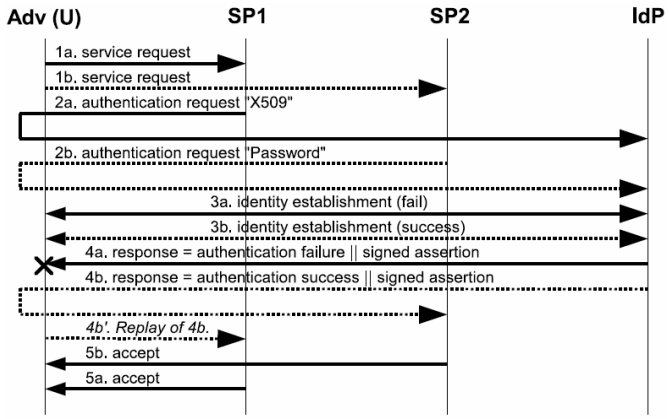


Figure 4.8. The Weakest Link Attack on a generic SSO solution. Figure courtesy of [20]

As seen in the figure, we have two service providers SP1 and SP2 that rely on a single identity provider IdP for user authentication. SP1 requires an authentication level H (certificate-based authentication, for example), while SP2 requires an authentication level L (password-based authentication, for example) where H > L. The adversary sends two service requests 1a and 1b to SP1 and SP2 respectively. SP1 asks the adversary to authenticate at level H with the IdP, while SP2 asks her to authenticate at level L. The adversary who captured user's password succeeds at authenticating at level L with the IdP, while she fails at authenticating at level H that requires certificates. Nevertheless, the adversary replays the response 4b (which is meant to access SP2) that she got from IdP in order to access SP1. The result is that the adversary will gain access to both SP1 and SP2. Thus, the Weakest Link Attack allows the adversary to succeed at a higher authentication level when she only succeeds at breaking authentication at a lower level. We will prove that Double SSO is immune to this attack in Chapter 5.

Chapter 5

Double SSO – A Prudent and Lightweight SSO Scheme

Double SSO is the major contribution of this thesis work, and as the title suggests, it is a new SSO scheme designed to be lightweight, efficient and safe to implement in any wired or wireless networking infrastructure where SSO is needed, especially if the devices used in that infrastructure are resource constrained. To quickly remind the reader of what was said in previous chapters, Double SSO is a web SSO solution (see Chapter 3), it is based on secure server-side credential caching (when considering De Clercq's taxonomy, see Chapter 2) and it can be seen as a proxy-based pseudo-SSO system (when considering Pashalidis-Mitchell's taxonomy, see Chapter 2). We expect the reader to be familiar with the building blocks of Double SSO that were introduced in Chapter 4. The chapter you are reading has the structure of a conventional conference paper, since Double SSO was submitted for publication.

Abstract

A network users is conventionally overwhelmed by the number of usernames and password she has to remember for every service with which she is registered. One solution to the security and usability implications of this situation is Single Sign-On (SSO), whereby the user authenticates only once to an identity provider and subsequently uses disparate service providers without necessarily re-authenticating. In this paper, we present Double SSO; a prudent and lightweight SSO scheme that relies on Identity-Based Signature (IBS) and comprises the following desirable features: (1) Double SSO executes a minimum number of computations on the user side and requires parties to maintain the bare minimum number of keys, (2) Double SSO is functional with identities of any type, nevertheless, it deters the Forward Search Cryptanalytic Attack, and other hostile attacks based on multiplicative relationships between different identities, (3) Double SSO can be scaled down to effectively function in ubiquitous smart environments with the additional benefit of key establishment, (4) One Stage in Double SSO can be extracted and used independently as an Identification Protocol, (5) Double SSO does not require time synchronization between involved parties, (6) Double SSO circumvent the Certificate Revocation Problem by achieving implicit certification and allowing for easy revocation of compromised identities, (7) Double SSO provably precludes the Replay Attack, the Man-in-the-Middle Attack and the Weakest Link Attack. Additionally, it is safe from repudiated parties when appropriate security devices are available.

1. Introduction

Wireless networks and the Internet have seen elevated growth in electronic commerce in recent years. With the increased dependence on computer networks for data transfer, security has turned into a predominant concern and emerged as an overarching strategy in IT infrastructures for conspicuous reasons. Among the multitude of security measures available, user authentication has been widely used to facilitate access control to networked applications and user accounts. However, as computer networks and systems proliferate to support more and more access operations, a user is required to maintain a set of authentication credentials for each service provider with which she is registered. Moreover, this user is forced to memorize all those credentials and provide them over and over again whenever she needs to access different service providers or even that same service provider multiple times. The result is a fatiguing user experience. Single Sign-On (SSO) has been proposed as a potential solution to the implications of security, credential management and usability. With SSO, a

user needs to authenticate herself to an identity provider only one, which in turn enables her to automatically log in to participating service providers she has permissions to access.
In this paper, we present Double SSO; a prudent and lightweight SSO scheme that relies on Identity-Based Signature (IBS) and comprises the following desirable features:

1. Double SSO executes a minimum number of computations on the user side and requires parties to maintain the bare minimum number of keys.
2. Double SSO is functional with identities of any type, nevertheless, it deters the Forward Search Cryptanalytic Attack, and other hostile attacks based on multiplicative relationships between different identities.
3. Double SSO can be scaled down to effectively function in ubiquitous smart environments with the additional benefit of key establishment.
4. One Stage in Double SSO can be extracted and used independently as an Identification Protocol.
5. Double SSO does not require time synchronization between involved parties.
6. Double SSO circumvent the Certificate Revocation Problem by achieving implicit certification and allowing for easy revocation of compromised identities.
7. Double SSO provably precludes the Replay Attack, the Man-in-the-Middle Attack and the Weakest Link Attack. Additionally, it is safe from repudiated parties when appropriate security devices are available.

The remainder of this paper is organized as follows: first, in Section 2, we introduce a comprehensive description of the Double SSO Scheme. Then, in Section 3, we describe how to achieve single sign-out. Thereafter, in Section 4, we scale down Double SSO to effectively function in ubiquitous smart environments. Subsequently, in Section 5, security analysis and immunity against various attacks are elucidated. Then, in Section 6, we discuss a selection of implementation issues. Last but not least, in Section 7, we compare and give critical judgment on related work. The paper concludes in Section 8.
It is noteworthy that, in our paper, we do not concern ourselves with the privacy implications SSO may have in civilian facilities (for example, see [21]).

2. Double SSO Scheme

Adopting the following notation:

Definition of Entity Symbols
IdP : Identity Provider U : User SP : Service Provider
Definition of Message Symbols
N_U : A nonce generated by U N_{IdP} : A nonce generated by IdP h : A publicly known collision-free one-way hash function
Denotation
$A \rightarrow B : \{\alpha, \beta\}$
A : Sender B : Receiver $\{\alpha, \beta\}$: A message containing two contexts that are α and β
$str_1 \parallel str_2$
String str_1 concatenated with string str_2

And assuming that U wants to authenticate to SP in order to use her services, we will break Double SSO into four stages.

Stage A. IdP Setup.

1. IdP generates RSA [22] public (e,n) and private (d,n) keys, where $n = pq$, $\varphi(n) = (p-1)(q-1)$ and p and p are secret large prime numbers. Both e and n are made public.
2. IdP constructs a secret redundant data block $seed$.

Stage B. U Registers With IdP.

1. U decides on an identity string id. The identity string is anything that can uniquely identify the user in a way that she cannot deny later [23], for example, email address, SSN, DNA, finger, face, iris, palm, gait or voice print, or any combination of these.
2. IdP constructs the block $id \,||\, seed \,||\, d_1 \,||\, d_2$ where d_1 is the identity issue date and d_2 is its expiration date. These two dates are used for identity revocation (with respect to biometrics) [24].
3. IdP generates the U's universally unique identification string using $ID = h(id \,||\, seed \,||\, d_1 \,||\, d_2)$.
4. IdP generates the U's private key x using: $x^e = ID \pmod n$. x can be easily computed by IdP since she knows $\varphi(n)$; however, and if the RSA problem is hard, attempts to solve for x by extracting the e-th roots $\bmod n$ will fail [23].
5. IdP signs the ID to get $sign(ID) = ID^e \pmod n$.
6. IdP gives ID, $sign(ID)$ and x to U. Both ID and $sign(ID)$ are made public; they need not be kept secret [25]. x must be held secret though, therefore, it should be delivered through an out-of-band channel e.g., face-to-face delivery, smart card, etc.

Stage C. U Proves Her Identity To IdP.

1. $U \rightarrow IdP : \{ID, sign(ID), N_U\}$.
2. IdP:
   2.1. IdP ensures that the ID is not expired. If it is, the flow is aborted, otherwise, it continues with Step 2.2.
   2.2. IdP verifies the correctness of the signature using $ID = sign(ID)^d \pmod n$. If verification failed, the flow is aborted, otherwise, it continues with Step 2.3.
   2.3. $IdP \rightarrow U : \{N_U + 1, N_{IdP}\}$.
3. U:
   3.1. U computes the hash $N = h(N_{IdP} + 1, N_U + 1)$.
   3.2. U uses x to sign N as follows [23]:
   - U chooses a random number t such that $0 < t < n$.
   - U computes $s_1 = t^e \pmod n$ and $s_2 = x.t^{h(s_1,N)} \pmod n$.
   - U constructs the signature $sign(N) = s_1 \,||\, s_2$.

   3.3. $U \rightarrow IdP : \{sign(N)\}$.
4. IdP verifies the correctness of the signature using $s_2^e = ID.s_1^{h(s_1,N)} \pmod n$. If verification failed, the flow is aborted, otherwise, it continues with Stage D.

Justification of Step 4.

$s_2^e = ID.s_1^{h(s_1,N)} \pmod n \Rightarrow s_2^e = x^e.t^{e.h(s_1,N)} \pmod n$

Since e is relatively prime to $\varphi(n)$, we can eliminate it from the exponent on both sides to get:

$s_2 = x.t^{h(s_1,N)} \pmod n$

Which is effectively the second part of the signature generated by U.

Stage D. IdP Authenticates U To SP.

1. $U \rightarrow SP$: request access.
2. $SP \rightarrow IdP$: request authentication of U.
3. IdP checks if U is identified. If she is not, IdP summons U to engage in Stage C above, otherwise, the flow continues with Step 4.
4. $IdP \rightarrow SP : \{R = r^e \pmod n\}$ where r is random number such that $0 < r < n$ (r and R are called the commitment and the witness respectively).
5. $SP \rightarrow IdP : \{c\}$ where c is a random number such that $0 < c < n$ (c is called the challenge).
6. $IdP \rightarrow SP : \{z = xrc \pmod n\}$ (z is called the response).
7. SP verifies that $z^e = IDRc^e \pmod n$. If it holds, SP understands that IdP possesses U's private key, therefore, SP accepts IdP's voucher for U and authenticates her, otherwise, U's authentication request is rejected.

Justification of Step 7.

$z^e = (xrc)^e \pmod n = x^e r^e c^e \pmod n = IDRc^e \pmod n$

Stage D can be rerun multiple times, without user intervention, to achieve a higher security level. In that case, U will be considered legitimate if and only if the verification condition $z^e = IDRc^e \pmod n$ holds for each run. IdP achieves SSO [26] by caching U's identification status permanently (Stage C). Every time U requests access to an SP, Stage D of the scheme will run without requiring U to re-identify to IdP. Note especially the novelty per se in Stage D where one can extract an Identification Protocol and use it for purposes other than SSO. An observant reader may have noticed that the proposed scheme derives from [S]hamir Identity-Based Signature Scheme [23], [S]chnorr Identification Protocol [27], [S]immons impersonation proof identity verification scheme [25]. Additionally, the scheme is immune to [S]immons Forward Search Cryptanalytic Attack [28], hence its name [Double] [S][S]O.

3. Achieving Single Sign-Out

In an empirical study on the usability of sign-out in an SSO system [29] carried out at Tampere University of Technology using SSO in its intranet, it was shown that all students had agreed that if there is single sign-on, then a sign-out of one of the services should sign out of all of the services. Bearing that in mind, the IdP in Double SSO achieves single sign-out by remembering the mappings between user identities and open SPs sessions. If a user requests a single sign-out, the IdP contacts each SP in turn in order to globally sign out the user.

4. Scaled-Down Double SSO Scheme for Ubiquitous Smart Environments

In an ubiquitous smart environment, computing resources are omnipresent [30], and access to these resources by end users is invisible and seamless. The advancement in wireless infrastructures, mobile computing and pervasive computing made it possible to construct such environments. In a smart environment, we can find a collection of embedded computing devices, network services, sensors, and wearable user devices that jointly achieve a seamless experience. When designing security models for a smart environment, the following considerations should be taken into account:

1. Since a user in a smart environment is often demonstrated in a form of a small mobile device or a Personal Digital Assistant (PDA), the computation power on the user side is often limited. Therefore, expensive cryptographic operations e.g., modular exponentiation should be avoided.
2. Because a server in a smart environment is often demonstrated in a form of a high-end computing device, the computation power on the server side is often high. Therefore, expensive cryptographic operations are affordable.
3. The connection between a user and a server in a smart environment has a low bandwidth. Therefore, the number of exchanged messages between the two should be limited, moreover, messages themselves should be short.
4. A large number of users enter and leave a smart environment per day and many of them may not stay in it for more than seconds. This makes the relationship between a user and a server in a smart environment short-lived and volatile. Therefore, storing user profiles or keying material on a server for a long period should be avoided.

We will scale down our Double SSO Scheme to fit in an ubiquitous smart environment. To save space, we will only highlight the differences between the general and the scaled-down versions of the scheme.

Stage A. IdP Setup.

No change.

Stage B. U Registers With IdP.

Step 2 is replaced with: IdP constructs the block $id \parallel seed$.

Step 3 is replaced with: IdP generates the U's universally unique identification string using $ID = h(id \parallel seed)$.

Step 7 is added: IdP destroys x.

Stage C. U Proves Her Identity To IdP.

Step 3.2 is cancelled.

Step 3.3 is replaced with: $U \rightarrow IdP : \{N\}$.

Step 4 is replaced with: IdP verifies that the hash was not tampered using $N = h(N_{IdP} + 1, N_U + 1)$. If verification failed, the flow is aborted, otherwise, it continues with Stage D.

Stage D. IdP Authenticates U To SP.

Step 5 is replaced with: $SP \rightarrow U : \{c\}$ where c is a random number such that $0 < c < n$.

Step 6 is replaced with: $U \rightarrow SP : z = xrc \pmod{n}$.

In this scaled-down version of Double SSO Scheme, we have the advantage of establishing a session key for subsequent confidential communications between U and SP (only). The session key is simply $h(z)$.

5. Security Analysis

In our security analysis, we assumed that the underlying cryptography; namely the hash function h, is invulnerable with regard to message integrity, hence, we did not consider attacks such as the Birthday Attack. On the other hand, we assumed that any attacker can inject messages on any link at any time. In addition, any attacker can eavesdrop, change, drop and redirect all exchanged messages along any link. Moreover, any attacker can resend exchange messages recorded from past communications.

5.1. Attacks on Security Parameters

The security parameters in Double SSO Scheme are:

1. IdP's security parameters: p, q, d and x for each U.

2. SP's security parameters: None.
3. U's security parameters: x.

The security parameters p, q and d are the IdP's RSA security parameters. As long as p and q are large enough prime numbers, breaking the RSA security parameters (computing p, q and d from e and n) is an exceedingly difficult computational task [22]. We recommend that p and q be at least 512-bit integers. Then the corresponding n would be 1024-bit integer. x is known only to IdP and U and attempts to extract it from ID will fail (for the reason mentioned above, in Step 4 of Stage B of the scheme), therefore x is secure. No security parameters need to be considered with regard to SP. In summary, all security parameters are secure under known security attacks.

5.2. Attacks on Identity Proof

The theft of other users' IDs and security parameters is impractical. Moreover, the use of nonce data in Stage C of the scheme renders replay and forge attacks on identity proof intractable.

5.3. The Replay Attack

Theorem. Double SSO is safe from the Replay Attack [31: 144].

Proof. We prove the safety from the Replay Attack as follows:

1. On Stage C: A Replay Attack for the message in Step 3.3. In this case, an attacker can try to attack by sending the captured message; however, the attacker cannot succeed because the signature in the captured message includes nonce data.
2. On Stage D: A Replay Attack for the message in Step 6. In this case, an attacker cannot reuse the message because the random numbers r and c are altered in each authentication.

5.4. The Man-in-the-Middle Attack

Theorem. Double SSO is *satisfactorily* safe from the Man-in-the-Middle Attack.

Proof. We prove the *satisfactory* safety from the Man-in-the-Middle Attack as follows:

1. On Stage C: The attacker A will start a Man-in-the-Middle Attack [32] on Stage C as follows.

- The attacker A has access to both ID and $sign(ID)$. She generates a bogus nonce N_A, constructs the message $\{ID, sign(ID), N_A\}$, and sends it to IdP.
- IdP completes Steps 2.1 and 2.2, and sends $\{N_A+1, N_{IdP}\}$ to the attacker A in Step 2.3.
- The attacker A computes the hash $N = h(N_{IdP}+1, N_A+1)$.
- The attacker A probes U with a polynomial number λ of queries so as to obtain the signatures $sign(N_i) = [s_1 \,||\, s_2]_i = t_i^e \pmod n \,||\, x.t_i^{h(s_1,N_i)} \pmod n$ where $N_i \neq N$ for any $i = 1,2,..., \lambda$.
- To succeed with her impersonation of U, the attacker A has to generate a well-formed signature $sign(N) = s_1 \,||\, s_2 = t^e \pmod n \,||\, x.t^{h(s_1,N)} \pmod n$ from (t, n, N) but without knowing x.
- The attacker A cannot proceed to achieve this through random search. Even though there is a large number of valid signatures of N depending on the attacker A choice of t, the density of these signatures is too low that a random search is extremely unlikely to discover any of them [23].

- One way the attacker A can proceed is through trying to isolate x by analyzing the valid signatures $sign(N_i)$ she received from U or by manipulating the verification conditions $s_2^e = ID.s_1^{h(s_1,N_i)} \pmod n$. In both cases, the attacker A will have to extract the modular roots which is believed to be an exceedingly difficult computational task.
- The last way the attacker A can proceed it through guessing the second half of the signature s_2 with a probability of $1/|x|$ of being right. If we suppose that p and q are both 512-bit integers, then x is 1024-bit integer, and the probability of a correct guess is only 2^{-1024}. The probability of success cannot be increased unless solving the RSA problem is easy.

2. On Stage D: The attacker A will start a Man-in-the-Middle Attack on Stage D as follows.

- The attacker A generates a bogus random number r_A such that $0 < r_A < n$, computes $R_A = r_A^e \pmod n$, and commits herself to SP by sending $\{R\}$.
- SP generates a random number c such that $0 < c < n$ and challenges the attacker A by sending $\{c\}$.
- The attacker A, who do not know x, has to compute a valid response $z_A = xr_Ac \pmod n$ in order to succeed with her impersonation of IdP.
- The only way the attacker can proceed it through guessing the response z_A with a probability of $1/|z_A|$ of being right. Again, if we suppose that p and q are both 512-bit integers, then z_A is 1024-bit integer, and the probability of a correct guess is only 2^{-1024}. The probability of success cannot be increased unless solving the RSA problem is easy.

5.5. The Weakest Link Attack

Theorem. Double SSO is safe from the Weakest Link Attack.
Proof. We prove the safety from the Weakest Link Attack as follows:
Double SSO is safe from the Weakest Link Attack [33] since all authentication operations done by IdP enjoy the same level of immunity.

5.6. Repudiated Parties

Theorem. Double SSO is safe from repudiated parties, when appropriate security devices are available.
Proof. We prove the safety from repudiated parties as follows:

1. On Stage C: After a complete run of Stage C, the alleged U [31: 142] can claim that IdP has produced the message in Step 3.3 herself. In fact, U succeeds since IdP has x and can produce a signature on N.
2. On Stage D: After a complete run of Stage D, the alleged IdP can claim that SP has produced the message in Step 6 herself. In fact, IdP succeeds since SP can generate c and z at random and put $R = z^e ID^{-1} c^{-e} \pmod n$.

In order to refute such claims made by U and IdP, a security device that does not pose an overhead on the system can be utilized e.g., the referee server described in [34]. The referee server generates binding information on the fly between entities and authentication messages, and retains these information signed by its private key in its local storage for future accusation.

5.7. The Forward Search Cryptanalytic Attack

Theorem. Double SSO is safe from the Forward Search Cryptanalytic Attack, and from other hostile attacks based on multiplicative relationships between different identities.

Proof. We prove the safety from the Forward Search Cryptanalytic Attack, and from other hostile attacks based on multiplicative relationships between different identities as follows:
IdP constructs a secret redundant data block $seed$ and plants it in identities before hashing them using h. This foils the Forward Search Cryptanalytic Attack and prevents other hostile attacks based on multiplicative relationships between different identities [23][28].

5.8. Other Security Issues

- User registration with IdP is important to eliminate the possibility for an illegitimate user to impersonate and/or steal a valid user's ID.
- The association between a user's ID and her private key x using $x^e = ID \pmod n$ is effective, strong and can only be determined by IdP.
- IdP does not sign a user's ID using her decryption exponent d as doing so would reveal the private key x associated with the ID.
- The identity check of user identity in Stage C is thorough.
- Time synchronization between U, IdP and SP is not necessary, because timestamps are not used to block the Replay Attack [35].
- The keys in Double SSO are used only to generate and verify signatures (never to encrypt and decrypt), thus, in Double SSO, we do not need to deal with identity status queries, because a user sends a proof of the validity of her identity to IdP on each run of Stage C of the scheme – remember that the issue and the expiration dates are planted in the ID [24]. This allows for easy revocation of compromised identities.
- Since private keys of all users are generated by IdP, a user can authenticate to SPs if and only if IdP has generated a private key to her. This achieves implicit certification, an important feature in IBS [24].
- Unfortunately, Double SSO still suffers of the major disadvantage in IBS; namely private key escrow [36][24][37][38] i.e., IdP knows each user's private key and can therefore authenticate to any SP claiming the identity of any user. Key escrow in Double SSO can be easily mitigated by coercing SP to challenge U in an additional run in Stage D of the scheme.

6. Implementation Issues

To implement Double SSO Scheme, each user needs to maintain the parameters ID, $sign(ID)$ and x. If we suppose that both p and q are 512-bit integers, h produces 128-bit hash values, then the total number of bits a user needs to store is $|ID|+|sign(ID)|+|x|$ = 128 + 2048 + 1024 = 3200 bit ≈ 0.39 kilobyte. This is small enough to fit in virtually any smart card.
In Stage C of the scheme, the computations required on the user side are very limited; only a hash and two modular exponentiations. These simple computations can be effectively implemented in any resource constrained environment e.g., smart cards or handheld devices. On the IdP and SP side, computations are as well minimal, although computations on this side are not much of a concern, because both IdP and SP are supposed to be powerful computing devices.
Because of the fear that IdP might become a single point of failure, an implementer of the Double SSO Scheme should invest a reasonable effort on availability issues by, for example, creating two separate subsystems; one for users' identities management and the other for identities proving and authenticating. The data related to users' identities is better stored under internal redundancy in a distributed database. Additionally, it is recommended to install the IdP itself on a number of servers to ensure high availability, load balancing and graceful

degradation. The IdP itself is an attractive target for DoS attackers, therefore, an effective architecture (for example Fosel [39][40]) capable of thwarting this attack should be considered as part of the implementation.

7. Related Work

In our consideration of the related work, we note that a division can be made according to some specific criteria. The criteria we realized are ubiquitous smart environments, dynamically created session passwords, filtering HTTP request using an SSO server and secret sharing.

7.1. Related Work in Ubiquitous Smart Environments

To our knowledge, the SSO and key establishment scheme proposed by Chan, Fleissner, Liu and Li [30] is the only scheme specifically designed for ubiquitous smart environments that appears in the literature. In Table 1, we compare between Scaled-Down Double SSO Scheme and Chan-Fleissner-Liu-Li SSO Scheme.

Table 1. Comparision between Scaled-Down Double SSO and Chan-Fleissner-Liu-Li SSO Scheme

Comparison Criterion	Scaled-Down Double SSO Scheme	Chan-Fleissner-Liu-Li SSO Scheme
Entity authenication is achieved?	Yes	Yes
Key establishment is achieved?	Yes	Yes
User anonymity is achieved?	Yes, through the application of h, a collision-free one-way hash function, on user's real identity	Yes, through a one-time identity alias $U(CTR)$
User anonymity is suitable for critical services such as account withdrawals?	Yes, because ID is eventually bound to a real user	No, a real user identity must be considered instead of the identity alias vector $U = U(1...n)$
IdP has access to the session key shared between U and SP?	No, provided that x is destroyed after communicating it to U	Yes, the key seed vector $SEED = SEED(1...n)$ cannot be destroyed on IdP, because it is included in the $< response >$ message sent from IdP to SP in Step 4 of the scheme
Heavy computations on the user side?	Only two modular multiplications	No
User profiles or keying material are stored on SP?	No	No
Total length of exchanged messages with the user	If we suppose that both p and q are 512-bit integers, h produces 128-bit hash values and the nonces are 128-bit length, we can gauge	If we suppose that each seed in the key seed vector (and each identity alias in the identity alias vector) is 128-bit length, the number of seeds (and the number of aliases) is 1 for one

	the total length of exchanged messages with the user: $\|\{ID, sign(ID), N_U\}\|$ = 128 + 2048 + 128 = 2304 bit $\|\{N_U + 1, N_{IdP}\}\|$ = 128 + 128 = 256 bit $\|\{N\}\|$ = 128 bit $\|\{c\}\|$ = 1024 bit $\|\{z = xrc \pmod n\}\|$ = 1024 bit Total length = 2304 + 256 + 128 + 1024 + 1024 = 4736 bit ≈ 0.58 kilobyte	sign-on attempt, the shared secret key is 128-bit length, *IdP* 's identity (and U's identity) is 128-bit length and the maximal size of the signature generated is 1024 bit, we can gauge the total length of exchanged messages with the user: $\|sk, U, SEED, n\|$ = 128 + 128 + 128 + 8 = 385 bit $\|Service\ request, U(CTR), IdP\|$ = 8 + 128 + 128 = 264 bit $\|CTR\ synchronization\|$ = 8 bit $\|accept / reject\|$ = 8 bit Total length = 385 + 264 + 8 + 8 = 665 bit ≈ 0.08 kilobyte
	The total length is around 7.25 times larger in the Scaled-Down Double SSO Scheme than it is in Chan-Fleissner-Liu-Li SSO Scheme in the first sign-on attempt. However, in the second and later sign-on attempt, only $\|\{c\}\|$ and $\|\{z = xrc \pmod n\}\|$ are exchanged with the user in Scaled-Down Double SSO. This reduces the total length to only 2048 bit ≈ 0.25 kilobyte; that is 3.12 times larger than it is in Chan-Fleissner-Liu-Li SSO Scheme.	

7.2. Related Work Based on Dynamically Created Session Passwords

Tiwari and Joshi [41] proposed an SSO scheme based on the generation of One-Time Passwords (OTPs) using the Lamport's scheme [42]. In their scheme, a database is maintained by a portal, and this database stores usernames, passwords, total number of times a password should be generated (the counter) and a challenge question for each user. In Tiwari-Joshi scheme, the initial secret used for subsequent OTPs generation is the user password itself hashed using SHA-1. Tiwari-Joshi scheme has the following downsides:

1. MD5 is the underlying hash function used in Lamport's scheme and MD5 is completely broken.
2. Lamport's scheme and its variants are vulnerable to the Replay Attack [43: 397] where an attacker intercepts and obtain an unused OTP for later impersonation purposes.
3. The application does not save the current OTP on its side, rather, the initial secret is transferred from the portal to the application on each sign-on attempt and over a non-authentic channel. This opens for various attacks on the initial secret.
4. The application does not maintain the counter on its side either, rather, the counter value is transferred again from the portal to the application on each sign-on attempt and over a non-authentic channel. An attacker can intercept the value setting it to zero on each sign-on attempt, thus denying access to the application, and locking the user in an infinite loop of password update requests.
5. The channel a user uses to update her password is neither secure nor authentic.

Fleury, Basney and Welch [46] proposed an SSO solution based on what they called "session passwords"; they are short-lived, dynamically created passwords that are used instead of

user's original long-lived password for repeated authentication. They are somewhat similar to OTPs. Fleury-Basney-Welch solution has the following downsides:

1. The solution is not generic; it enables an SSO experience only from a Java Web Start (JWS) application launched from the user's local file system.
2. Launching any JWS application requires downloading a Java Network Launching Protocol (JNLP) file to the local file system, and in Fleury-Basney-Welch solution, the session password is stored in that file. Unfortunately, neither the browser nor JWS makes any effort to ensure the privacy of the JNLP file or remove it when it is no longer needed. This means that the session password can be compromised before expiration especially in multi-user systems.
3. The solution uses MyProxy X.509 credential management system. MyProxy CA will ultimately face the Certificate Revocation Problem [24].

7.3. Related Work Based on Filtering HTTP Request Using an SSO Server

Pashalidis and Mitchell [44] proposed an SSO system that is based on a trusted HTTP proxy, and that is suitable for use from untrusted network access devices e.g., public terminals. The proxy keeps a copy of user's long-term authentication credentials in a protected credential database. All traffic between the untrusted device and the network service provider is routed through the proxy. The proxy intercepts user's login requests and asks her for authentication using a challenge/response mechanism. After successful authentication to the proxy, the proxy performs legacy authentication to the network service provider on the behalf of the user using the long-term credentials. The advantage of this approach is that long-term credentials never reach the untrusted device. Pashalidis-Mitchell-2 system has the following downsides:

1. The proxy requires a large amount of customization to be able to recognize and detect user's login requests to different network service providers. Any change in the authentication interface of the network service provider requires a customization of the proxy.
2. Forcing each HTTP request to route through the proxy may degrade performance and detract from user's experience.
3. A browser (and possible a firewall) configuration is needed to put the proxy to work.

Geihs, Kalcklösch, and Grode [45] proposed an SSO solution based on a reverse SSO server that transcodes HTTP requests. They extended the Apache HTTP Server with a module written in Perl. This module parses the contents of plain HTML documents, looks for embedded hyperlinks pointing to protected web applications and replaces them with ones pointing to the reverse SSO server instead. A user is expected to authenticate to the reverse SSO server only once. Geihs-Kalcklösch-Grode solution has the following downsides:

1. The solution needs an SSL/TLS channel between the user's browser and the reverse SSO server. An SSL/TLS channel increases authentication latency, moreover, it requires a Public Key Infrastructure (PKI) to be in place which further complicates the solution.
2. The reverse SSO server cannot handle web content written in languages other than HTML.
3. The Perl Parser module used in the implementation can degrade connection quality.
4. Forcing each HTTP request to pass through the reverse SSO server may degrade performance and detract from user's experience.

7.4. Related Work Based on Secret Sharing

Chen, Zhu, Li and Cheng [51] proposed a web-based, distributed SSO system, called ThresPassport, based on secret sharing. Chen-Zhu-Li-Cheng system has the following downsides:

1. The system requires service providers to maintain a pair of public and private keys. This renders the system inappropriate for web-based SSO authentication services.
2. The threshold value used to divide secret keys is fixed during the setup phase of the system, although it may be necessary to change this value later on.

Brasee, Makki and Zeadally [52] proposed an SSO scheme based on secret sharing too. Brasee-Makki-Zeadally scheme has the following downsides:

1. The parties involved in the scheme are required to keep a pair of public and private keys. This renders the scheme inappropriate for web-based SSO authentication services.
2. Too many assumptions were made about a functionally specified device, which is the USBID. Many of these assumption do not hold in reality.

Furukawa, Sako, and Obana [53] proposed an SSO system based on secret information shared between users' IC cards and portals. Furukawa-Sako-Obana system stores a password and an authentication key for each registered user in a portal. The password and the authentication key are encrypted using the user's secret key, which prevents the portal from impersonating the users. The password authenticates the user while the authentication key authenticates the user's IC card. If both the user and the IC card are authenticated, the portal transmits the encrypted secrets to the user via a secure channel. The IC card decrypts the encrypted secrets and uses them to authenticate the user to the service. Although Furukawa-Sako-Obana system remains secure even if an attacker succeeds in mounting two attacks e.g., stealing the password and comprising the portal, and although it was proven to be SSO-AKE-MA-secure, it involves the use of a multitude of signing/verifying keys and requires multiple runs of Authenticated Key Exchange (AKE) protocols and Password-Based Authenticated Key Exchange (PAKE) protocols, in addition to the use of Tweakable Block Cipher, first introduced in [54]. This heavy-duty usage of cryptographic primitive greatly complicates the system.

7.5. Other Related Work

Pashalidis and Mitchell [26] proposed an SSO protocol where a user's Trusted Platform (TP) plays the role of an Authentication Service Provider (ASP) that subsequently signs on the user to a number of Service Providers (SPs) without the need to re-authenticate. The TP used in Pashalidis-Mitchell protocol conforms to the specification of the Trusted Computing Platform Alliance (TCPA). Pashalidis-Mitchell-1 protocol has the following downsides:

1. The protocol is vulnerable to the Man-in-the-Middle Attack [32], and to counter it, the authors suggested the installation of an SSL/TLS channel. An SSL/TLS channel increases authentication latency, moreover, it requires a Public Key Infrastructure (PKI) to be in place which further complicates the protocol.
2. The protocol may require a third party to maintain Certificate Revocation Lists (CRLs) the SPs consult to check the status of the certificates used. CRLs are known to be an inefficient solution to the Certificate Revocation Problem [24]. As the number of users grows, CRLs become quite long and transmitting them to all SPs is costly.
3. Identities used in the protocol are bound to the TPs where they were created and can only be used on them or on a TCPA-compliant device. This greatly reduces users' mobility.
4. The protocol is quite complex and has a high operational overhead as certificates are heavily issued and validated.

Zhu and Diao [47] proposed an SSO solution based on an Authentication Broker Server, a repository of account information and a monitoring plug-in installed on user's Internet Explorer. Zhu-Diao solution has the following downsides:

1. The solution requires the server to return a credential list when a user signs on. As the number of users grows, the credential list becomes quite long and transmitting it to all users is costly.

2. SSL/TLS is used in the solution which increases authentication latency, moreover, it requires a Public Key Infrastructure (PKI) to be in place which further complicates the solution.
3. The IE plug-in used in the solution acts much like a spyware on the user in that it captures HTTP post data and sends it to the Authentication Broker Server for distilling and processing. An attacker can spoof IP addresses and have HTTP post data forwarded to her own machine. This can compromise the whole list of user's credentials.

Wu, Yao and Bao [48] proposed a robust mechanism for defeating phishing and pharming attacks against SSO schemes. Wu-Yao-Bao scheme is cookie-based. The browser stores a cookie composed of username, password and server's public key on the user machine. When the browser opens an SSL/TLS channel with the server, the stored cookie is retrieved, doubly-encrypted (first with the server's public key and then with the SSL/TLS session key), then it is sent to the server. The server performs double-decryption to extract the username and the password and compare them against a database before deciding on authenticating the user. The presence of the cookie on the user machine achieves an SSO experience since the user needs to fill in her credentials only once. Wu-Yao-Bao scheme has the following downsides:

1. Registering a cookie and reading it require the browser to establish an SSL/TLS channel with the server. An SSL/TLS channel increases authentication latency, moreover, it requires a Public Key Infrastructure (PKI) to be in place which further complicates the scheme.
2. Cookie-based SSO schemes suffer from session hijacking i.e., exploitation of valid session. A subsequent user may recover the preceding user's authentication cookie to impersonate her and illegally access service providers [49].

Wu, Yao and Bao compared their work with earlier countermeasures against the phishing and pharming attacks. The comparison was in terms of client effort in online transaction, and it proved the effectiveness of Wu-Yao-Bao scheme. For further information, refer to [48].

Ren [50] proposed an SSO scheme based on the intractability of the RSA problem, the discrete logarithm problem and the subset-sum NP-complete problem all combined. Ren scheme has the following downsides:

1. The setup phase of the scheme is exhausting. It requires a user to register with the trusted authority and with each organization. Moreover, it requires each organization to register with the trusted authority.
2. The identity proof phase of the scheme should be repeated multiple times before a user can be considered legitimate.
3. The parties involved in the scheme are required to keep a multitude of secret parameters when compared to the minimal number of secret parameters they are required to keep in Double SSO.
4. The space requirements on the (normally resource constrained) user side are more than they are in Double SSO.

Singh and Pais [55] proposed an SSO framework for web applications based on Identity-Based Encryption (IBE) instead of Public Key Infrastructure (PKI). In Singh-Pais framework, messages exchanged during the runtime phase are encrypted with the recipient's URL as a public key, but their authenticity and freshness are not guaranteed. This opens for the Replay Attack and Man-in-the-Middle Attack.

Hillenbrand, Göotze, Müller and Müller [56] proposed an SSO architecture based on extending the Kerberos protocol to provide authorization information and delegation of access privileges. The major disadvantage of Hillenbrand-Göotze-Müller-Müller architecture is that Kerberos is not suitable for use in untrusted and highly dynamic environments, such as the Internet [57].

Josephson, Sirer and Schneider [58] proposed a distributed SSO service called CorSSO. Additionally, Hui and Ting [59] proposed an SSO protocol that is based on a security token. This token stores the session key between a user and the authentication server in order to promote performance. The major disadvantage of both Josephson-Sirer-Schneider and Hui-Ting schemes is that parties involved are required to keep a pair of public and private keys. This renders both schemes inappropriate for web-based SSO authentication services.
The works of Eslami-Darvish-Rahmani [60] and Ma-Chen-Li-Luo [61] rummage through a collection of ideas that do not add to the comparative part of our paper.

8. Conclusions

In this paper, we presented Double SSO; a prudent and lightweight SSO scheme that relies on Identity-Based Signature (IBS) and comprises a bunch of desirable features. We started with a detailed description of the Double SSO Scheme, then we described how to achieve single sign-out. Thereafter, we scaled Double SSO down to function in ubiquitous smart environments. Subsequently, we showed how immune Double SSO is to various attacks; starting with attacks on security parameters and identity proof, moving to the Replay Attack, the Man-in-the-Middle Attack, the Weakest Link Attack and the Forward Search Cryptanalytic Attack, and ending with repudiated parties. We further showed how Double SSO circumvent the Certificate Revocation Problem by achieving implicit certification and allowing for easy revocation of compromised identities. Finally, we gave critical judgment on related work, and compared the scaled-down version of Double SSO with the similar work in [30].
The minimum number of keys and computations by which Double SSO runs makes it an appealing and low-cost SSO choice in wireless networks, ubiquitous smart environments and resource constrained devices. We encourage the reader to verify the evident appeal of Double SSO by implementing it and analyzing its performance. We also encourage cryptanalysts to expose it to their logic analyzers and model checking tools.

Chapter 6

Conclusions and Future Work

Despite the wide acceptance of SSO as a convenient access control method, and despite the large number of SSO solutions that were developed so far and pumped into the market, choosing the right SSO solution for a specific organization is still a daunting and confusing task for most security professionals.

In this thesis, we tried to resolve this confusion by presenting the taxonomies of SSO solutions and their qualities, and by describing the architectures and operations of a selection of SSO solutions in use today. This was done in Chapters 2 and 3.

We also had to pay attention to the large number of SSO schemes that appear in the academic literature. Studying those schemes, we realized that many of them either require parties to maintain a large number of keys (and therefore require the exchange of a large number of messages), or assume a cryptographic protocol in place (for example, SSL/TLS). As a consequence, many of those schemes suffer of a high operational overhead. In an attempt to increase efficiency while preserving the SSO experience, we developed a new SSO scheme that we suppose efficient, safe and suitable for any networking infrastructure, especially for resource constrained ones. Chapters 4 and 5 of this thesis work concentrated on our new scheme.

Unfortunately, the analysis of SSO schemes that appear in the literature (including our new scheme) is done manually by observing traditional attacks (such as, the Reply Attack) and how they can be mounted against a scheme run. In addition, the proofs given in most analyses are not based on sound logic. This level of informality is inevitable since automatic verification tools at the moment (for example, Scyther [62]) can only handle concrete cryptographic protocols that have a narrower purpose that that of SSO schemes. A possible future line of research would be to expand the functionality of current verification tools and logic analyzers to contain SSO schemes.

References

[1] J. De Clercq. Single Sign-On Architectures. *Proceedings of the International Conference on Infrastructure Security*, Bristol, United Kingdom, 2002.

[2] J. De Clercq and G. Grillenmeier. *Microsoft Windows Security Fundamentals*. Elsevier, Oxford, UK, 2007.

[3] Courion PasswordCourier. Available at: http://www.courion.com/products/PasswordCourier.html.

[4] Citrix Password Manager. Available at: http://www.citrix.com/metaframepasswordmanager.

[5] A. Pashalidis and Chris J. Mitchell. A Taxonomy of Single Sign-On Systems. *Proceedings of the 8th Australasian Conference on Information Security and Privacy*, Wollongong, Australia, 2003.

[6] H. F. Tipton and M. Krause. *Information Security Management Handbook, Sixth Edition*. Auerbach Publications, Boca Raton, Florida, USA, 2007.

[7] SAML Single Sign-On (SSO) Service for Google Apps. Available at: http://code.google.com/googleapps/domain/sso/saml_reference_implementation.html.

[8] SAML-based Single Sign-On for Google Hosted Services - Demo Tool. Available at: http://code.google.com/apis/apps/sso/saml_static_demo/saml_demo.html.

[9] Introduction to Windows Live ID. Available at: http://msdn.microsoft.com/en-us/library/bb288408.aspx.

[10] Introducing Windows CardSpace. Available at: http://msdn.microsoft.com/en-us/library/aa480189.aspx.

[11] Understanding Windows Live Delegated Authentication. Available at: http://msdn.microsoft.com/en-us/library/cc287613.aspx.

[12] K. Geihs, R. Kalcklösch and A. Grode. Single Sign-On in Service-Oriented Computing. *Proceedings of the 1st International Conference on Service-Oriented Computing*, Trento, Italy, 2003.

[13] The Project Liberty. Available at http://www.projectliberty.org.

[14] A. Shamir. Identity-Based Cryptosystem and Signature Scheme. Proceedings of CRYPTO 84, Santa Barbara, California, USA, 1984.

[15] A. J. Menezes, P. C. van Oorschot and S. A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 2001.

[16] U. Fiege, A. Fiat and A. Shamir. Zero knowledge proofs of identity. *Proceedings of the nineteenth annual ACM symposium on Theory of computing*, New York, USA, 1987.

[17] C. P. Schnorr. Efficient Identification and Signatures for Smart Cards. *Proceedings of EUROCRYPT 89*, Houthalen, Belgium, 1989.

[18] L. C. Guillou and J.-J. Quisquater. A practical zero-knowledge protocol fitted to security microprocessor minimizing both transmission and memory. *Proceedings of EUROCRYPT 88*, Davos, Switzerland, 1989.

[19] G. J. Simmons. An Impersonation-Proof Identity Verification Scheme. *Proceedings of CRYPTO 87*, Santa Barbara, California, USA, 1987.

[20] Y.-Y. Chan. Weakest Link Attack on Single Sign-On and Its Case in SAML V2.0 Web SSO. *Proceedings of the International Conference on Computational Science and Its Applications*, Glasgow, United Kingdom, 2006.

[21] R. R. Heckle and W. G. Lutters. Privacy implications for single sign-on authentication in a hospital environment. *Proceedings of the 3rd Symposium on Usable Privacy and Security*, Pittsburgh, Pennsylvania, United States, 2007.

[22] R. L. Rivest, A. Shamir and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, Volume 21, Issue 2, pp. 120-126, 1978.
[23] A. Shamir. Identity-Based Cryptosystem and Signature Scheme. *Proceedings of CRYPTO 84*, Santa Barbara, California, United States, 1985.
[24] C. Gentry. Certificate-based encryption and the certificate revocation problem. *Proceedings of the 22nd International Conference on Theory and Applications of Cryptographic Techniques*, Warsaw, Poland, 2003.
[25] G. J. Simmons. An Impersonation-Proof Identity Verification Scheme, *Proceedings of a Conference on the Theory and Applications of Cryptographic Techniques on Advances in Cryptology*, 1987.
[26] A. Pashalidis and C. J. Mitchell. Single Sign-On Using Trusted Platforms. *Proceedings of the 6th International Conference on Information Security*, Bristol, United Kingdom, 2003.
[27] C. P. Schnorr. Efficient Identification and Signatures for Smart Cards. *Proceedings of EUROCRYPT 89*, Houthalen, Belgium, 1989.
[28] G. J. Simmons and D. Holdridge. Forward Search as a Cryptanalytic Tool Against a Public Key. *Proceedings of the 1982 IEEE Symposium on Security and Privacy*, Oakland, California, USA, 1982.
[29] M. Linden and I. Vilpola. An Empirical Study on the Usability of Logout in a Single Sign-on System. *Proceedings of the 1st International Conference on Information Security Practice and Experience*, Singapore, 2005.
[30] Y.-Y. Chan, S. Fleissner, J. K. Liu and Jin Li. Single Sign-On and Key Establishment for Ubiquitous Smart Environments. *Proceedings of the International Conference on Computational Science and Its Applications*, Glasgow, United Kingdom, 2006.
[31] R. Panko. *Corporate Computer and Network Security, Second Edition*. Prentice Hall, 2010.
[32] B. B. Bhansali. Man-In-The-Middle Attack – A Brief, 2001, http://www.ouah.org/mitmbrief.htm [accesses September 12th 2010].
[33] Y.-Y. Chan. Weakest Link Attack on Single Sign-On and Its Case in SAML V2.0 Web SSO. *Proceedings of the International Conference on Computational Science and Its Applications*, Glasgow, United Kingdom, 2006.
[34] K.-W. Park, S. S. Lim and K. H. Park. Computationally Efficient PKI-Based Single Sign-On Protocol, PKASSO for Mobile Devices. *IEEE Transactions on Computers*, Volume 57, Issue 6, pp. 821-834, 2008.
[35] L. Hui and S. Ting. A Token-Based Single Sign-On Protocol. *Proceedings of the International Conference on Computational Intelligence and Security*, Xi'an, China, 2005.
[36] D. Boneh and M. Franklin. Identity-Based Encryption from the Weil Pairing. *Proceedings of CRYPTO 2001*, Santa Barbara, California, USA, 2001.
[37] S. S. Al-Riyami and K. G. Paterson. Certificateless Public Key Cryptography. *Proceedings of ASIACRYPT 2003*, Taipei, Taiwan, 2003.
[38] B. Lee, C. Boyd, E. Dawson, K. Kim, J. Yang and S. Yoo. Secure key issuing in ID-based cryptography. *Proceedings of the second workshop on Australasian information security, Data Mining and Web Intelligence, and Software Internationalization*, Dunedin, New Zealand, 2004.
[39] H. Beitollahi and G. Deconinck. FOSeL: Filtering by Helping an Overlay Security Layer to Mitigate DoS Attacks. *Proceedings of the 7th IEEE International Symposium on Network Computing and Applications*, Cambridge, Massachusetts, United States, 2008.

[40] H. Beitollahi and G. Deconinck. Empirical Study of Tolerating Denial-of-Service Attacks with the Fosel Architecture. *Proceedings of the 8th IEEE International Symposium on Network Computing and Applications*, Cambridge, Massachusetts, United States, 2009.
[41] P. B. Tiwari and S. R. Joshi. Single sign-on with one time password. *Proceedings of the 1st Asian Himalayas International Conference on Internet*, Kathmundu, Nepal, 2009.
[42] N. Haller. The S/KEY One-Time Password System. *Proceedings of the Internet Society Symposium on Network and Distributed Systems*, 1994.
[43] A. J. Menezes, P. C. van Oorschot and S. A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 2001.
[44] A. Pashalidis and C. J. Mitchell. Impostor: a single sign-on system for use from untrusted devices. *Proceedings of 2004 IEEE Global Telecommunications Conference*, Dallas, Texas, United States, 2004.
[45] K. Geihs, R. Kalcklösch and A. Grode. Single Sign-On in Service-Oriented Computing. *Proceedings of the 1st International Conference on Service-Oriented Computing*, Trento, Italy, 2003.
[46] T. Fleury, J. Basney and V. Welch. Single sign-on for java web start applications using myproxy. *Proceedings of the 3rd ACM workshop on Secure web services*, Alexandria, Virginia, United States, 2006.
[47] F. Zhu and H. Diao. Single Sign-On Assistant: An Authentication Broker for Web Applications. *Proceedings of the 3rd International Conference on Knowledge Discovery and Data Mining*, Phuket, Thailand, 2010.
[48] Y. Wu, H. Yao and F. Bao. Minimizing SSO Effort in Verifying SSL Anti-phishing Indicators. *Proceedings of the IFIP TC 11 23rd International Information Security Conference*, Milano, Italy, 2008.
[49] V. Samar. Single Sign-On Using Cookies for Web Applications. *Proceedings of the 8th Workshop on Enabling Technologies on Infrastructure for Collaborative Enterprises*, Stanford, California, United States, 1999.
[50] J. Ren. An Identity Based Single-Sign-On Scheme for Computer Networks. *Proceedings of IEEE Workshop on Signal Processing Applications for Public Security and Forensics*, Washington, DC, United States, 2007.
[51] T. Chen, B. B. Zhu, S. Li and X. Cheng. ThresPassport – A Distributed Single Sign-On Service. *Proceedings of the International Conference on Intelligent Computing*, Hefei, China, 2005.
[52] K. Brasee, S. K. Makki and S. Zeadally. A Novel Distributed Authentication Framework for Single Sign-On Services. *Proceedings of the 2008 IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing*, Taichung, Taiwan, 2008.
[53] J. Furukawa, K. Sako and S. Obana. IC card-based single sign-on system that remains secure under card analysis. *Proceedings of the 5th ACM Workshop on Digital identity Management*, Chicago, Illinois, United States, 2009.
[54] M. Liskov, R. L. Rivest and D. Wagner. Tweakable Block Ciphers. *Proceedings of CRYPTO 2002*, Santa Barbara, California, United States, 2002.
[55] R. K. Singh and A. R. Pais. Secure Web Based Single Sign-On (SSO) Framework Using Identity Based Encryption System. *Proceedings of the International Conference on Advances in Recent Technologies in Communication and Computing*, Kottayam, Kerala, India, 2009.
[56] M. Hillenbrand, J. Götze, J. Müller and P. Müller. A single sign-on framework for web-services-based distributed applications. *Proceedings of the 8th International Conference on Telecommunications*, 2005.

[57] J. De Clercq. Single Sign-On Architectures. *Proceedings of the International Conference on Infrastructure Security*, Bristol, United Kingdom, 2002.
[58] W. K. Josephson, E. G. Sirer and F. B. Schneider. Peer-to-Peer Authentication with a Distributed Single Sign-On Service. *Proceedings of the 3rd International Workshop on Peer-to-Peer Systems*, La Jolla, California, United States, 2004.
[59] L. Hui and S. Ting. A Token-Based Single Sign-On Protocol. *Proceedings of the International Conference on Computational Intelligence and Security*, Xi'an, China, 2005.
[60] M. E. Chalandar, P. Darvish and A. M. Rahmani. A centralized cookie-based single sign-on in distributed systems. *Proceedings of the 5th International Conference on Information and Communications Technology*, Cairo, Egypt, 2007.
[61] Y. Ma, X. Chen, L. Li and Y. Luo. P2P-Based Single Sign-On. *Proceedings of IEEE International Conference on Dependable, Autonomic and Secure Computing*, Chengdu, China, 2009.
[62] The Scyther tool. Available at: http://people.inf.ethz.ch/cremersc/scyther/index.html.

Appendix A

Double SSO Sequence Diagrams

In this appendix, we include a sequence diagram for each stage in the Double SSO scheme and in its scaled-down version. These diagrams are aid tools in understanding the textual description of the scheme given in Chapter 5. Since Stage A is Scaled-Down Double SSO is identical to Stage A in Double SSO, its sequence diagram does not appear in this appendix.

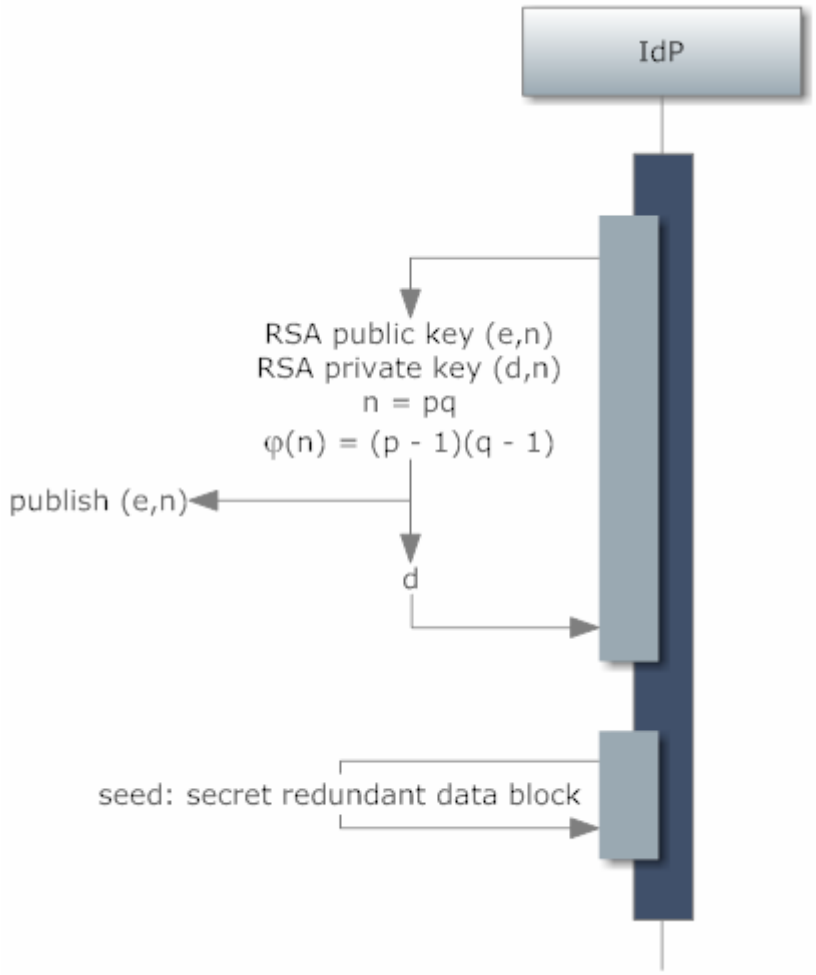


Figure 1. Stage A of Double SSO

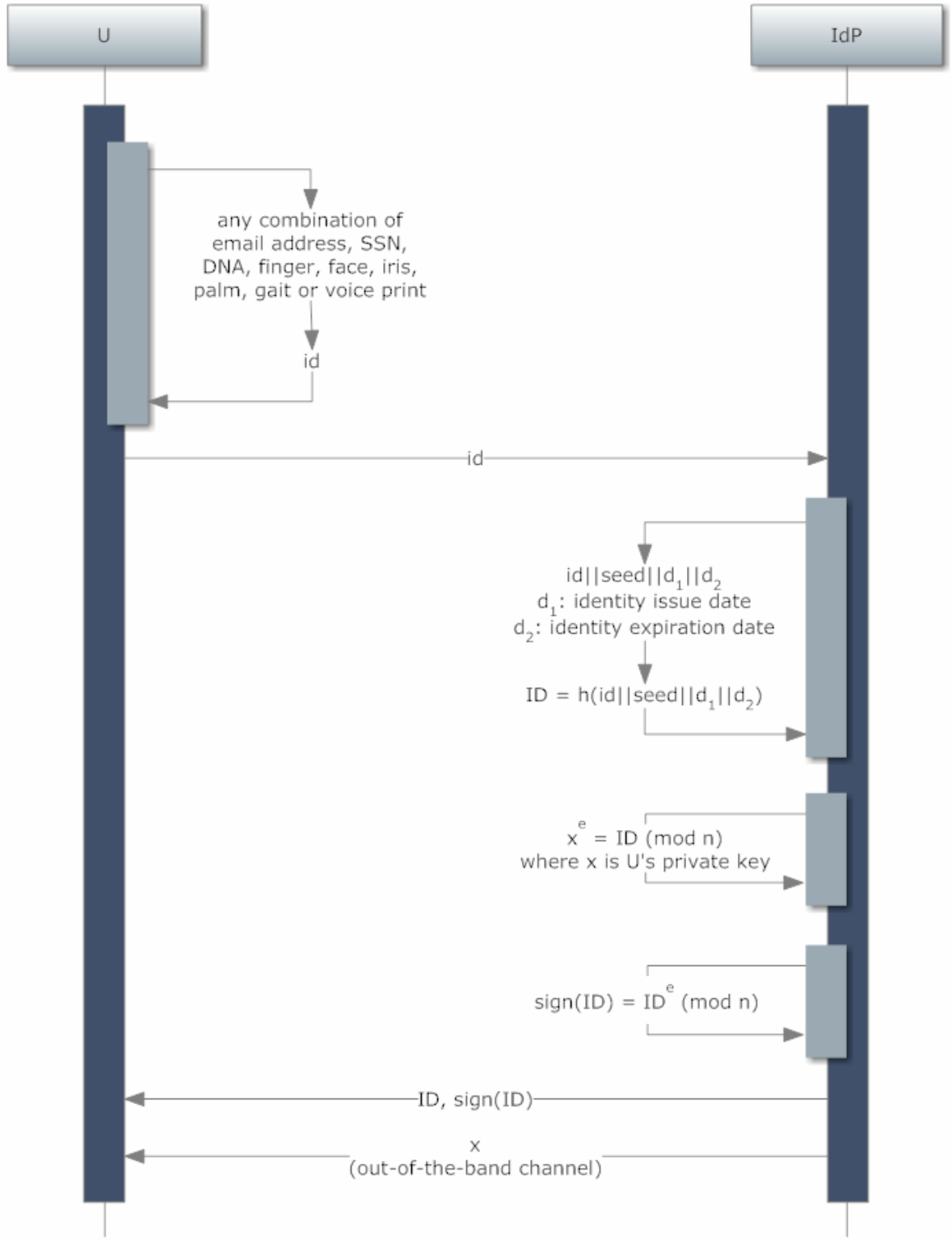


Figure 2. Stage B of Double SSO

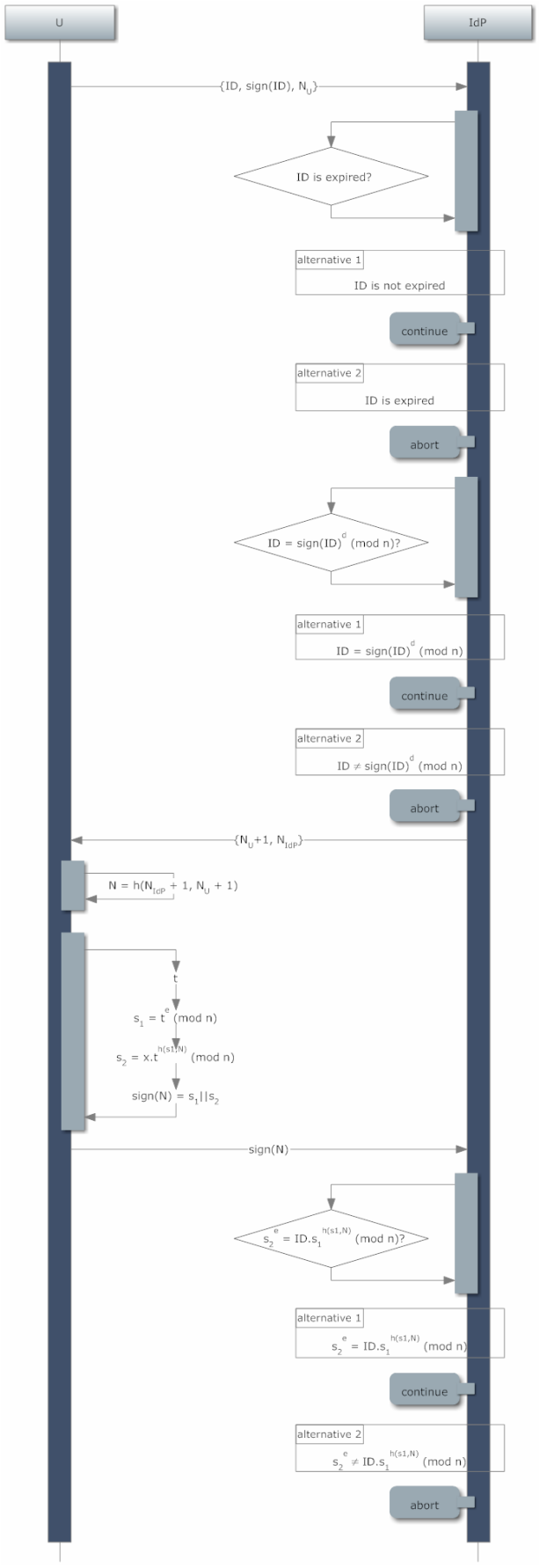


Figure 3. Stage C of Double SSO

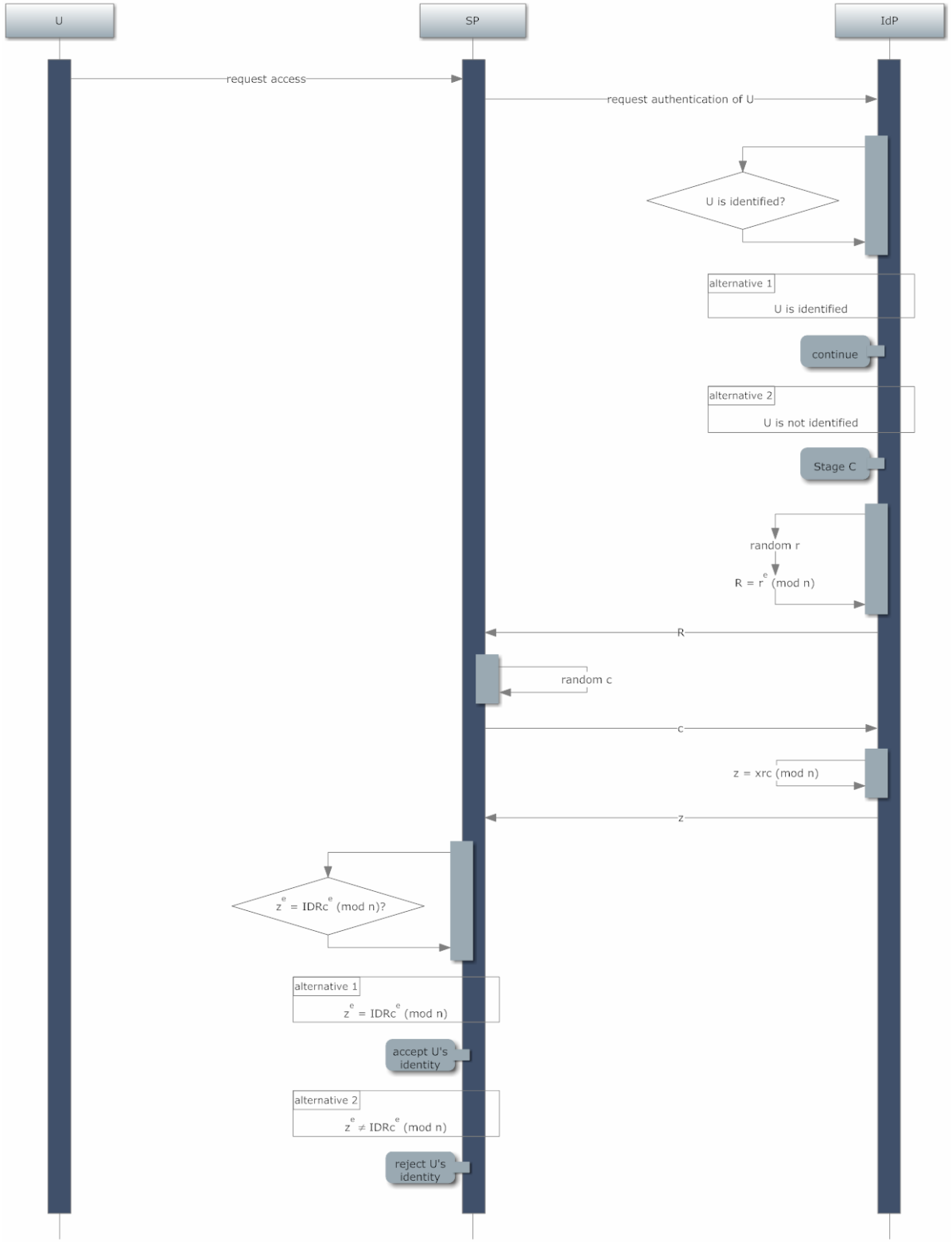


Figure 4. Stage D of Double SSO

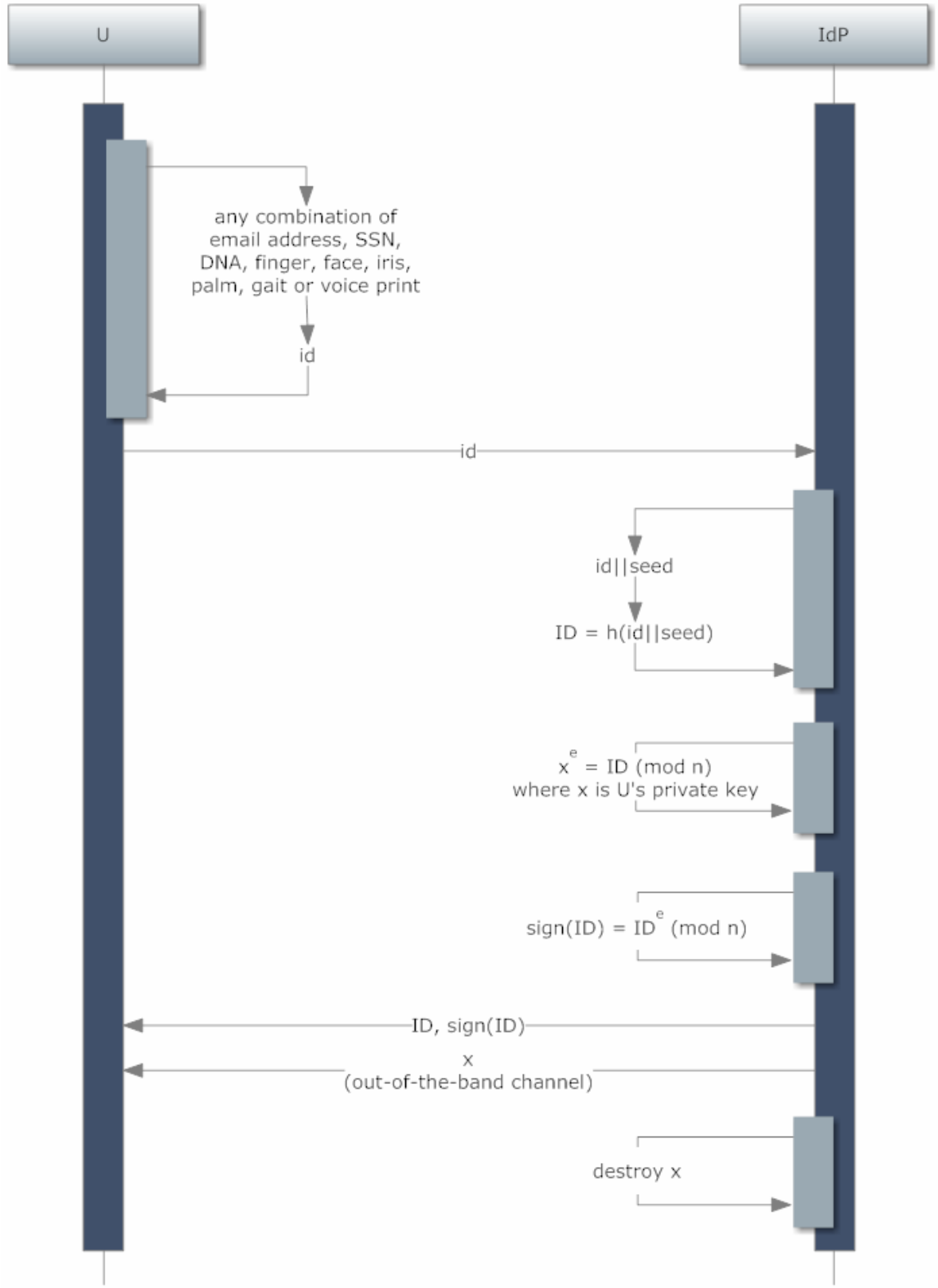


Figure 5. Stage B of Scaled-Down Double SSO

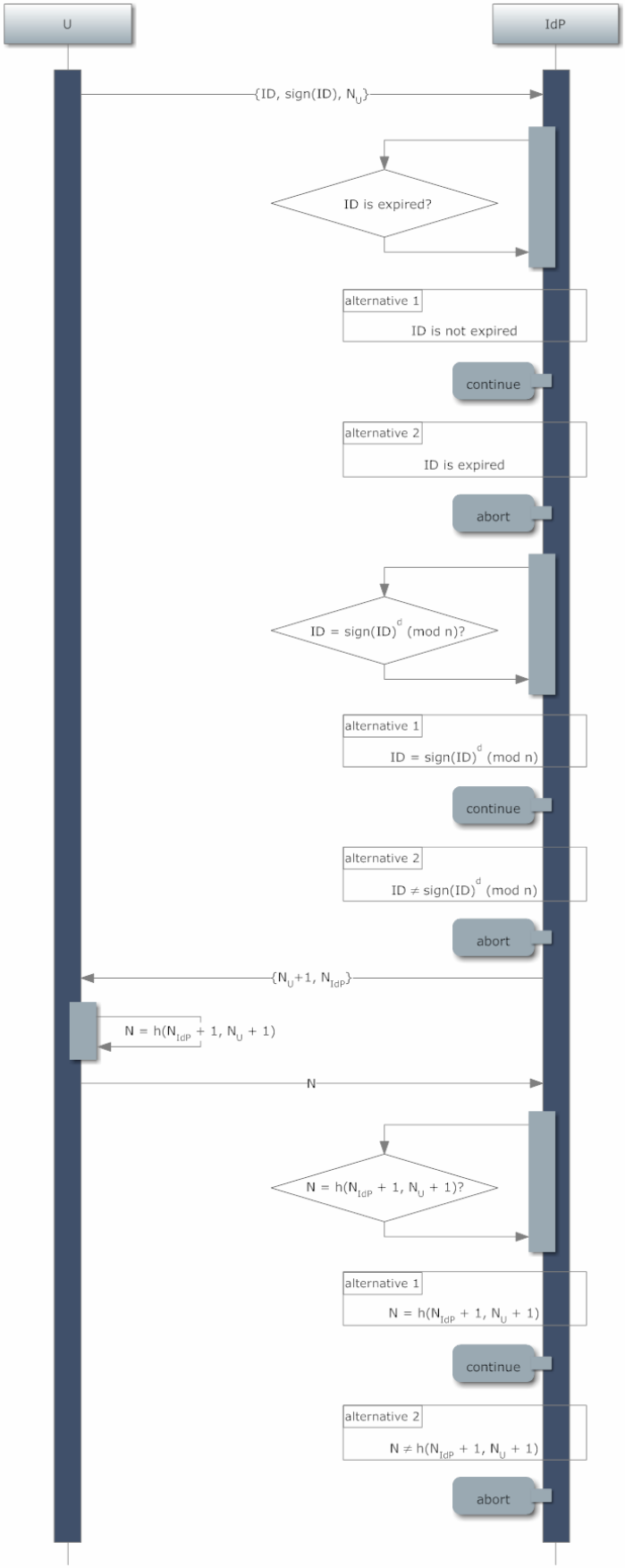


Figure 6. Stage C of Scaled-Down Double SSO

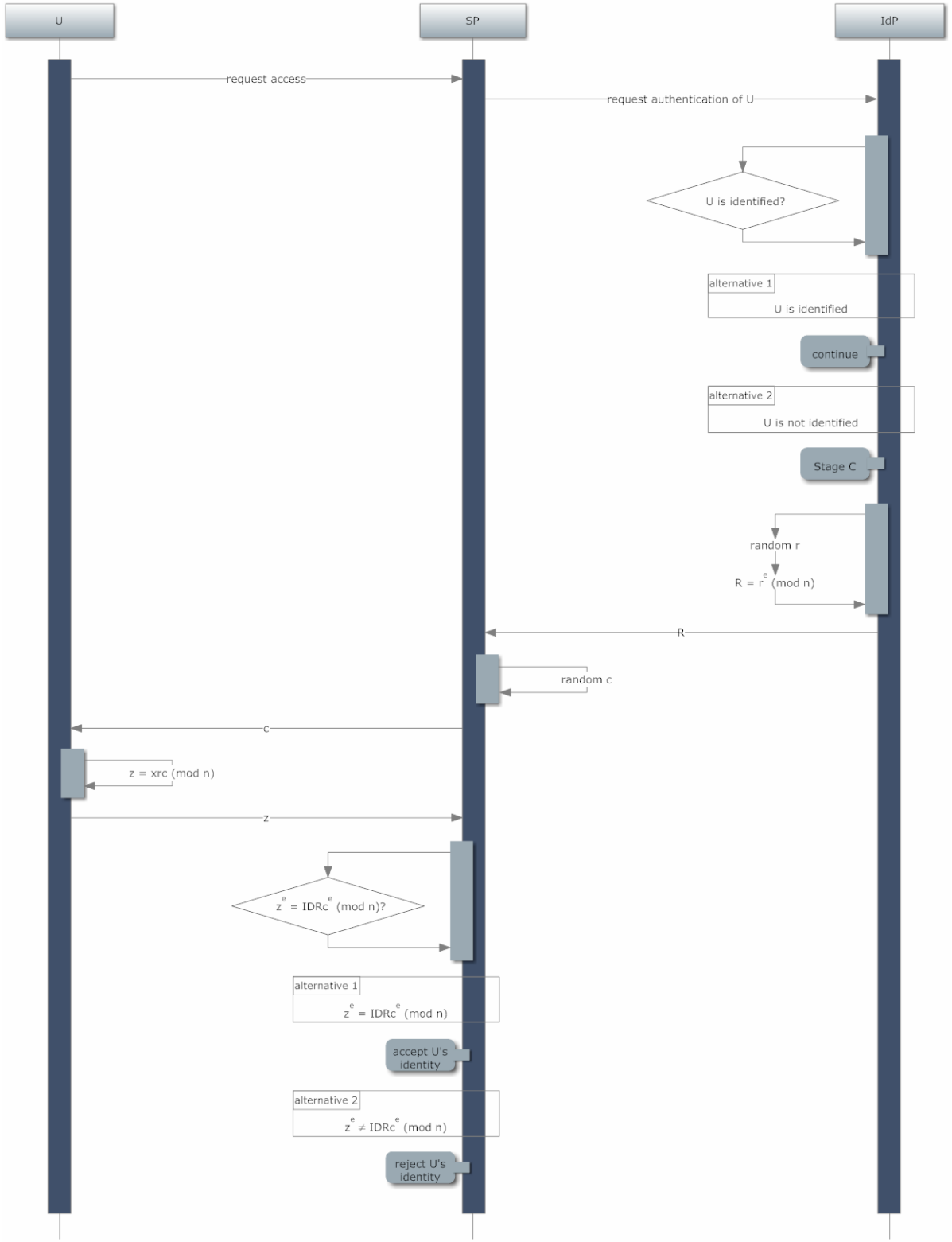


Figure 7. Stage D of Scaled-Down Double SSO