OPENID WITH CERTIFICATE-BASED USER AUTHENTICATION ON SMARTCARD

A THESIS

SUBMITTED TO THE DEPARTMENT OF COMPUTER ENGINEERING

AND THE GRADUATE SCHOOL OF ENGINEERING AND SCIENCE

OF BILKENT UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

MASTER OF SCIENCE

By

Bahar Berna Kişin

May, 2013

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Assist. Prof. Dr. Ali Aydın Selçuk(Advisor)

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Assoc. Prof. Dr. İbrahim Körpeoğlu

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Assoc. Prof. Dr. Mustafa Akgül

Approved for the Graduate School of Engineering and Science:

Prof. Dr. Levent Onural
Director of the Graduate School

ABSTRACT

OPENID WITH CERTIFICATE-BASED USER AUTHENTICATION ON SMARTCARD

Bahar Berna Kişin
M.S. in Computer Engineering
Supervisor: Assist. Prof. Dr. Ali Aydın Selçuk
May, 2013

From the point of its users, federated identity systems provide great convenience to log in to varied web sites without bothering of registration in advance. Looking from a vantage point, federated identity management gives the opportunity to users of one IT system to access data and sources of another IT system seamlessly and securely without handling a complete user administration. Single sign-on mechanisms manage user authentication process of these systems prompting log in once and assure access control across those multiple independent systems. OpenID is a widely used federated identity/single sign-on scheme generally implemented with username-password authentication. In this work, we augment the user authentication phase of OpenID with certificate-based authentication using smartcard technology. Our solution provides a secure method to authenticate the user with user's digital certificate written on the smartcard.

Keywords: OpenID, digital certificate, federated identity, single sign-on, certificate-based user authentication, smartcard, smartcard-based OpenID.

ÖZET

AKILLI KARTTA SERTİFİKA TABANLI KULLANICI KİMLİK DOĞRULAMALI OPENID

Bahar Berna Kişin
Bilgisayar Mühendisliği, Yüksek Lisans
Tez Yöneticisi: Y. Doç Dr. Ali Aydın Selçuk
Mayıs, 2013

Kullanıcılarının gözünde, federe kimlik sistemleri çeşitli web sitelerine önceden kayıt olma derdi olmaksızın giriş yapmada büyük kolaylık sağlıyor. Daha yukarıdan bakarsak, federe kimlik yönetimi bir IT sisteminin kullanıcılarına tam bir kullanıcı bilgi yönetimine ihtiyaç kalmadan diğer IT sisteminin kaynaklarına güvenli ve kesintisiz erişim imkanı verir. Tek oturum açma mekanizmaları bu tür birbirinden bağımsız sistemlerin kullanıcı kimlik doğrulama sürecini giriş yapılmasını tek bir kez isteyip devamında erişime izin verilmesini temin ederek sağlamış olur. OpenID genellikle kullanıcı adı-parola kimlik doğrulamalı olarak sıkça kullanılan federe kimlik/tek oturum açma sistemidir. Bu çalışmada, OpenID'nin kullanıcı kimlik doğrulama kısmını akıllı kart teknolojisi kullanarak sertifika tabanlı kullanıcı kimlik doğrulama ile güçlendiriyoruz. Çözümümüz kullanıcının akıllı kartındaki elektronik sertifikayı kullanarak güvenli bir kimlik doğrulama metodu sağlıyor.

Anahtar sözcükler: OpenID, elektronik sertifika, federe kimlik, tek oturum açma, sertifika tabanlı kullanıcı kimlik doğrulama, akıllı kart, akıllı kart tabanlı OpenID.

Acknowledgement

I would like to express my special gratitude to my supervisor Assist. Prof. Dr. Ali Aydın Selçuk for his precious time to encourage and support me throughout my graduate study, his continued trust in our work and his kindly patience during the supervision of this thesis. I am honoured to have the opportunity to be inspired by his wisdom.

I would like to express my special appreciation to Assoc. Prof. Dr. İbrahim Körpeoğlu and Assoc. Prof. Dr. Mustafa Akgül for showing keen interest to the subject matter, accepting to read and evaluate the thesis and their valuable advices.

I would like to show my deepest thanks and loves to my dad and teacher, Süleyman Akkoç for his far and wide light of Mathematics for my entire life and my mum, Memnune Akkoç for her endless trust and preaching ambition for all my goals and also my brother, Bahadır for being a close friend as well.

I would like to show special thanks to my love Bora Kişin and my family-in-law for being patient and their morale support during my hard work.

I thank Türktrust research people and Dr. Tolga Tüfekçi for their support and motivating.

Finally, thanks to Red Bull, Google and Wikipedia for making this possible...

Bahar Berna Kişin

.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

As human civilization thrives, it has been the major concern that one side needs to confirm identity of other side before presenting their vulnerabilities. Castle gates of middle age are opened to foreigners after seeing an ally flag or banks of early modern age accepts customers with their birth certificates. What changed today? Actually nothing! Today, we are accessing our email, bank or social network accounts via user name and/or passwords which we use to prove that we are the one.

User authentication is a "sine qua non" for IT systems since in a digital environment, there is nothing concrete to ensure your real identity unless you have a secret "something you have, something you know or something you are". Opposite party confirms your identity by your security token-digital certificate-cell phone or password-pass phrase-challenge response or fingerprint-retinal pattern-DNA sequence. By being confirmed, you are allowed to access data, authorized to gain knowledge.

IT systems need to manage identities of their users and there are lots of them! They have to store identity information, have to do some calculations about those

information when the user asks for authorization. Furthermore, they have to do all such things securely; of course if they want to keep pace with the times.

ID management is considered to be centralized when it is within same network, or same domain of control. However, increasingly, users are accessing external systems, out of single domain. ID management is now associated with cross-company, cross-domain, cross-system properties. So, federated (decentralized) ID management rises. Federated identity management links user identity across distinct systems enabling attribute exchange. User authentication part of federated identity systems can be handled by single sign-on (SSO) mechanisms. SSO enables access control of multiple independent IT systems. Login page is prompted once, then user gains access through all the systems related.

OpenID is widely used SSO and federated identity authentication open standard which authenticates users in a decentralized manner. There is no need for services to provide their ad hoc authentication system. OpenID users control their consolidated identifiers and this leads to a user-centric digital identity management.

OpenID does not insist on a specific user authentication scheme. Yet, it is generally managed by user name-password authentication as large percent of IT systems does. However, user name-password authentication has some problematic issues when security notion comes out. Password authentications are open to phishing attacks which lead to identity provider impersonation. Even strong password authentication lets stolen-verifier problem, replay attacks and denial service attacks [1]. In studies [2, 3, 4], offline password guessing attacks are exploited. Users change passwords rarely or choose weak passwords and use the same password everywhere. Moreover, password management is also a fatigue for enterprises. These difficulties address us to augment user authentication of OpenID by a certificate based authentication.

1.2 Scope of the Thesis

This thesis focuses on OpenID scheme with a safe user authentication. In this scope, we offer a user-friendly and secure OpenID solution with certificate-based user authentication. User certificate is stored on a smart card and identity verification is proceeded on it. To meet this objective, we followed the latest OpenID specification [5] and Public-Key Cryptography Standards (PKCS) [6]. As prestudy, we practised previous related OpenID works in the literature, analysed their security and usability. We also surveyed smart card technology, digital certificates and Public Key Infrastructure (PKI) concept.

With our proposed solution we cope with password fatigue of authentication phase of OpenID and enhance user authentication of OpenID with a secure scheme. The solution provides user-friendly and simple user interface of certificate authentication.

Thesis outline is structured as: In Chapter 2, background is given. Federated identity systems and its main approaches are described. OpenID standard basics and its flow are explained; OpenID is examined with its security aspect. Certificate based authentication is declared widely with concepts of PKI, digital certificates, Certificate Authority (CA). A study of smart card technology and known attacks are given briefly. In Chapter 3, related works are explained and their security discussions are given. In Chapter 4, proposed system is described with its architecture, components and operation flow. Chapter 5 collects experimental results of performance and usability. We conclude and mention about future work in Chapter 6.

Chapter 2

Background

This chapter gives background information about the concepts of federated identity, OpenID, certificate-based authentication and smart card technology.

2.1 Federated Identity Systems

Federated identity is an IT system where user identity information is stored at an identity provider (IdP) and accessed by disparate IT systems. Federated identity is combination of standards, system components and concepts for managing user authentication and authorization in a decentralized manner. So, identity and attributes of a person are passed to multiple distinct IT systems securely. Main concept is "trust": the party which provides you an identity and the party which requests your identity must trust each other [7]. Authentication phase is performed by a trustworthy entity named IdP. Opposite entity does not have its ad hoc system for authentication and relies on IdP about user's identity. IdP provides a service to pass identity information to opposite and those two share a secret.

The advantages are: user credentials are safe with IdP, users do not need to serve their identities to every party. Also it is seamless to user since she/he

does not notice authentication procedure. Opposite entity does need to cope with authentication itself and account/identity management costs reduce. Use of SSO decreases login prompts and user information is passed across other entities automatically. It is a good solution for integrating organisations with the help of removal of the need for migrating user credentials. Any user authentication scheme can be used.

However, some disadvantages exist: When IdP is compromised, attacker will access all the parties. Parties need to configure their system to use federated identity, this has a cost of hardware and software. Limited documentation, expertise, guidance struggle application development and corporation.

OpenID, OAuth, security tokens, web service specifications, Windows Identity Framework, claims-based identity are the technologies to enable federated identity systems. OpenID, which is the main topic of this thesis, will be discussed in next section elaborately.

Identity Commons [8] is a non-profit organisation which was founded to support communities working on different aspects of user-centric identity. Some of member groups are Internet Identity Workshop, OpenID Foundation, Information Card Foundation, Higgins Project, Data Portability Project, OSIS Interoperability Efforts, ProjectVRM.

Further information about federated identity technologies can be found in [9].

2.1.1 OAuth

OAuth [10] is an open standard which allows secure authorization. After you are authenticated, authorization takes to the stage and OAuth plays the role. User authentication is out of the scope of OAuth. With OAuth, it is possible to access authorized resources of user by a third party without a need to share user credentials like user name and password. Firstly, creators of OAuth think about implementing authorization on OpenID, but after realizing that there is no standard for access delegation, OAuth emerges to form a standard. Client, server

and resource owner are the parties of OAuth. OAuth is simply summarized as that client asks for owner resources from server. Overall flow is listed in [11] and as follows:

1. User requests a service from client
2. Client redirects user to authorization server
3. Authorization server authenticates user and gets approval
4. Authorization server redirects user back to client with an authorization code
5. Client uses authorization code to request authorization token from authorization server
6. Authorization server validates authorization code and returns authorization token

2.1.2 Using Security Tokens

Using security tokens is the other technology. Information is passed by them. Examples are Simple Web Tokens (SWT), JSON Web Tokens (JWTs) and Security Assertion Markup Language Tokens (SAML). They consist of name/value pairs or JSON objects or are XML-based according to their architecture. We will discuss on SAML briefly, further information about others can be found in [9]. SAML is an XML-based standard for identity information exchange [12]. SAML consists of assertions, protocols, binding and profiles. SAML assertions are used to create identity statements for authentication, attributes and authorization. SAML protocols manage requests and responses for authentication, authorization and attribute queries and their result sets. SAML bindings are used to relate with other protocols such as SOAP. SAML profiles are collection of assertions, protocols and bindings.

2.1.3 Windows Identity Foundation

One other technology is Windows Identity Foundation [13]. It is a framework to create claims-based services for identity delegation. A claim, is a statement put in a token, which is about the user. Issuer issues the token, then user passes token to the opposite party. Opposite party authenticates or authorizes user with those claims.

2.1.4 OpenID and OAuth Hybrid Extension

OpenID is interested in user authentication within federated identity systems. After development of OAuth, the idea of using OpenID and OAuth together allowing a user to share their identity as well as their grant emerges [14]. A small group of Google developers is working on developing an extension to OpenID in order to achieve this. However, at this moment, the project has a draft spec, not a finalized version. The project aims to embed an OAuth approval request into an OpenID authentication request and assertion verification mechanism.

2.1.5 OpenID Connect

OpenID Connect [15] is a framework for identity interactions via REST like APIs. It allows clients to request and receive information about identities and currently authenticated sessions. The specification also allows encryption of identity data, discovery of OpenID provider, and advanced session management, including logout. OpenID Connect performs the same tasks of OpenID but built on OAuth with the extension called OpenID/OAuth hybrid. Its specification is in draft phase, at this moment.

2.1.6 Google+ Sign-in

Google APIs use OAuth 2.0 for authentication and authorization [16]. Google's authentication system provides outsourcing user authentication for other web applications. This eliminates the need to create, maintain and secure a username and password and facilitates to gain access to user profile.

2.1.7 Facebook Connect

Facebook Connect [17] is a component of Facebook Platform which enables users to login with their Facebook identity, friends and privacy to any other site. Facebook connect provides trusted authentication, connecting with real identity including profile information, photos, events and more, friends access besides privacy settings. Neither OAuth nor OpenID provides such a wide attribute exchange rather than offering credentials, hence Facebook Connect has its own structure to access deep information. Nevertheless, the main idea is inspired from both OpenID and OAuth.

Further information on both centralized and federated identity management can be found in [18].

2.2 OpenID

OpenID, is an open standard maintained by OpenID foundation [19], which enables to authenticate user by IT systems without having their ad hoc systems. With the foundation's motto "a safe, faster, and easier way", OpenID enables user to manage their consolidated identities across distinct web sites. OpenID is created for authenticating users in decentralized manner. There is no need for services to provide their own authentication system. Also there is no central authority to register relying parties or IdPs. These two exchange information on OpenID basics. OpenID Provider (OP) is the party who authenticate user and

also an IdP. Relying Party (RP) is the party who asks for user's authentication. Users control their identifier by choosing which OP to use and preserving their different identities. So, a portable, user centric, free and decentralized digital ID emerges.

OpenID is popular amongst web sites so that, according to [20], there are over one billion OpenID enabled user accounts and over 50,000 websites accepting OpenID for logins. OpenID authentication is accepted or provided by many large organisations today. Some of them are Google, Facebook, Yahoo!, Microsoft, AOL, MySpace, Sears, Universal Music Group, France Telecom, Novell, Sun, Telecom Italia and so on [20].

Benefits for users and websites are listed in [21, 22] and as follows:

Fast sign up at websites: Users have single accounts to use at several websites. Not creating new accounts saves time.

Reduce frustration of remembering multiple user names and passwords: Users need to remember only one of them.

Users have control on their identity: Decentralized system has not got a particular, single IdP.

Websites increase registration rates: Fast sign up is liked by users who generally do not like long registration forms

Reduce account storage costs: Websites do not have their authentication systems, so they do not need to recover accounts.

2.2.1 OpenID Protocol Basics

OpenID allows users to use an existing account to sign in to multiple websites, without need for creating new accounts, with SSO facility. ID management issue is decentralized and relying applications do not have their own authentication services. The protocol itself undertakes information exchange issue, messages

are transmitted by HTTP requests and responses [5]. As user identifiers, both URLs and XRIs can be used; it uses Yadis protocol [23] for identity provider discovery [24].

Before getting into the protocol details, it is time to describe the terminology of OpenID:

Identifier: is either a HTTP or HTTPS URI or an XRI [25]. End users and OPs have identifiers.

User-Agent (UA): End user's HTTP/1.1 [26] web browser.

Relying Party (RP): A web application which asks for a proof that end user is the one who submits the identifier.

OpenID Provider (OP): An OpenID authentication server and ID provider of end user on which a RP relies for an assertion that end user is the one who submits the identifier.

OP Endpoint URL: URL to which OpenID protocol messages come, it is obtained by discovery process on user-supplied identifier by RP.

OP Identifier: Identifier of an OP.

User-Supplied Identifier: End user submits this identifier to RP.

Claimed Identifier: End user claims this identifier to own; the protocol aims to verify this claim throughout.

OpenID system has three main components: OpenID provider (OP), OpenID relying party (RP) and user agent (UA). In Figure 2.1 below, an overview of OpenID is demonstrated. OP is the identity provider and performs authentication of end user. RP is an OpenID enabled application that uses OpenID protocol to have the end user authenticated. This separation of the protocol enables end users to choose the OP to manage their identity. End users create their accounts once on OP and use it to access several RPs through their UA (here web browser, generally).

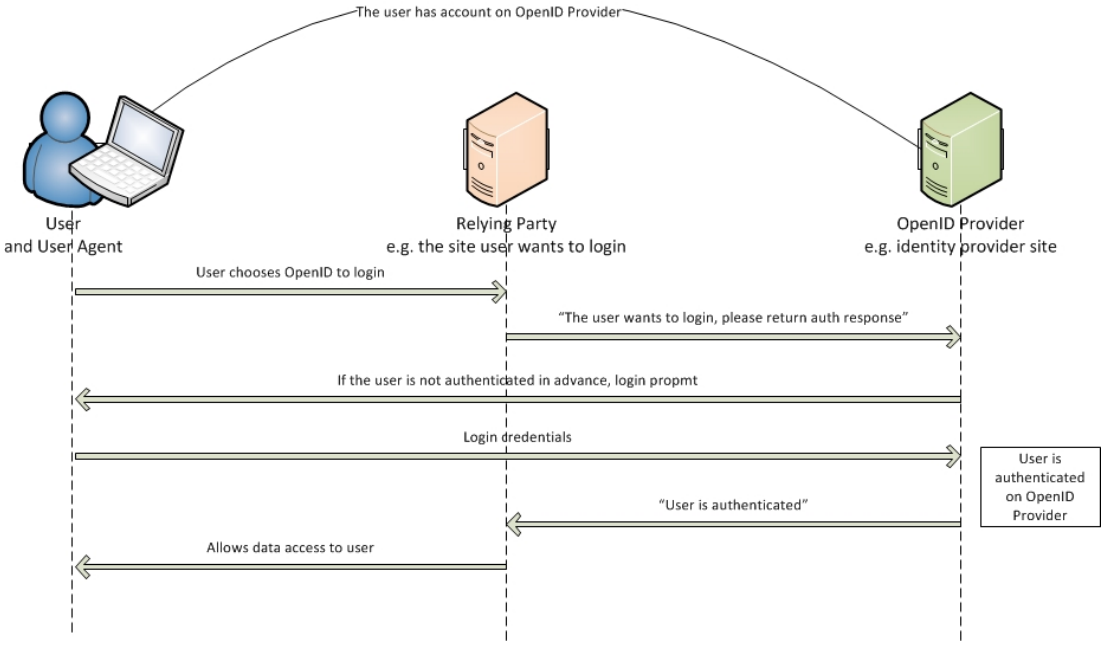


Figure 2.1: OpenID overview

2.2.1.1 Main Layers of OpenID

Now, let's look at the four main layers [24, 5] of OpenID Protocol:

Identifiers: End users control their identities via *identifiers* which have following architecture:

Address-based identity: uses a unique digital address to identify user. This digital address is normalized to discover and launch authentication process.

Card-based identity: uses a digital token that contains attributes which identifies user. This is approach is implemented by technologies such as Microsoft's CardSpace [27] and Higgins Project [28]. As an identification method for OpenID, CardSpace and detailed information is given in [18].

OpenID resolves addresses according to these standards:

- An OpenID-enabled URL like `http://example.com/user`.

- An XRI formed i-name like `xri://=example.user`

Discovery: After RP obtains OpenID digital address, *discovery* is performed by RP to reach user's identity provider. Yadis discovery protocol or XRI resolution protocol are used to resolve URL or XRI respectively. Both of them need Extensible Resource Description Sequence (XRDS) format by OASIS XRI Technical Committee. XRDS is XML-based document which stores OP information. The process is like: an XRDS document is retrieved via HTTP(S). This XRDS document is resolved for obtaining service addresses. If Yadis protocol does not work on URL, HTML-based discovery can be attempted by looking for information in the header of HTML page.

Authentication: OpenID *authentication* is the process which proves users themselves control the identifier they submit. It consists of several cryptographic proof functions and secure communication. These are creation of a shared secret with Diffie-Hellman key agreement between OP and RP. This is called *association* and used for verifying protocol messages. Also, authentication requests for *assertions*, verifying this assertion signatures (with HMAC-SHA1 or HMAC-SHA256 algorithms), URLs, discovered informations and authentication responses are parts of authentication.

Data Transport: OpenID *data transport* provides a robust data exchange between RP and OP. Protocol messages are mapping of keys to values which permits Unicode character set. All the messages are sent as HTTP requests (GET or POST). Communication types are direct (with HTTP POST) and indirect (with HTTP redirects and HTML form submission). Data transport enables exploiting secure messaging, data synchronization.

2.2.1.2 Operational Flow of OpenID:

After giving main concepts above, we will look into operational flow of OpenID. User has an $identifier$ from an OpenID provider, in advance. With this identifier, user wants to login a RP. But RP will require an *assertion* about user's identifier.

Assertion will be verified by OP. In Figure 2.2 below, protocol flow of OpenID is shown.

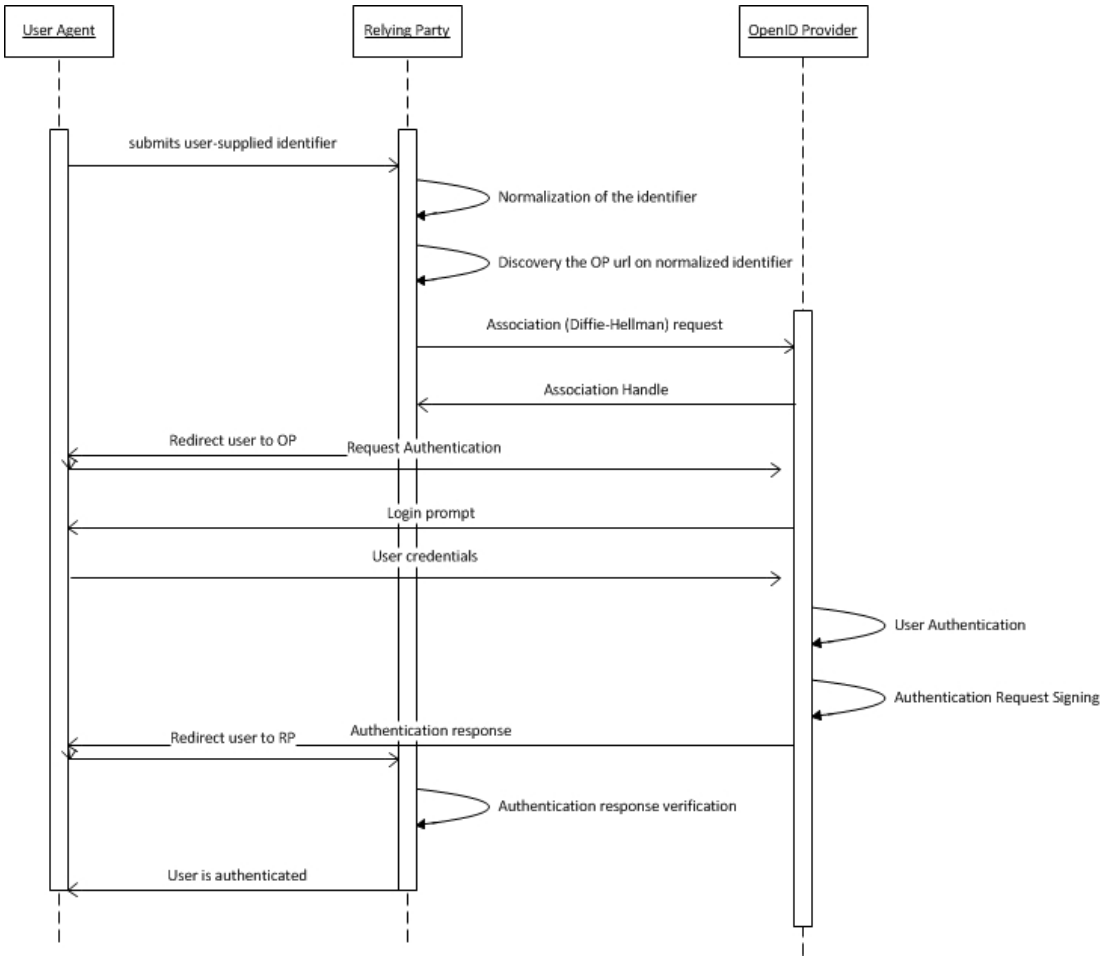


Figure 2.2: OpenID operational flow

1. User visits RP page with UA and chooses logging with OpenID.
2. User submits his $user - supplied\ identifier$ to RP to start process.
3. RP performs *normalization* on user input to transform it into an $identifier$
4. RP performs *discovery* on the identifier using XRI-Resolution or Yadis protocol to retrieve $OP\ endpoint\ URL$, $claimed\ identifier$, protocol version. In case of that user-supplied identifier is supplied by an $OP\ identifier$, claimed identifier can be null.

5. RP and OP generate a shared key during *association* using Diffie-Hellman (DH) key exchange protocol. This key is used in signing process of protocol messages between OP and RP. Also signature on messages are verified by this key. By this process, association is established between OP and RP.

6. RP redirects UA to OP with a redirection message consisting of OP endpoint URL, $return-to-URL$ (url of RP to go back after user authentication), DH association handle and a request to identify user who submits the identifier.

7. OP authenticates user using any authentication scheme.

8. If authentication succeeds, OP redirects UA back to RP with an *assertion* message which proves user is the one.

9. RP verifies OP returned assertion using shared key or asks OP with a request

10. User now, can access RP website.

2.2.2 Security Analysis of OpenID

Like every protocol, OpenID has got some security weakness, too. In OpenID specification [5], some security considerations are discussed. They suggest a set of preventions to handle these weakness. User agents are another weak points of the protocol if they are infected. Like many protocol, OpenID needs to rely on security of user agents. Nevertheless, we will discuss about following security issues:

2.2.2.1 Eavesdropping attacks

In the specification, they mention about preventing an interception during authentication assertion by checking nonce every time. Yet, this does not work for eavesdropping itself. OpenID protocol does not use any encryption, so an

eavesdropper can track messages and gain information. Transport Layer Security (TLS) should be used over OpenID.

2.2.2.2 Integrity Protection

OpenID messages are not integrity protected by default. Assertion response message is the single one. So, an attacker can change every message except it. For example, attacker changes user-supplier identifier with user's another identifier; gains user credentials if they are same.

2.2.2.3 Man-in-the-middle attacks

After association established, tampering of data could be prevented. Before association, it could be also hard to change signed messages with usage of MAC in condition that MAC key is perfect random.

OPs can be impersonated after DNS resolution or transport layer compromise. Also, altering XRDS document which is not integrity protected may lead to redirection to any other OP that attacker chooses. This can be prevented by signing the document by XMLDSIG.

One another solution to prevent man-in-the-middle attacks is using SSL in all communications. The specification also recommends it, yet this may be expensive.

RP may also be a man-in-the-middle. Discovery process enables data collection on several parties. Furthermore, RP may redirect UA to itself in order to gain information.

2.2.2.4 Phishing attacks

Again a malicious RP redirects UA to a fake OP to collect user credentials by exploiting phishing attack.

User may be mislead to a fake RP and then user submits his identifier to fake site, eventually logs in this site.

2.2.2.5 Denial of Service attacks

According to the specification, OpenID is open to DoS attacks at protocol level, since it does not check reality of requests. However it relies on IP based banning techniques where servers run.

On the other hand, in [29], a security evaluation of OpenID protocol is given. The study analyses OpenID security with protocol analyser tool AVISPA. The study shows that if DNS or routing infrastructure is attacked, an intruder may be authenticated to RP without being noticed

2.3 Certificate Based Authentication

In means of authentication, passwords and pass-phrases are considered, in some degree, weak since they are not generally chosen strong enough. Authentication based on cryptographic functions can indicate an evidence that user owns attributes he presents. One such technology is public key cryptography proposed by Diffie-Hellman firstly. In time, this structure has been developed; Public Key Infrastructures (PKI) and use of public key certificates have emerged. Main concepts and detailed information about PKI can be accessed in [30].

2.3.1 Certificate Authentication

In certificate based authentication, user is assumed to have a valid certificate which is used to identify the user. Instead of passing password and user name data, user certificate is used in authentication. Authentication phases are like below and demonstrated in Figure 2.3:

- Data X is a challenge string with specific content and format.
- Private key for user's certificate is used to digitally sign data X (or its hash).
- This data X (or its hash) and digital signature together constitute *evidence*.
- The digital signature can be verified with the corresponding public key.
- In order to authenticate user, the user's certificate, the evidence, the signed data $S(X)$ is needed.
- User's certificate chain is verified firstly. Then digital signature is processed with user's public key, result is compared with data X (or its hash) from evidence to complete verification.

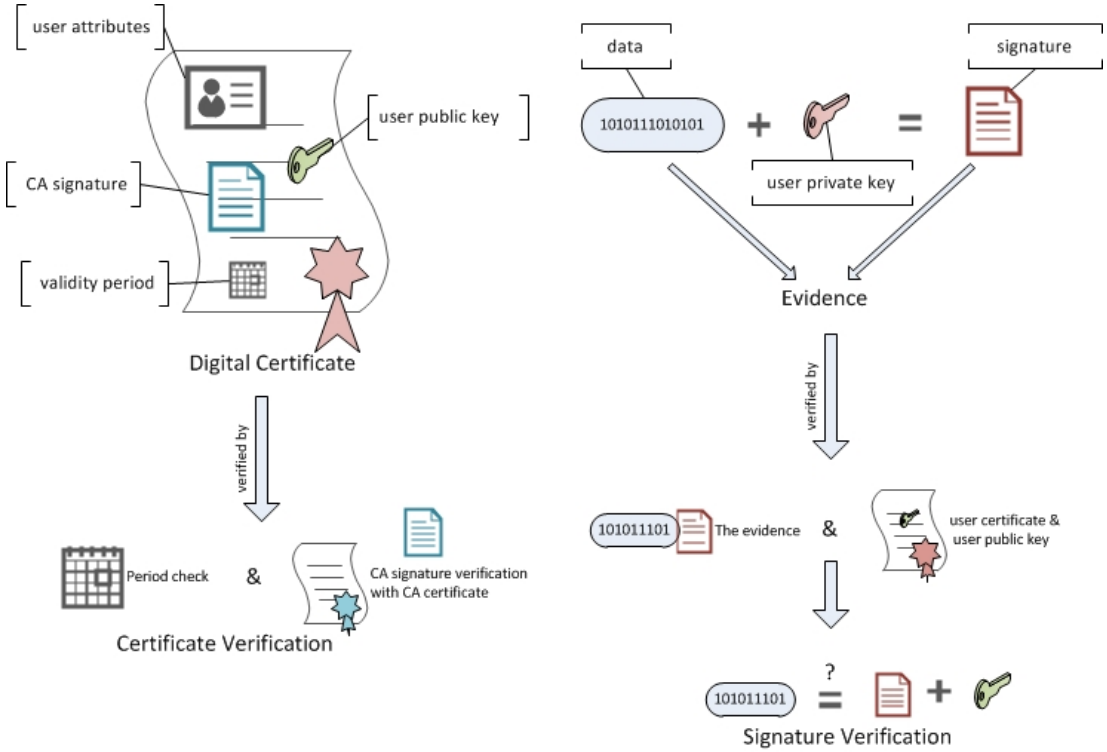


Figure 2.3: Certificate authentication

2.3.2 Digital Certificate

A digital certificate is an electronic identity which defines a person, an organisation or a machine. The entity attributes, the entity's public key and a CA's signature constitute the digital certificate. X.509 [31] is an ITU-T standard for a PKI to format digital certificates. The contents of an X.509 certificate are:

Serial Number: Unique identifier of the certificate.

Subject: The entity identified.

Signature Algorithm: The algorithm used to form the signature.

Signature: The actual signature.

Issuer: The entity which issued the certificate (e.g. CA).

Valid-From: The start time of the certificate validity.

Valid-To: The expiration date of the certificate.

Key-Usage: Purpose of public key (e.g. signature, certificate signing, etc.).

Public Key: The public key or its hash.

Thumbprint Algorithm: The algorithm used to hash the certificate.

Thumbprint: The hash of the certificate itself.

2.3.3 Certificate Authority

Certificate Authorities (CA) are organizations which issue certificates and provide services to validate certificates. They are trusted organizations resulting from the legal policies assigned to their services. This trust gives a reliance to someone who wants to trust the certificate owner. Example CAs are VeriSign, Thawte, Tubitak, Türktrust etc.

CAs have their root certificates which issue themselves and the other digital entities are issued with these root certificates or their sub roots. The root-sub root-certificate signing sequence is named certificate chain. Before using the certificate, certificate chain must be verified as follows:

- The certificate validity period is checked
- The certificate signature is verified using the public key in the issuer certificate.

After giving some basics detail, let's discuss:

Today, use of cryptographic material is increasing rapidly. According to Cryptography in the Enterprise Survey 2005 by nCipher, enterprises need better key management due to the increment of cryptographic solutions. Another survey of Symantec, 2011 Enterprise Encryption Trends Survey, shows that in spite of the increase in using encryption frequently, enterprises do not have the ability to manage encryption keys properly.

In [32], a comparison of the authentication and authorization infrastructures is given. In the same study, at least two reasons are stated to take caution in X.509-v3 certificates: the different departments of the same organisation may not have the equal competence and so, validity period of the certificate may be long for the assignment of the person.

In [33], it is underlined that the importance of the digital certificates is underestimated and also actions to minimize the risks are suggested.

In [34], it is stated that X.509 certificates, like many ID-cards, reveal too many information than necessary and also information-parsimonious authentication techniques are explained. Again in [35], a design for digital certificates which preserves privacy without decreasing security is proposed.

2.4 Smart Card Technology

Smart card is an integrated circuit card which provides secure solutions for data storage, identification, authentication and processing [36]. They have standardized dimension and made up of plastic. User access to card is possible with a PIN code. They have a microcontroller containing a CPU and memories RAM, ROM, EEPROM [37]. They have a tamper-resistant secure processor and a file system. To change data on its memory, a programmable security code (PSC) must be verified before writing on the card. Communication is possible with card-reading devices supporting them. They are separated into two: contact and contactless. Contactless one uses RF-induction technology in communication [38] and are not subject of this thesis. Contact smart cards have a contact area made up of golden plates which enable communication via a card reader. Application Protocol Data Unit (APDU) [39] commands are sent to communicate with cards. Applications can be deployed on cards for secure processing purpose. Cards do no need a power supply, use energy through card reader.

ISO/IEC 7810 and ISO/IEC 7816 standards define how they work, physical and electrical characteristics, communication protocols, commands.

2.4.1 Usages of Smart Cards

Smart cards are used in wide area, from basic storages to secure applications.

Identification: Smart cards are used for authentication, commonly with PKI. A digital certificate is stored on card. Access and communication with the cryptographic smart cards is done with a PKCS#11 library. These cards have file system, based on the PKCS#15 standard which allows cryptographic tokens to identify themselves to applications.

Financial: Smart cards are used as credit, ATM or cell phone SIM, public transport cards.

Tracking: Smart cards are used in schools, dormitories, hospitals for tracking attendance, food services and entrance control or payment per item purposes.

System security: Smart cards are used for storing keys, certificates, secure elements and secure processing [40].

2.4.2 Security of Smart Cards

If smart card user's computer hosts malware, smart card security may be broken. Man-in-the-browser malware could modify a transaction via keyboard and application screen, unnoticed by user.

Some other attacks perform extensive physical manipulation and destruction of card electronics or exploit weaknesses in card design logically to expose private keys and manipulate secure data [41]. For example, rewiring a circuit on chip adding or cutting tracks, exposing heat, UV radiation, or x-ray [42]. Some known invasive and non-invasive attacks are collected in [43] and as follows:

De-packaging: removing circuit from plastic card by dissolvers, knives

Layout Reconstruction: an invasive attack to create a map of processor, bus lines, module boundaries, chip architecture.

Manual Micro-probing: attacker establishes electrical contact with on-chip bus lines

Using Beam Technologies: exploiting from Ion beams, electron beams, infrared laser

Key and Memory Reading: micro-probing, optical reconstruction, software abusing are used to observe entire bus and record values in memory.

Key Retrieval with Overwriting: attacker modifies ROM and EEPROM content using parity checks or bit by bit changing to extract the key.

EEPROM Altering: information can be trapped by raising or dropping supplied voltage to the micro-controller or UV light to erase lock bit.

Timing Analysis Attacks: are based on measuring time taken for a unit to perform operations, which leads to gain information about key.

Power Consumption Attacks: using a resistor in power supply, fluctuations can be measured in current consumed by card to track operations.

Simple Power Analysis (SPA): is a technique that involves directly interpreting power consumption measurements collected during cryptographic operations.

Differential Power Analysis (DPA): exploits statistical functions tailored to target algorithm using correlated date due to instruction sequence.

Differential Fault Analysis (DFA): exploits erroneous output under stress and comparing these outputs to extract processors elements, including keys.

Glitch Attacks: attacker generates a malfunction that causes one or more flip-flops to adopt wrong state and replaces a critical instruction.

Chapter 3

Related Work

In this chapter, we will look into related works about OpenID and smart cards, OpenID and EAP-SIM, OpenID and GBA and some other schemes of federated identity.

3.1 OpenID and Smart Card

3.1.1 Smart OpenID

In [44], Leicher et al. present an enhancement to OpenID protocol with use of smart card for existing security infrastructure of mobile network operators (MNO). They also implemented an ad-hoc Android application. To achieve this, they added a new concept: *Local OP* which mainly controls user authentication and issues OpenID assertions. It runs on smart card and has UICC capabilities which enables secret storing and performing cryptographic operations. Local OP cannot be reached by RPs due to mobile network restrictions. So, they added a new concept, *OP Support Function* (OPSF) which is operated by MNO. It is accessible by public internet. To communicate with local OP, it shares a secret S which is stored on Universal Integrated Circuit Card (UICC). S is either preinstalled on smart card or gained with Over-the-air programming (OTA)

capabilities of the MNO. So, general overflow is as follows:

1. RP performs discovery and association on OPSF.
2. OPSF looks up S, creates association handle: $asc - hdl$, creates $S_a = KDF(S, asc - hdl)$ for OpenID session and returns $asc - hdl$ and S_a to RP
3. RP encrypts S_a with DH secret established between RP and OPSF and redirects user to local OP
4. Local OP authenticates user by asking PIN, derives S_a with same KDF using S and $asc - hdl$ from redirect request, calculates assertion signature using S_a and redirects the browser to RP with a message consists of OpenID parameters and signed assertion
5. RP verifies assertion with S_a

In this work, they implemented local OP as an Android application on a Nexus One phone. For OPSF and local OP they used Python based OpenID libraries and SL4A on an Android phone. The demo was software based and did not yet include access to UICC.

To discuss, this solution augments OpenID by device based PIN code login without sending user credentials over the internet. The system reduces phishing, man-in-the-middle and silent logon based attacks. Also using anti-temper cards increases security by decreasing anxiety of altering data stored on card. The two-factor authentication based on both device and network provides a more secure solution. Known attacks on smart cards described in Section 2.4 must be considered. Especially, it is possible to insert malicious code into card when card security code is known. The overall protocol does not use a secure tunnel, the system is open to eavesdropping.

3.1.2 OpenID with SSL Smart Cards

In [45], P. Urien studied a strong authentication for OpenID using SSL smart cards. In this study, smart card is in USB dongle with flash memory and card reader. Proxy software is in an USB dongle. User authentication is done with SSL session performing mutual authentication and all SSL functions are handled by smart card. URLs are pushed to smart card via ISO 7816 commands and response to it returns in XML format. In authentication, RSA key pairs and X.509 certificates are used. Overall flow is as follows:

1. Browser goes to RP and starts HTTP session with user's OP identifier.
2. RP performs XRI discovery and establishes association to OP and sends authentication request to OP. Until now, it is the standard procedure.
3. Then, OP shows its login page which uses proxy software on the browser. In HTTP header of page, a cookie is placed.
4. When user clicks the button, she is authenticated by her X.509 certificate and the cookie is set to proxy address.
5. Second exchange between OP and browser, is verification of association.
6. Then OP returns authentication response and redirects to RP, as usual.

To discuss, use of user name/password is removed from OpenID, this provides a more secure authentication. Besides mutual authentication with SSL certificates is performed. All the authentication process is handled by USB dongle apart from OP. It also enhances security, since the dongle is tamper-resistant and far from network communication itself. Known attacks on smart cards described in Section 2.4 must be considered.

3.1.3 OpenID for USIM Cards

In [46], P. Urien proposes an OpenID system using a 3G dongle equipped with Universal Subscriber Identity Module (USIM) for mobile operators. The system merges PKI (with SSL) and Authentication and Key Agreement (AKA) which is worked by 3G/4G mobile operators, on a USIM smart card. This USIM smart card (ETSI TS 102.221 standard) stores International Mobile Subscriber Identity (IMSI) and computes AKA. In the system, High-Speed Downlink Packet Access (HSDPA) modems are used for authentication to login radio 3G network and they are equipped with USIM module containing SSL stack (with RSA X.509 certificates). The smart card runs both USIM application and TLS application. So, overall flow is as follows:

1. User plugs 3G dongle and registers SIM on a 3G network with AKA authentication and then runs proxy software on the USB dongle.
2. Terminal equipment (PC, USIM module, HSPDA modem) performs discovery from the URL.
3. The proxy software starts a TLS session between USIM and OP.

To discuss, this system shows deployment of 3G services with SSL enabled services, as an example of SSL enabled OpenID access. Considerations of the previous related work is still valid for this study too. This study adds mobile authentication concept to OpenID. So the system relies on security concepts of AKA and IMSI.

3.2 OpenID and EAP-SIM

In [47], a service is proposed where mobile operator is not only an OpenID Id provider but also a SIM authenticator as a part of EUREKA Mobicome project. This service merges the mobile network and OpenID with (U)SIM cards. SIM

authentication refers to use of authentication mechanisms of GSM (e.g. A3 algorithm) for authentication of internet-based services via SIM cards. Authentication messages of GSM are transported via IP instead of mobile wireless infrastructure. The system components are: a Java applet which communicates SIM with USB using APDU and communicates an authenticator with Extensible Authentication Protocol-SIM (EAP-SIM) using HTTP; an authenticator which communicates an authentication, authorization, and accounting server(AAA); a AAA server which communicates SS7 network; a RP and an OP. So, overall flow is as follows:

1. User enters identifier, RP discovers and redirects to OP, OP starts user authentication, as usual.
2. The user authentication is done with EAP-SIM performed between USIM module and authenticator.
3. Then OP responses authentication and redirects to RP, RP allows access, as usual.

The EAP-SIM authentication: is for mutual authenticating a user device (GSM SIM card) to a network with a session key agreement. EAP-SIM frames are packed into RADIUS packets at access point (IEEE 802.1x a). RADIUS server uses GSM triplets to challenge. Triplets are provided by an Home Location Register (HLR) or Authentication Center (AuC) through SS7 network [48]. EAP-SIM is a challenge-response protocol based on Extensible Authentication Protocol (EAP). EAP is a protocol which defines message formats for transmitting over datalink layer; is an authentication framework for transporting authentication protocols, not a wire protocol itself.

To discuss, user authentication is performed by EAP-SIM using SIM cards. So the system relies on security concepts of EAP-SIM. Furthermore, authentication is handled by a Java Applet in web browser. Applet code must be signed with code signing certificate. SSL/TLS should be used to prevent eavesdropping, since no secure tunnelling is used.

3.3 OpenID and GBA

In [49], a user-friendly mobile OpenID solution integrated with Generic Bootstrapping Architecture (GBA) is presented. GBA extends secure infrastructure of cellular technology with the Internet infrastructure and is standardized by 3rd Generation Partnership Project (3GPP). In fact, 3GPP also specifies a solution for OpenID with GBA [50]. The study follows it.

Generic Bootstrapping Architecture: generates a temporary shared key between user device and mobile network operator. UICC and Home Subscriber Server (HSS) share a master shared key, a sequence number, and an IMSI. HTTP Digest Authentication and HTTP Digest AKA is used.

The system components are: a HSS which provides subscriber identity; a Bootstrapping Server Function (BSF) which handles bootstrapping process; a Network Application Function/OP (NAF-OP) which is an application server supporting GBA based user authentication and also an OP server; a User Equipment (UE) which is a user agent owning a USIM and a GBA client; a RP. So, overall flow is as follows:

1. User submits an identifier to RP.
2. RP performs discovery, establishes DH association and redirects user agent to NAF-OP.
3. Between NAF-OP and UE, HTTP digest authentication occurs.
4. Between UE and BSF, HTTP Digest AKA occurs.
5. UE returns HTTP digest response to NAF-OP.
6. NAF-OP requests the same digest response from BSF to verify, then redirects to RP.
7. RP allows access.

To discuss, HTTP is an insecure channel, so an eavesdropper can listen or man-in-the-middle attack may occur. However, the protocol is controlled by random nonce and encrypted sequence number, also by protection parameter message integrity is controlled. Also, an attacker cannot derive session key which is based on several parameters and cryptographic hash functions. However the protocol uses MD5 which has known weakness (chosen plain text attack, hash collusion attack). By overall, a secure tunnel (SSL/TLS) should be used.

3.4 Others

In [51], they propose a privacy friendly user-centric federated identity management scheme with trusted secure elements like smart cards or an embedded module. The main concepts are combination of PKI and federated ID systems. They are: trusted module, service provider, identity provider, revalidation authority, certification authority, audit authority, card issuer. Service providers and identity providers generate key pair. Certificate authority certifies public key in a certificate and this certificate also includes access rights. The authentication is performed by certificate validation and signature verification. Smart card and user mutually authenticate each other. Privacy is handled by separation of ID management issue amongst different parties.

In [52], they propose a smart card based single sign-on solution. The solution is integrated with a browser extension. Main concept is that account information is stored on file system of the smart card separately; when user wants to login, authentication manager retrieves user name and password information from card and submits to web site. An account manager is implemented to add or remove user accounts. The solution contains no certificate or signature authentication. Encryption is used only in backup the card information. Communication is not through a secure channel (e.g. SSL/TLS) either.

In the study about OAuth, CardSpace, SAML combination [53], they propose a solution to integrate OAuth and an Information Card system like CardSpace or

Higgins. With the solution, Information Card users can obtain a security token from an OAuth-enabled system. They implemented a browser extension. Overall flow is as follows:

1. User visits a CardSpace-enabled RP page.
2. User chooses a personal card from selector.
3. Selector sends a Request Security Token (RST) to IdP, which responds with a Request Security Token Response (RSTR).
4. The extension prompts user to enter IdP URL and RP identifier.
5. Browser extension uses RSTR to construct an OAuth request to forward to the IdP
6. IdP validates the received request, verifies client identity by comparing the previously registered, authenticates user over a TLS-protected channel.
7. IdP redirects user to RP.
8. Browser extension constructs a 'CardSpace-like' SAML token to submit RP
9. RP verifies SAML token (response signature, nonce, time-stamp verification)

Chapter 4

Proposed System

As we mentioned previously, OpenID does not specify a particular user authentication scheme. However, the most used one is user name/password authentication. In addition to known weak issues of user name/password scheme [54, 55], organisations don't seem to be worried about weakness of passwords. According to a Symantec Research in [56], 27% of corporates use simple user name and passwords, 30% plus an additional factor and 43% uses additional factor when using VPN only. Also the same research says that 67% of corporations do not require strong or two-factor authentication from partners to access their networks.

As speaking about OpenID, widely used user name/password mechanisms bring weakness with them. User submits user credentials; yet it is not secure enough, they can be revealed especially in a system where a single account is used to access multiple parties. Also, user accounts should not be both stored and authenticated in a single provider itself in a federated identity system.

Our proposed system, uses a secure user authentication scheme added to OpenID with a security token: a smart card containing several certificates of the user. With this solution, OP does not need to store user information or credentials, hence gets rid of account keeping/recovery costs. Users get their identity (certificates) from a CA, OP only authenticates the user via user certificate. Therefore, the system proposes a more decentralized system of identity

management than classical OpenID-enabled system. System can verify any certificate obtained from any issuer, the certificate authentication scheme is not specific to this solution. This system provides a ready plug and play OpenID solution for certificate authorities. A Java applet launching from OP web site, runs on user browser, so user authentication phase is handled locally with smart card communication on user agent, not on OP server physically. Users also have the ability to choose identity to login with. Proposed system facilitates account management from the aspect of user with a certificate selection mechanism.

In this chapter, we present a scheme of smart card based user authentication with digital certificates, on OpenID protocol. System architecture with components and operational flow, deployment environment will be given in this chapter.

4.1 System Architecture

The proposed system adds some new components and new operational steps to classical OpenID in order to handle certificate based smart card user authentication. This architecture provides a plug and play mechanism for identity tokens. We followed OpenID specification [5] and PKCS [6] standards in implementation.

4.1.1 System Components

Proposed system adds these components: a Java applet, a security token, a digital certificate and a certification authority to classical OpenID components: a user agent, a relying party, an OpenID provider. Proposed scheme does not require any change in existing relying parties and browsers. OpenID providers need modification to authenticate user with digital certificates. The CAs are not strictly dependent to the system, so there is no change required for them too. System components and explanations are shown in Figure 4.1 :

User Agent (UA): The browser on user terminal, it is same UA of classical OpenID. Our system does not require any change on it.

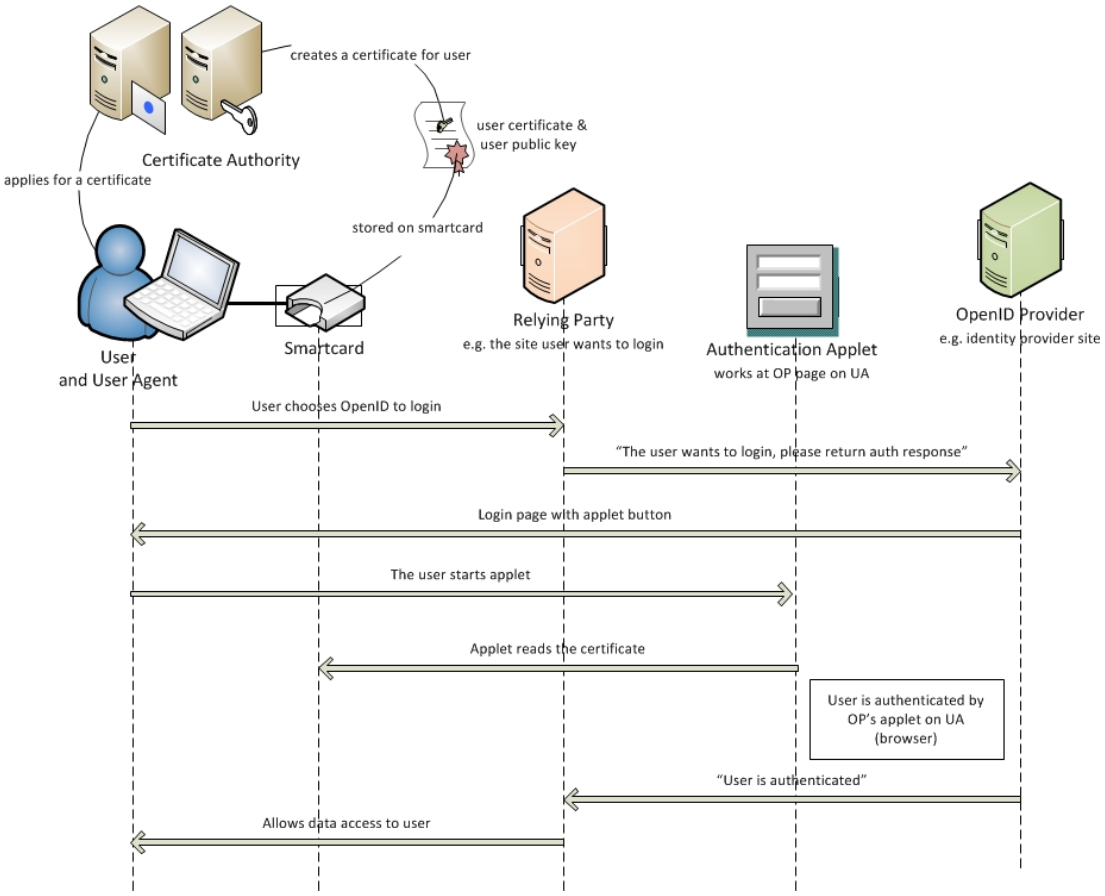


Figure 4.1: The proposed system overview

Relying Party (RP): A web application which asks for authenticating a user; same relying party as classical OpenID. Our system does not require any change on it.

OpenID Provider (OP): An OpenID authentication server on which RP relies for an authentication response. In this scheme, OP is not an identity provider any more (except the case that CAs are OPs). It only authenticates user via a digital certificate taken from a CA (or any issuer). Identity provider is the CA hereafter. OP verifies digital signature which authentication applet generates on smart card during user authentication phase. OP communicates with RP and returns authentication response after user is authenticated.

Smart Card (SC): It is a security token which contains digital certificates and their private keys on behalf of user. It provides PIN protection in case of

stealing. It is tamper-resistant and provides secure login. Private key cannot be excluded from smart card. It needs a smart card reader to be communicated. Card is communicated with Personal Computer/Smart Card (PC/SC) standard and uses PKCS#11 Cryptographic Token Interface Standard. The card file system is based on PKCS#15 Cryptographic Token Interface Standard. X.509 certificates (up to 4096 bytes), RSA public and private keys (up to 2048 bits) are stored on card file system besides plain data objects. Digital signing is performed on smart card operating system. The card file system is protected with a default issuer PIN. This issuer PIN is initialized with the value passed for PUK in card initialization phase. Password based encryption scheme described in PKCS#12 is used.

Authentication Applet (AA): It is a Java applet runs on browser. It is deployed on OP web site and is loaded when OP page is opened, launches when user clicks. AA handles communication with smart card, user certificate verification and generation of digital signature.

Digital Certificate (DC): It is a X.509 digital certificate which contains user attributes, user public key, CA signature and certificate attributes such as validity period. It is obtained from a CA and stored on smart card. It is used to generate a digital signature for login and also used to verify this signature. User authentication phase of OpenID is based on this certificate.

Certification Authority (CA): It is not a concrete party of our solution. The digital certificate is obtained from CA and is written on a smart card by this CA. Hence any certificate can be used for this solution. Digital certificate carries CA signature on it.

4.1.2 Operational Flow

This section gives an operational flow of the proposed scheme. Overall flow is demonstrated in Figure 4.2. In advance, user applies to a CA and obtains a (or more) digital certificate. CA delivers certificate written on a smart card. User

plugs smart card via a USB card reader to PC before login process. So, steps are:

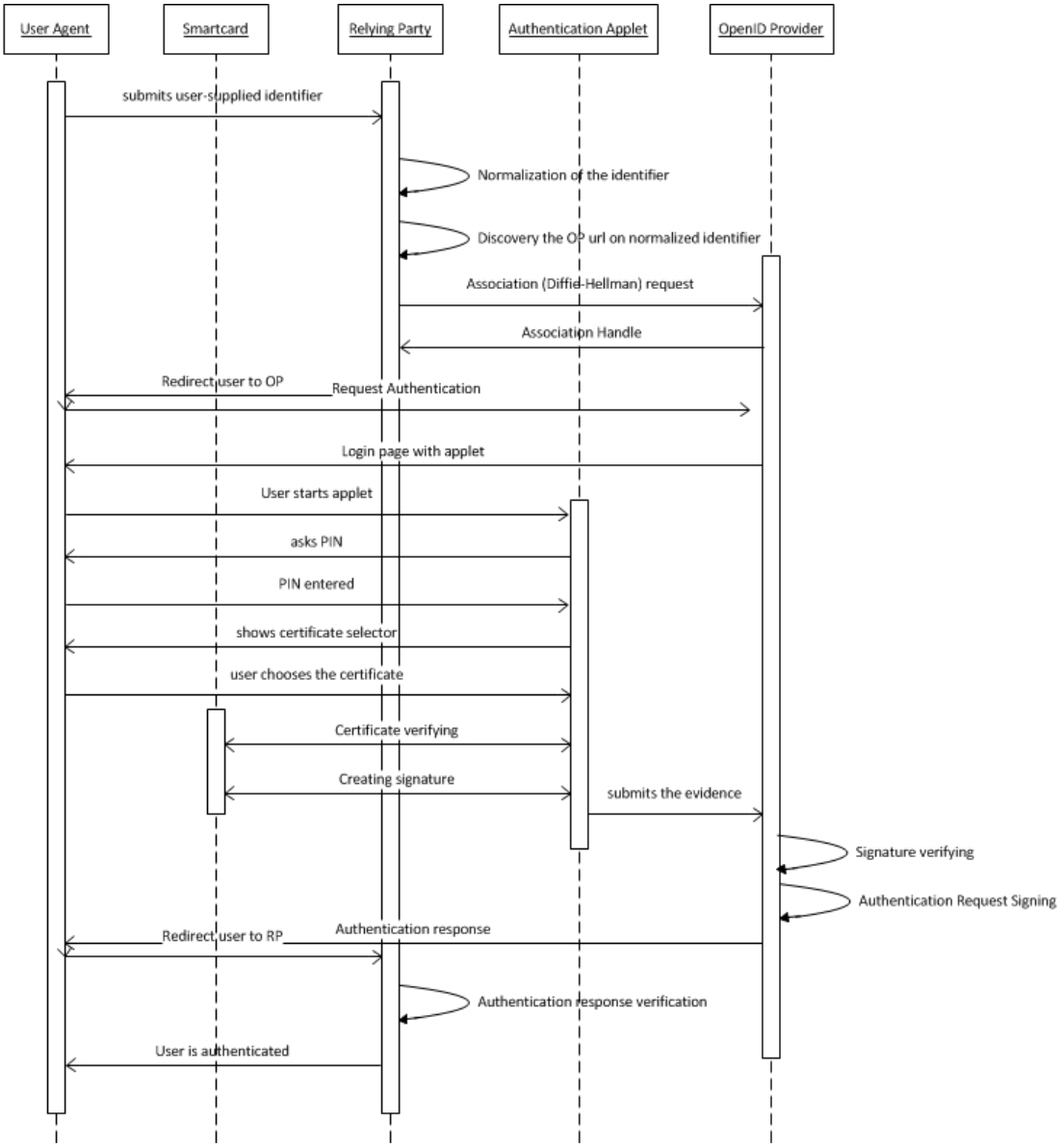


Figure 4.2: The proposed system operation flow

Step 1: User visits RP web page and chooses standard login with OpenID option, submits OP identifier as user-supplied identifier.

Step 2: RP performs normalization on the submitted input to transform it into an identifier, as usual.

Step 3: RP performs discovery on the identifier, as usual, to retrieve OP

end-point URL (in our implementation we prepeared XRDS document, therefore discovery is performed via Yadis)

Step 4: RP and OP perform association using Diffie-Hellman (DH) key exchange, as usual.

Step 5: RP redirects UA to OP with a redirection message, as usual.

Step 6: User clicks the button on OP page and AA (the applet) starts.

Step 7: AA asks user to enter PIN code.

Step 8: After PIN is entered correctly, AA shows certificate selector window.

Step 9: User selects the certificate he wants to use.

Step 10: AA checks certificate validity period and verifies CA signature on the certificate.

Step 11: AA keeps a string data ($st - data$) consisting of *login-phrase* + *system-time*.

Step 12: AA calculates a compare value ($comp - val$) adding constants *const1* and *const2* to $st - data$.

Step 13: AA forms signed attributes ($signed - attr$) consisting of digital certificate ($signing - cert$) and digest of $st - data$.

Step 14: AA sings hash of $signed - attr$ on smart card with user's private key to form signature (*signature*).

Step 15: AA forms a signature data ($sign - data$) with signed attributes and *signature*.

Step 16: AA submits $sign - data$ and hash of $comp - val$ to OP form.

Step 17: OP extracts original data $org - data$ from $sign - data$ and adds the same constants to it ($const1 - org - data - const2$) and checks if hash of

$const1 - org - data - const2$ is the same as hash of $comp - val$.

Step 18: OP extracts $signed - attr$ and $signing - cert$ from $sign - data$, verifies $signing - cert$ then operates user's public key obtained from $signing - cert$ on $signature$, eventually compares hash of result with hash of $st - data$.

Step 19: Then, if equal, user is authenticated.

Step 20: Then, OP redirects UA back to RP with an assertion message, as usual.

Step 21: RP verifies OP returned assertion, as usual.

Step 22: User now, can access RP website.

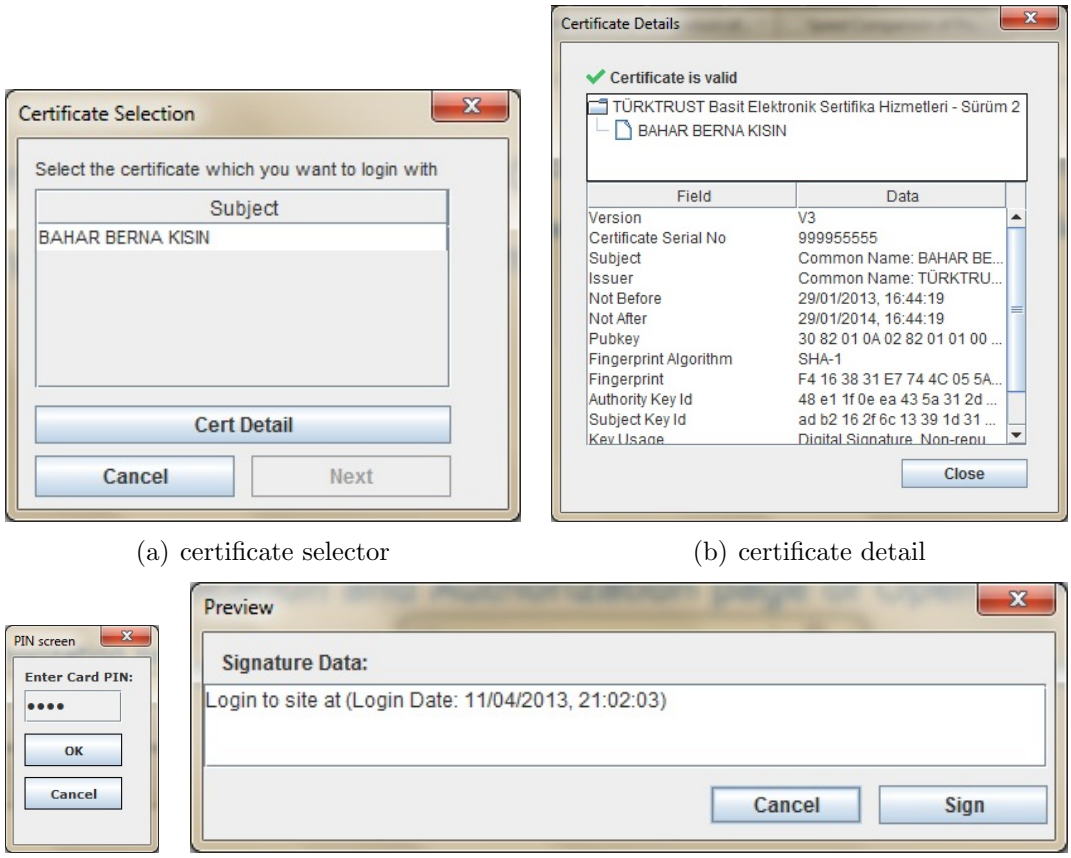
4.2 Implementation

In prototype, we implemented a pseudo-CA application which creates a digital certificate and writes certificate and private key to smart card. We also implemented a RP, an OP and an AA which communicates with card, verifies user certificate and generates signature. NetBeans IDE 7.2 is used as development environment.

Pseudo-CA application: was written in Java (JDK 1.7) language. The application generates an RSA key pair with SHA1PRNG using Java security package and will put public key in subject public key info field of the certificate. SHA-1 digest algorithm is used. X.509 [31] certificate fields are filled appropriately: subject info and public key are set to To-Be-Signed (TBS) Certificate; TBS certificate is signed with SHA1withRSA algorithm using signer (root or sub-root certificate) private key; TBS certificate, signature algorithm and signature itself are set into X.509 certificate. Then application initializes smart card environment by obeying PKCS#11 standard using card manufacturer's pkcs11 library. After that, it opens session on smart card with its PIN code and writes certificate and private key on it.

RP and OP: were implemented in Java and Java Server Pages (JSP) using an open source OpenID library: OpenID4Java [57]. OP contains a Java applet (AA) to meet requirements of the proposed scheme. After AA works, OP verifies signature submitted from the applet and so authenticates user.

AA: was implemented in Java (JDK 1.7) by Swing components using Java applet technology. Using applet technology, we are enabled to communicate to a device locally even though it is loaded within a web page. When web browser processes OP page that contains our applet, the applet is transferred to UA's system and executed by browser's Java Virtual Machine (JVM). The applet finds smart card reader and opens a session after asking user to enter PIN code. After reaching certificates repository on the card, it shows certificates on card for user selection. Then it validates validity period and verifies CA signature of the selected certificate. In this prototype root certificate is encoded statically on the program, however in real system a certificate database file can be used. Then a login phrase and system time (dataToBeSigned) are kept to sign and are shown to user. Applet screens are shown in Figures 4.3. Signature is performed on smart card and thereafter, signature and dataToBeSigned are submitted to OP form for verification.



(a) certificate selector

(b) certificate detail

(c) PIN

(d) signature preview

Figure 4.3: Applet screens

Chapter 5

Experimental Results

This chapter analyses the proposed system from the aspect of performance and users' opinion. Firstly, we measured the computation time for certificate verifying, signature generation and signature verification with different RSA key lengths. Secondly we analysed page loading time for login pages and Java applet. Lastly, we performed a user opinion study and measured task completion time for the proposed system.

Experiment Environment: In prototype experiments, we used an Intel Core2 Duo CPU 2.20 GHz with 3 GB RAM and 64 bit Windows 7 machine as local machine. The other machine in the LAN was an Intel Core2 Duo CPU 2.26 GHz with 3 GB RAM and 32 bit Windows 7 machine. We deployed RP on Glass-Fish Server Open Source Edition 3.1.2.2 (build 5). We deployed OP on Apache Software Foundation Tomcat Server 7.0.34. We chose Firefox 19.0.2 as UA with Java Plug-in 10.15.2.03. AA run with that Java plugin on JRE version 1.7.0-15-b03 Java HotSpot (TM) Client Virtual Machine. We used CardOS V4.3B smart card which run on the module Infineon SLE66CX322P with 32 Kbytes EEPROM and against differential fault analysis, simple power analysis, differential power analysis. We used ACS ACR38U smart card reader with USB 2.0 interface 4 MHz clock frequency. For the remote cases we used a 3G wireless access point via a cellphone; for the LAN and Localhost, we used an ADSL modem wireless access point.

5.1 RSA Key Length Comparison

We collected the computation time for certificate verification, signature generation and signature verification with 1024 and 2048 bits RSA key. The task was performed 20 times. Both OP and RP launched at localhost. Signature generation was performed on smart card; certificate verification and signature verification was performed on OP server.

The computation time in seconds is shown below in Table 5.1.

	1024			**2048**		
	cert vrf	**sign gen**	**sign vrf**	**cert vrf**	**sign gen**	**sign vrf**
	2,89	599,09	0,47	1,57	918,45	1,50
	1,50	601,61	0,41	1,56	925,42	1,47
	2,80	593,06	0,43	2,77	950,72	2,01
	1,84	589,50	0,40	2,93	937,53	1,47
	2,75	587,36	0,47	2,77	925,71	1,71
	2,76	588,65	0,44	2,80	918,14	1,42
	1,74	599,32	0,41	2,67	923,68	1,69
	2,80	655,44	0,41	1,50	926,95	1,43
	1,83	594,97	0,49	2,81	924,32	1,45
	2,77	646,59	0,41	2,75	916,86	1,67
	1,55	561,11	0,41	1,36	926,92	1,36
	2,55	548,60	0,46	1,36	963,76	1,35
	1,97	549,07	0,47	2,36	918,68	1,32
	1,02	552,82	0,46	2,46	929,69	1,24
	2,16	550,74	0,47	1,03	914,77	2,05
	1,00	550,21	0,44	1,23	918,67	1,05
	1,06	549,52	0,46	2,05	958,62	1,48
	2,64	552,02	0,47	2,65	918,07	1,64
	1,17	549,74	0,44	1,54	916,56	1,36
	2,31	548,62	0,41	1,03	924,22	1,47
	1,02	549,62	0,40	1,37	919,28	1,65
avr	**2,01**	**577,03**	**0,44**	**2,03**	**927,48**	**1,51**
stdev	**0,70**	**0,70**	**0,03**	**0,69**	**13,87**	**0,23**

Table 5.1: RSA key length comparison (milliseconds)

The average computation time for certificate verification and signature verification is shown in Figure 5.1.

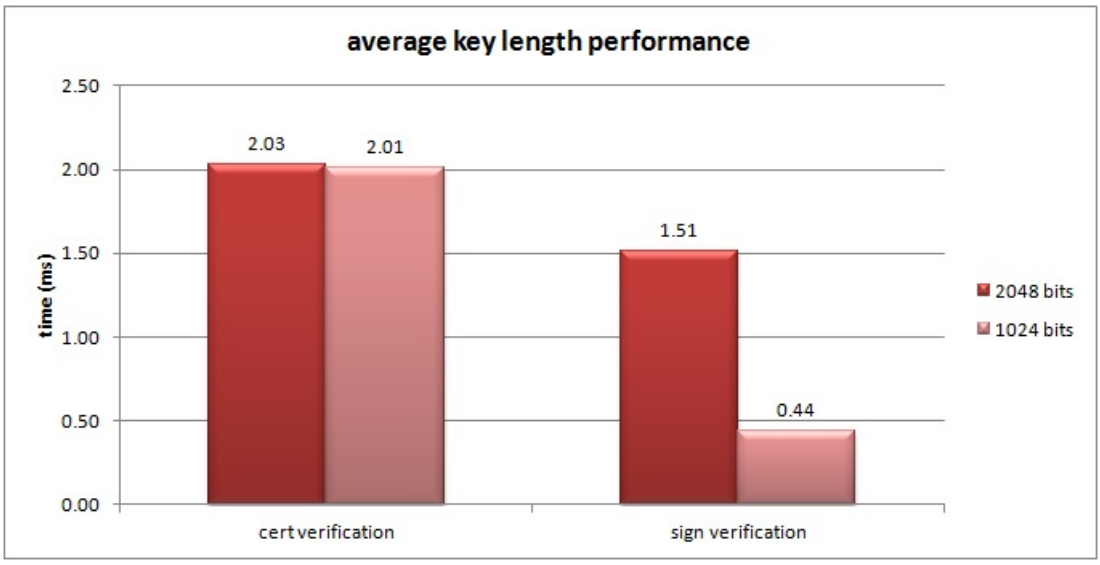


Figure 5.1: Average computation time according to key length

According to Table 5.1 and Figure 5.1, we can say that signature verification time depends on the key size. The computation time of signature generation process of 2048 bits RSA is 3 times bigger than 1024 bits RSA. Certificate verification contains CA signature verification; in the experiment we used the same root certificate for the two certificates, so the certificate verification process time becomes same and slightly differs from signature verification. Certificate verification occurs in approximately 2 milliseconds.

The average computation performance of signature generation is shown below in Figure 5.2.

According to Table 5.1 and Figure 5.2, we can say that signature generation depends on key size. In the experiment, signature generation in 2048 bits RSA is 60% longer than in 1024 bits RSA. Signature generation with 2048 bits RSA takes approximately 0.9 seconds and with 1024 bits 0.5 seconds. This process lasts much more than signature verification because, signature generation occurs in smart card processor and operations on smartcard are slower than operations on machines we used for hosting.

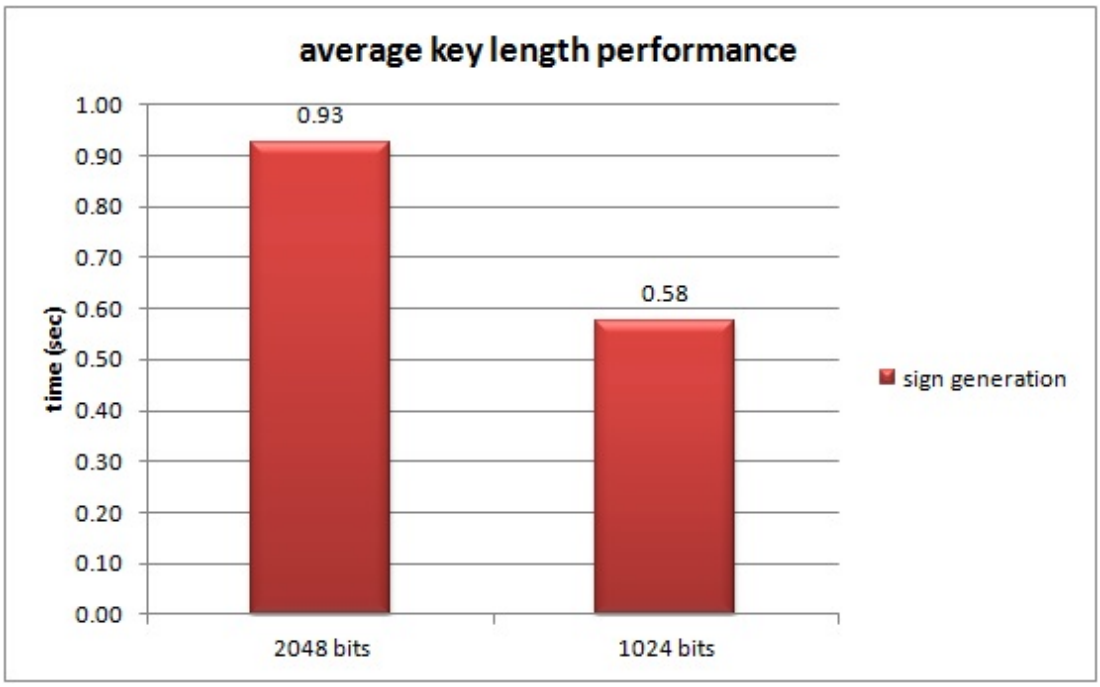


Figure 5.2: Average computation time of signature generation

5.2 Page Loading Time Comparison

We measured page loading time for OP login page. The comparison is between an implemented username-password authentication prototype and proposed system. Proposed system login page contains a Java applet and the other prototype contains username, password fields and a submit button. We performed the task 20 times and collected the results via a Firefox extension named Lori 0.2 (Life-of-request info). This tool measures time-to-first byte (first) and time-to-completion (comp) for html components. We measured applet loading time (app) computationally and evaluated it separately from time-to-completion. That is, page is loaded once, then applet is loaded. We performed the task when OP and RP is in the same machine (Local), in the same LAN (RP in different machine) and at Remote (with a real RP site: ficly.com).

The proposed system's page loading time is shown on the Table 5.2 below:

According to Table 5.2 and Figure 5.3, we can say that page loading time

	local app.	first	comp	lan app	first	comp	remote app.	first	comp
	0,03	0,16	0,24	0,03	0,42	0,51	0,09	2,92	2,95
	0,06	0,14	0,21	0,03	0,54	0,61	0,03	3,12	3,45
	0,04	0,04	0,13	0,03	0,05	0,10	0,03	1,35	1,41
	0,08	0,04	0,06	0,04	0,04	0,08	0,04	1,62	1,80
	0,03	0,02	0,06	0,03	0,07	0,11	0,03	1,71	1,73
	0,04	0,03	0,06	0,03	0,06	0,09	0,04	2,31	2,34
	0,03	0,03	0,05	0,03	0,04	0,08	0,03	1,88	2,08
	0,08	0,05	0,07	0,03	0,05	0,08	0,03	4,52	4,55
	0,03	0,04	0,08	0,03	0,04	0,08	0,03	2,24	2,27
	0,08	0,03	0,06	0,03	0,04	0,07	0,03	1,66	1,69
	0,03	0,03	0,05	0,03	0,19	0,23	0,04	1,53	1,71
	0,03	0,08	0,11	0,03	0,05	0,08	0,03	1,34	1,37
	0,03	0,04	0,06	0,03	0,04	0,08	0,05	2,62	2,66
	0,03	0,04	0,07	0,03	0,06	0,09	0,09	1,50	1,54
	0,10	0,04	0,06	0,03	0,04	0,07	0,05	1,84	1,88
	0,04	0,03	0,06	0,03	0,06	0,09	0,03	2,08	2,11
	0,04	0,03	0,06	0,03	0,04	0,07	0,05	1,94	1,97
	0,03	0,03	0,06	0,03	0,04	0,06	0,04	1,49	1,53
	0,09	0,03	0,07	0,03	0,15	0,18	0,08	2,68	2,71
	0,05	0,04	0,06	0,03	0,05	0,08	0,03	2,72	2,77
avr	**0,05**	**0,05**	**0,08**	**0,03**	**0,10**	**0,14**	**0,04**	**2,15**	**2,23**
sdev	**0,02**	**0,04**	**0,05**	**0,00**	**0,14**	**0,15**	**0,02**	**0,78**	**0,79**

Table 5.2: Proposed system page loading time (seconds)

and applet loading time do not much differ in Localhost and LAN and also their average are both under 0.1 second. However, page loading for remote machines take much more time with an average of 2.27 (time-to-comp + applet loading time). This is 4 times bigger than the Localhost. In addition, there is a very slight difference between the time-to-first byte and time-to-completion in Localhost and LAN with an average of 0.03 seconds and 0.08 seconds in remote machine. Nevertheless the page loading time in remote case is reasonably good.

The comparison of username-password authentication and proposed system page loading time is shown in Table 5.3 below. In this result set, applet loading time is added to the time-to-completion for the proposed system.

According to Table 5.3 and Figure 5.4 below, we can say that page loading

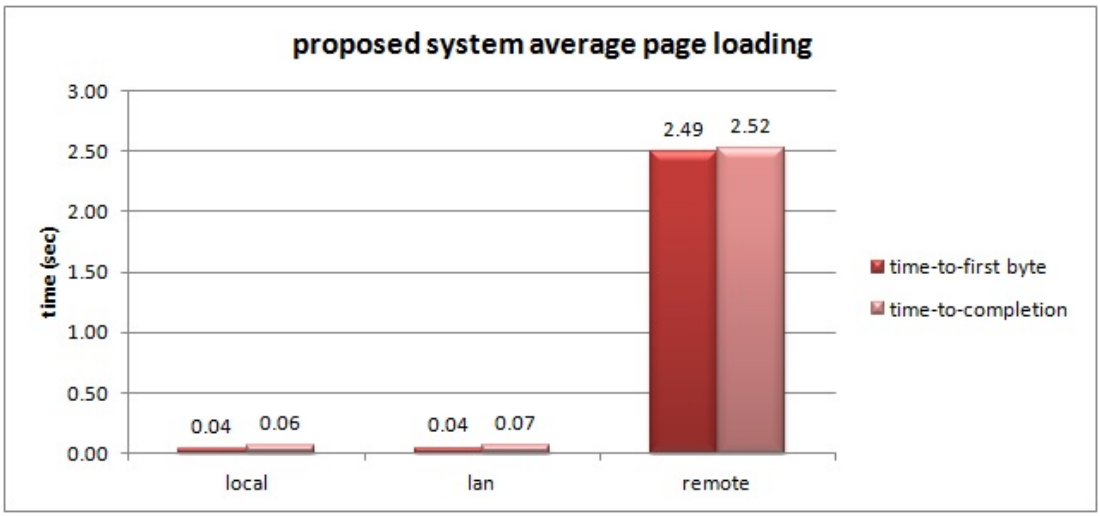


Figure 5.3: Proposed system average page loading time

time of the two scheme does not much differ in Localhost and LAN which is under 0.2 seconds for both cases. Page loading time for proposed system is 0.06 and 0.1 second respectively longer than the username-password authentication. However, for the remote case, the page loading time is above 2 seconds in both the username-password and proposed scheme. Nevertheless, the proposed scheme's page loading time is 0.6 seconds less than the username-password scheme for the remote case. So, the proposed scheme gives 25% better results in remote systems (the real RP site) which also constitute the real use case of the proposed system.

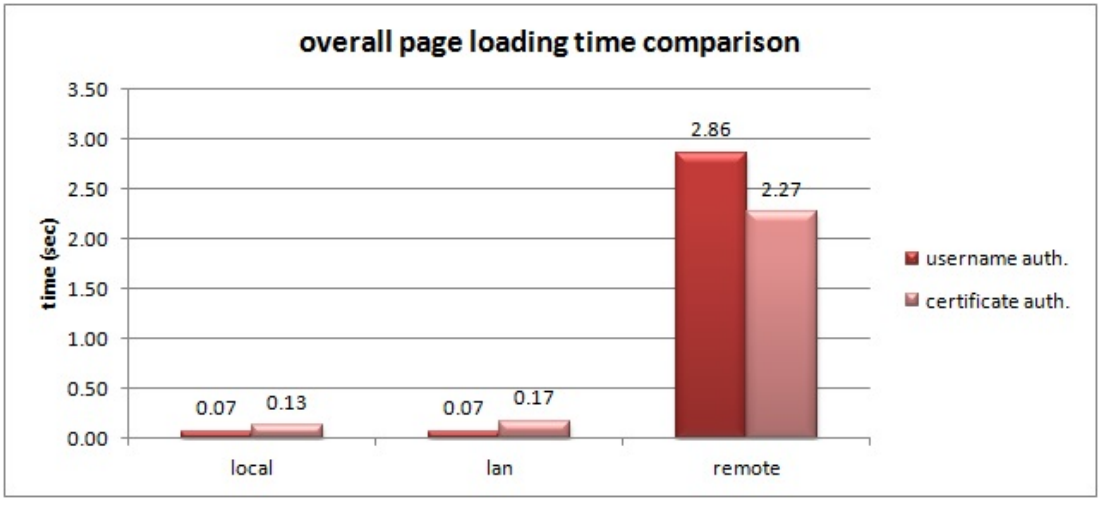


Figure 5.4: Average page loading time comparison

	local		lan		remote	
	usr-pass	**cert auth**	**usr-pass**	**cert auth**	**usr-pass**	**cert auth**
	0,06	0,28	0,08	0,55	3,07	3,03
	0,12	0,27	0,07	0,65	3,70	3,48
	0,11	0,17	0,06	0,13	2,16	1,44
	0,10	0,14	0,07	0,11	2,83	1,83
	0,11	0,09	0,07	0,14	2,99	1,77
	0,09	0,10	0,07	0,12	5,24	2,38
	0,05	0,08	0,06	0,10	4,22	2,11
	0,04	0,16	0,06	0,11	2,70	4,58
	0,05	0,11	0,07	0,10	2,91	2,30
	0,05	0,14	0,06	0,09	2,56	1,72
	0,05	0,08	0,06	0,26	1,84	1,75
	0,05	0,14	0,06	0,10	2,27	1,40
	0,06	0,10	0,07	0,11	1,98	2,71
	0,05	0,10	0,06	0,11	2,03	1,63
	0,06	0,16	0,07	0,10	1,89	1,93
	0,08	0,10	0,06	0,11	2,91	2,14
	0,11	0,09	0,08	0,10	3,07	2,02
	0,05	0,09	0,10	0,09	2,76	1,57
	0,04	0,16	0,07	0,21	2,46	2,79
	0,10	0,11	0,06	0,11	3,62	2,80
avr	**0,07**	**0,13**	**0,07**	**0,17**	**2,86**	**2,27**
stdev	**0,03**	**0,06**	**0,01**	**0,15**	**0,84**	**0,79**

Table 5.3: Time to page load completion comparison (seconds)

5.3 User Study

We asked 20 participants to login a real RP site: ficly.com (which supports OpenID for login) with our proposed system and also with an implemented OP with username-password authentication. The task begins with visiting the real RP site, ends with successful login and contains operations therebetween. We collected their task completion time, user authentication time and opinions in the matters of ease, safety, like to use, satisfaction and overall with points from 1 (the least) to 10 (the most). The results are shown in Tables 5.4 and 5.5. Task completion time refers to total time spent. Authentication time refers to time spent in user authentication phase (e.g. for proposed system signing and verification time on smartcard; for username scheme password authentication time).

During measuring authentication time, network delay and user interaction time was ignored.

partic-ipants	**ease**	**safe**	**like to use**	**satis-faction**	**overall**	**completion time (s)**	**authen. time (s)**
p1	9	8	7	8	8	20,55	2,22
p2	9	7	8	8	8	21,85	2,24
p3	8	8	5	6	7	20,86	2,30
p4	8	8	7	8	8	20,41	2,24
p5	7	7	5	7	7	20,23	2,24
p6	10	9	9	9	9	21,56	2,22
p7	10	9	9	8	9	20,54	2,23
p8	10	9	6	8	8	20,25	2,23
p9	7	9	9	10	9	19,32	2,22
p10	9	9	7	9	9	20,74	2,21
p11	9	9	6	9	8	20,54	2,29
p12	8	10	8	8	8	20,63	2,22
p13	10	9	9	8	9	21,96	2,23
p14	7	10	9	9	9	21,58	2,25
p15	8	8	8	7	8	21,31	2,22
p16	9	8	7	6	7	20,13	2,22
p17	9	9	7	8	8	20,53	2,30
p18	8	9	6	8	8	20,16	2,25
p19	8	10	8	6	8	20,42	2,22
p20	8	10	8	9	8	19,74	2,23

Table 5.4: Proposed system user opinion, task completion and authentication time (seconds) with 20 participants

According to Figure 5.6, participants find the proposed system as easy as common username-password scheme. On the other hand, participants think the proposed scheme is more secure with a percentage of 50% more than the username-password scheme. All of the users have been satisfied to use such a system with a high percentage of 80% which is also more than the username-password scheme. Users think generally the proposed system as likely-to-use with a percentage of 70% which is more than the other scheme.

partic-ipants	**ease**	**safe**	**like to use**	**satis-faction**	**overall**	**completion time (s)**	**authen. time (s)**
p1	10	5	7	8	8	15,25	0,01
p2	8	7	7	8	8	15,54	0,01
p3	10	6	8	8	8	15,36	0,01
p4	9	5	5	5	6	14,96	0,00
p5	10	4	4	5	5	15,89	0,01
p6	6	8	5	8	7	15,36	0,01
p7	10	7	7	8	8	15,42	0,01
p8	9	6	7	7	7	15,28	0,00
p9	9	7	9	10	8	14,56	0,00
p10	10	5	6	8	8	14,21	0,00
p11	10	4	6	7	7	14,24	0,01
p12	9	7	8	5	7	14,98	0,01
p13	9	8	7	5	7	15,75	0,01
p14	9	6	6	6	6	15,41	0,01
p15	8	6	7	8	7	16,21	0,00
p16	8	5	7	6	6	15,63	0,00
p17	10	6	6	5	6	15,82	0,01
p18	10	5	5	6	6	15,24	0,01
p19	8	5	4	7	6	15,32	0,01
p20	10	4	5	7	6	15,96	0,00

Table 5.5: Username scheme user opinion, task completion and authentication time (seconds) with 20 participants

According to Figure 5.5, participants spent more time on proposed system, since communication with smart card made them lose time. Also the fact that smart card operations are not as fast as server machine operations, leads to an increase in task completion time. Nevertheless, this difference is not much not to consider the proposed system as a login scheme.

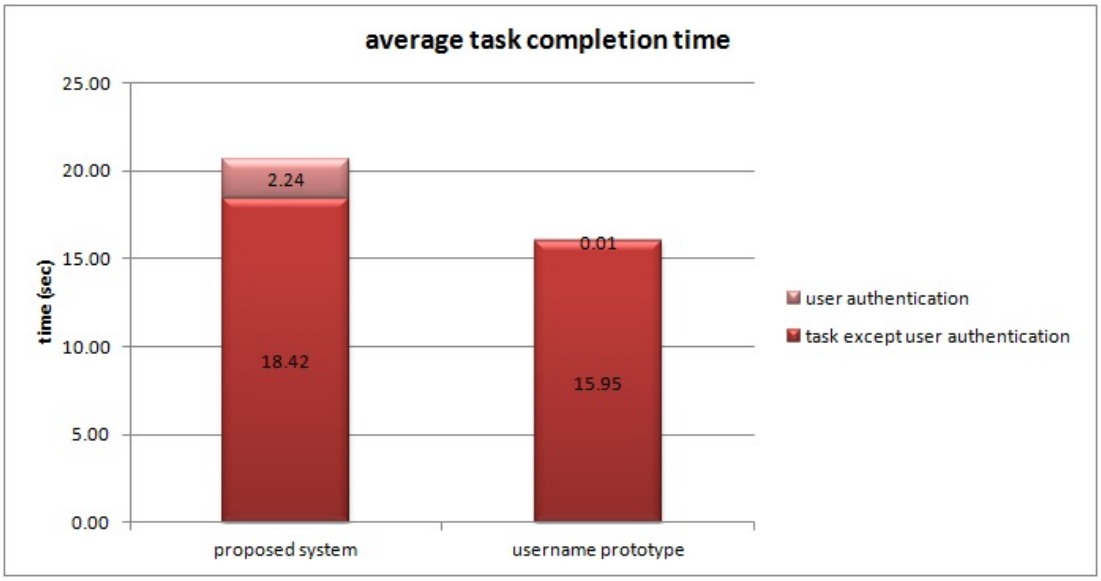


Figure 5.5: Average task completion time

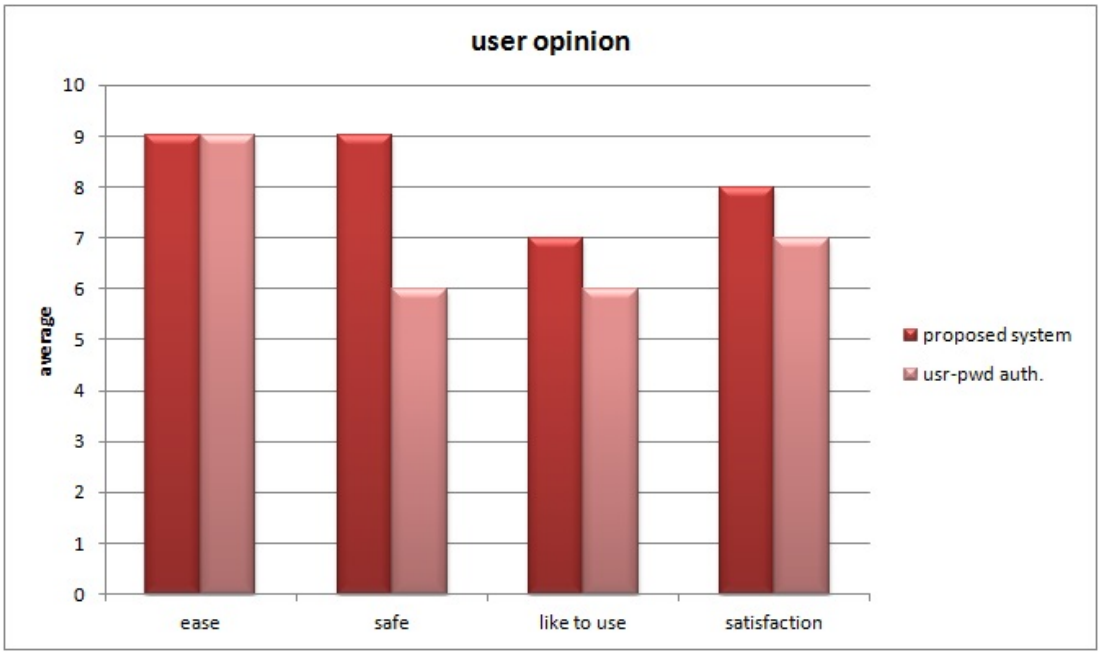


Figure 5.6: Participant opinion

Chapter 6

Conclusion

During the thesis study, we worked on improving security of user authentication scheme of OpenID with a user-friendly solution. A background information about federated identity systems including OpenID, OAuth, SAML and also smart card technology and certificate based authentication is given. The security weakness of OpenID is studied and related works are discussed with respect to security.

In order to meet the requirement, we developed a system which integrates certificate based user authentication via smart card technology with OpenID protocol. The proposed system authenticates users of digital certificates who has smart card to real relying parties. Besides, the system rescues OpenID providers from keeping and managing identity information. The system authenticates user with a digital signature generated while logging in RP site using a digital certificate stored on user's smart card. The users also have the ability to manage their identities using the proposed scheme by selecting an appropriate digital certificate. The system provides facilities of OpenID such as single-sign on and decentralized identity management. The system uses an implemented Java Applet in order to communicate smart card and perform cryptographic operations. This applet enables to authenticate user at OpenID provider but on user terminal locally. To security considerations, knowledge of issuer PIN allows installing additional packages on smart card. The standard PC/SC stack is used to communicate with smart card. This communication might be intercepted to spy out sensitive

information or to pretend wrong results. Effective measures should be taken to protect the overall system security to keep Trojan Horses and other malicious software.

The experimental results of the proposed system shows that the solution is fast in means of page loading and cryptographic computation. In addition it is well-accepted by the participants which we asked for testing the system. Also, we included a comparison study about different key lengths of RSA scheme in cryptographic functions.

6.1 Future Work

The future goals may be to improve security and usability of the proposed system:

The thesis proposed work does not include SSL or TLS as a secure tunnel which is provided by a trusted party, which may lead to a weak security issue such as eavesdropping and tampering. A secure tunnel should be applied over OpenID protocol.

The communication with smart card and operations performed on smart card makes user lose time, which may be tedious for users. The smart card operations may be augmented to be faster.

OpenID offers attribute exchange across the parties. The digital certificates are convenient to keep user attributes, especially with extensions. So, it may be valuable to study to exchange OpenID user attributes via digital certificates.

The solution may be extended to support OpenID Connect, which combines OpenID and OAuth, when it becomes a finalized specification.

Bibliography

[1] C. Lin, H. Sun, and T. Hwang, "Attacks and solutions on strong-password authentication," *IEEE Transactions on Communication*, vol. E84-B, no. 9, 2001.

[2] K. Shim, "Off-line password-guessing attacks on the generalized key agreement and password authentication protocol," *Applied Mathematics and Computation*, vol. 169, no. 1, pp. 511 – 515, 2005.

[3] J. Munilla and A. Peinado, "Off-line password-guessing attack to Peyravian-Jeffries's remote user authentication protocol," *Computer Communications*, vol. 30, no. 1, pp. 52 – 54, 2006.

[4] K. A. Shim, "Security flaws of remote user access over insecure networks," *Computer Communications*, vol. 30, no. 1, pp. 117 – 121, 2006.

[5] "OpenID Authentication 2.0 - Final," 2007. `http://openid.net/specs/openid-authentication-2_0.html`.

[6] "Public-Key Cryptography Standards - PKCS." `http://www.rsa.com/rsalabs/node.asp?id=2124`.

[7] D. Rountree, "Chapter 2 - What Is Federated Identity?," in *Federated Identity Primer*, pp. 13 – 36, Boston: Syngress, 2013.

[8] "Identity Commons." `http://www.idcommons.net/`.

[9] D. Rountree, "Chapter 3 - Federated Identity Technologies," in *Federated Identity Primer*, pp. 37 – 60, Boston: Syngress, 2013.

[10] "OAuth Community." http://oauth.net/.

[11] B. Leiba, "OAuth web authorization protocol," *Internet Computing, IEEE*, vol. 16, no. 1, pp. 74–77.

[12] "Online Community for SAML OASIS Standard." http://saml.xml.org/.

[13] "Windows Identity Foundation." http://msdn.microsoft.com/en-us/library/hh377151.aspx.

[14] "OpenID and OAuth Hybrid Extension." http://wiki.openid.net/w/page/12995194/OpenID%20and%20OAuth%20Hybrid%20Extension.

[15] "OpenID Connect." http://openid.net/connect/.

[16] "Google Accounts Authentication and Authorization," 2013. https://developers.google.com/accounts/docs/OAuth2Login.

[17] D. Morin, "Announcing Facebook Connect," 2008. http://developers.facebook.com/blog/post/2008/05/09/announcing-facebook-connect/.

[18] J. M. Seigneur and T. E. Maliki, "Chapter 17 - Identity Management," in *Computer and Information Security Handbook* (J. Vacca, ed.), pp. 269 – 292, Boston: Morgan Kaufmann, 2009.

[19] "OpenID Foundation." http://openid.net/.

[20] "What is OpenID." http://openid.net/get-an-openid/what-is-openid/.

[21] "OpenID Benefits for Individuals." http://openid.net/get-an-openid/individuals/.

[22] "OpenID Benefits for Accepters." http://openid.net/add-openid/.

[23] J. Miller, "Yadis Specification 1.0."

[24] D. Recordon and D. Reed, "OpenID 2.0: a platform for user-centric identity management," in *Proceedings of the second ACM workshop on Digital identity management*, DIM '06, (New York, NY, USA), pp. 11–16, ACM, 2006.

[25] D. Reed and D. McAlpin, "Extensible Resource Identifier (XRI) Syntax V2.0," 2005. https://www.oasis-open.org/committees/download.php/15376.

[26] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee, "Hypertext Transfer Protocol - HTTP/1.1, RFC 2616," 1999. http://www.ietf.org/rfc/rfc2616.txt.

[27] D. Chappell, "Introducing Windows CardSpace." http://msdn.microsoft.com/en-us/library/aa480189.aspx.

[28] "Higgins Personal Data Service," 2009. http://www.eclipse.org/higgins/.

[29] A. Lindholm, "Security Evaluation of the OpenID Protocol," M.Sc. Thesis, Royal Institute of Technology, Stockholm, Sweden, 2009.

[30] C. Adams and S. Lloyd, "Understanding PKI: concepts, standards, and deployment considerations," Boston: Addison-Wesley, 2nd ed., 2003.

[31] D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile," 2008. http://tools.ietf.org/html/rfc5280.

[32] J. Lopez, R. Oppliger, and G. Pernul, "Authentication and authorization infrastructures (AAIs), a comparative survey," *Computers and Security*, vol. 23, no. 7, pp. 578 – 590, 2004.

[33] J. Chau, "Digital certificates Is their importance underestimated?," *Computer Fraud and Security*, vol. 2005, no. 12, pp. 14 – 16, 2005.

[34] J. Camenisch, "Information privacy?!," *Computer Networks*, vol. 56, no. 18, pp. 3834 – 3848, 2012.

[35] S. Brands, *Rethinking Public Key Infrastructure and Digital Certificates-Building in Privacy.* Ph.D. Thesis, Eindhoven Institute of Technology, Eindhoven, The Netherland, 1999.

[36] T. M. Jurgensen and S. Guthery, *The Smart Cards: A Developer's Toolkit*. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2002.

[37] C. Fancher, "In your pocket: smartcards," *Spectrum, IEEE*, vol. 34, no. 2, pp. 47–53.

[38] J. P. Thomasson and L. Baldi, "Smartcards: portable security," in *Innovative Systems in Silicon, 1997. Proceedings., Second Annual IEEE International Conference on*, pp. 259–265.

[39] "ISO 7816-4, Identification cards - Integrated circuit cards - Organisation, security and commands for interchange," 2005. `http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=36134`.

[40] N. Itoi and P. Honeyman, "Practical security systems with smartcards," in *Hot Topics in Operating Systems, 1999. Proceedings of the Seventh Workshop on*, pp. 185–190.

[41] A. Matthews, "Side-channel attacks on smartcards," *Network Security*, vol. 2006, no. 12, pp. 18 – 20, 2006.

[42] M. Kuhn and O. Kmmerling, "Physical security of smartcards," *Information Security Technical Report*, vol. 4, no. 2, pp. 28 – 41, 1999.

[43] H. B. El, "Known attacks against smartcards." `http://www.infosecwriters.com/text_resources/pdf/Known_Attacks_Against_Smartcards.pdf`.

[44] A. Leicher, A. Schmidt, and Y. Shah, "Smart OpenID: A smart card based OpenID protocol," *Information Security and Privacy Research*, pp. 75–86, 2012.

[45] P. Urien, "An OpenID provider based on SSL smart cards," in *Consumer Communications and Networking Conference (CCNC), 2010 7th IEEE*, pp. 1–2.

[46] P. Urien, "Convergent identity: Seamless OPENID services for 3G dongles using SSL enabled USIM smart cards," in *Consumer Communications and Networking Conference (CCNC), 2011 IEEE*, pp. 830–831.

[47] I. Jrstad, T. Johansen, E. Bakken, C. Eliasson, M. Fiedler, and D. V. Thanh, "Releasing the potential of OpenID and SIM," in *Intelligence in Next Generation Networks, 2009. ICIN 2009. 13th International Conference on*, pp. 1–6, Oct.

[48] R. Penz, "Analysis and Design of a SIM based authentication solution for WLANs," diplomarbeit, Technische Universit Munich, Germany, September 2004.

[49] A. S. Ahmed, "A user friendly and secure OpenID solution for smart phone platforms," Master's thesis, Aalto University, Espoo, Finland, 2010.

[50] "Identity management and 3GPP security interworking; identity management and generic authentication architecture (GAA) interworking, release-9," 2009. `http://www.3gpp.org/ftp/Specs/html-info/33924.htm`.

[51] J. Vossaert, J. Lapon, B. D. Decker, and V. Naessens, "User centric identity management using trusted modules," *Mathematical and Computer Modelling*, vol. 57, no. 78, pp. 1592 – 1605, 2013.

[52] E. Erdem, K. Kucukkurt, K. Samurkas, E. Kanargi, and U. Celikkan, "A smart card based single sign-on and password management solution as a browser extension," in *Education and Management Technology (ICEMT), 2010 International Conference on*, pp. 539–543.

[53] H. Al-Sinani, "Integrating OAuth with information card systems," in *Information Assurance and Security (IAS), 2011 7th International Conference on*, pp. 198–203.

[54] D. L. Jobusch and A. E. Oldehoeft, "A survey of password mechanisms: Weaknesses and potential improvements. part 1," *Computers and Security*, vol. 8, no. 7, pp. 587 – 604, 1989.

[55] L. Shinder and M. Cross, "Chapter 11 - Passwords, Vulnerabilities and Exploits," in *Scene of the Cybercrime (Second Edition)*, pp. 467 – 503, Burlington: Syngress, second edition ed., 2008.

[56] T. Matthews, "Passwords are not enough," *Computer Fraud and Security*, vol. 2012, no. 5, pp. 18 – 20, 2012.

[57] "OpenID 2.0 Java Libraries." `http://code.google.com/p/openid4java/`.