



Joonas Venho

Mobile Signature Service Integration

Metropolia University of Applied Sciences
Bachelor of Engineering
Information Technology
Thesis
7 May 2012

Tekijä Otsikko	Joonas Venho Mobiilivarmenteen integrointi
Sivumäärä Aika	51 sivua 7.5.2012
Tutkinto	insinööri (AMK)
Koulutusohjelma	tietotekniikka
Suuntautumisvaihtoehto	ohjelmistotekniikka
Ohjaajat	IBM SWG Middleware Software Services Country Manager Finland – Alekski Tuovinen IBM SWG Information security specialist - Risto Rantalaiho Yliopettaja Jarkko Vuori
<p>Tivoli Access Manager for e-business (TAMeb) on IBM Security Systemsin tuote, jolla tuotetaan kertakirjautumisratkaisuja (SSO) web-palveluille. Se on ohjelma, joka käsittelee todennukseen liittyviä asioita ja hallinnoi sovellusten resurssien tietoturvasääntöjä.</p> <p>Normaalin todennuksen lisäksi TAMeb tarjoaa rajapinnan ulkoiseen tunnistautumiseen. Viime vuonna Suomessa ilmestyneen mobiilivarmennepalvelun myötä nousi kysymys, voisiko palvelun integroida Tivoli Access Manageriin ulkoisen tunnistautumisrajapinnan avulla.</p> <p>Tämän projektin tarkoitus oli luoda toimiva testiohjelma edellä mainitusta aiheesta. Tulos saavutettiin käyttämällä mobiilivarmennetestipalveluita, joita DNA, Elisa ja Sonera tarjosivat. Nämä yhtiöt kehittivät mobiilivarmennepalvelun yhdessä Suomeen ja ovat kaikki Finnish Federation for Communications and Teleinformatics (FiCom) jäseniä.</p> <p>Palvelu perustuu European Telecommunication Standards Institutun (ETSI) standardeihin, jotka määrittelevät mobiilivarmenne- ja mobiiliverkkovierailupalvelut. Standardien pohjalta FiCom julkaisi MSS FiCom -soveltamisohjeen suomalaisille palveluntarjoajille. Sovellus, joka tässä työssä kehitettiin, pohjautuu FiComin dokumentaatioon.</p> <p>Sovellus toimi odotetulla tavalla. Tunnistautumispyyntö lähetettiin operaattorin palveluun, puhelin vastaanotti operaattorin lähettämän tunnistautumispyynnön ja käyttäjätiedot palautettiin ohjelmalle.</p> <p>Työn lopullinen versio jäi testaamatta ja vaatii lisää hienosäätöä ennen tuotantoa, mutta saavutettu testivaihe osoittaa jo integraation toimivuuden. Integraatio tarjoaa jo nykyisellään uusia liiketoimintamahdollisuuksia.</p>	
Avainsanat	mobiilivarmenne, TAMeb, IBM, tunnistautuminen.

Author Title	Joonas Venho Mobile signature service integration
Number of Pages Date	51 pages 7 May 2012
Degree	Bachelor of Engineering
Degree Programme	Information Technology
Specialisation option	Software Engineering
Instructor(s)	Aleksi Tuovinen IBM SWG – Middleware Software Services Country Manager Finland Risto Rantalaiho IBM SWG - Information security specialist Jarkko Vuori Principal Lecturer
<p>Tivoli Access Manager for e-business (TAMeb) is an IBM Security Systems product for single sign-on solutions (SSO) on the web. It is software for handling authentication and authorization issues and managing the security policies of application resources.</p> <p>In addition to normal authentication TAMeb provides an interface for external authentication. The recent appearance of mobile signature service in Finland, which was developed jointly by DNA, Elisa and Sonera, raised the question of whether the service could be integrated to the Tivoli Access Manager with an external authentication interface. As a result this project was started and the purpose of it was to build working test software. The goal was achieved by using the test services provided by the operators. These operators are part of the Finnish Federation for Communications and Teleinformatics (FiCom).</p> <p>The service is based on European Telecommunication Standards Institute (ETSI) standards which specify the Mobile Signature Service and the mobile signature roaming services. According to these standards the FiCom Implementation guideline was introduced to Finnish application providers. The solution developed in this project is based on that documentation.</p> <p>The solution itself worked as expected. The authentication request was sent from the solution to the test service, the mobile telephone received the authentication request sent by the test service and the user information was sent back to the software.</p> <p>The final version of the solution requires more fine-tuning and testing before actual implementation but the current test phase already shows that the concept works. The integration already provides new business possibilities as it is.</p>	
Keywords	Mobile Signature Service, TAMeb, IBM, authentication

Table of contents

Abbreviations and terms

Preface

1	Introduction	1
1.1	Topic	1
1.2	Tools & IDE	2
1.2.1	WebSphere Application Server	3
1.3	TAMeb	4
1.3.1	GSKit	5
1.3.2	LDAP	5
1.3.3	WebSEAL	5
1.3.4	EAI	7
2	About Mobile Signature Service	9
2.1	General	9
2.2	Business Possibilities	13
2.3	SATU	15
2.4	ETSI-standard & FiCom-Implementation Guideline	16
2.4.1	Information Security	22
2.4.2	Spam Prevention Code	22
3	Solution	24
3.1	High Level Structure	24
3.2	Laverca API	27
3.3	Features	32
3.3.1	SSL	33
3.4	Configuration	33
3.5	Source Code	34
3.5.1	SSL Class	35
3.5.2	MSSrequest Class	36
3.5.3	ResponseHandler Class	38
4	Testing & Demonstration	39

5	Comparing the Solutions	42
5.1	TAMeb MSS Integration vs Vetuma Integration	42
5.2	ETSI-interface vs Mobile TUPAS	42
5.3	Alternative Solutions	44
6	Conclusion	46
6.1	Operator Differences	46
6.2	MSS Advantages and Disadvantages	47
	References	49

Abbreviations and terms

AE	Acquiring Entity
AP	Application Provider
EAI	External Authentication Interface
ETSI	European Telecommunication Standards Institute
FiCom	Finnish Federation for Communications and Teleinformatics
GSKit	Global Secure ToolKit
HETU	Finnish term for social security number
HMSSP	Home Mobile Signature Service Provider
IDE	Integrated development environment
LDAP	Lightweight Directory Access Protocol
Mobile Signature	Digital signature made on a mobile device which is based on Public Key Infrastructure (PKI). Used for user authentication.
MSS	Mobile Signature Service
MSSP	Mobile Signature Service Provider
RAD	Rational Application Developer
SATU	Electronic Identity given for Finnish citizens
SIM	Subscriber identity module
SMS	Short Message Service
SOAP	Simple Object Access Protocol
SSL	Secure Sockets Layer, a protocol for encrypting information over the Internet.
TAMeb	Tivoli Access Manager for e-business

TUPAS

Finnish banks' authentication method created by Federation of Finnish Financial Services. It is provided by major Finnish banks for web services that require strong authentication.

WAS

WebSphere Application Server

Preface

I would like to thank all the teachers of the Metropolia University of Applied Sciences who have helped me through my studies. Special thanks to Jarkko Vuori for providing excellent feedback on this project.

I would also like to express my gratitude to my supervisor Aleksu Tuovinen at IBM and my mentor Risto Rantalaiho who helped me through this project and provided insight when I found myself in a dead-end. Finally big thanks to Janne Lähteenmäki, Antti Merihaara and Harri Stranden at IBM for giving professional opinions and guidance.

1 Introduction

1.1 Topic

In Finland mobile signature service (MSS) is an authentication service provided by Finnish operators: Elisa, DNA and Sonera (cf. figure 1). The strong authentication and the ease of using it resulted in questions from the clients whether the service could be used with IBM security systems authentication and authorization product called Tivoli Access Manager for e-business (TAMeb) as an external authentication method. Because of this the project was started to create MSS integration for TAMeb. By the time the project was started, there were already clients waiting for the product.



Figure 1: Mobile Signature Service logos by operators

The actual feature desired by IBM is the mobile identification but the standard for mobile signature service includes both signature and identification. Mobile signature can be used as a legal replacement for the physical signature made with a pen and it can be used to make legally binding commitments. Mobile identification on the other hand is the service that can be used to prove identity. Both of these features are used by entering a personal code on the mobile device.

Mobile signature service is already used by a few large companies such as the insurance company If and it is also used in practically all major city portals and university portals in the web. According to If's specialists, the Finnish bank ID TUPAS revolutionized their online services and caused a massive growth in usage. Over 50% of their insurance claims are done over the internet and over 80% of these claims require strong authentication. Now the current TUPAS authentication is a really good solution and it is easy to use but according to the specialists the mobile signature service is the next step to make it even easier and more convenient. [1.]

The main reason for using mobile signature service with TAMEb is to achieve strong authentication for the users and also a secure way to get the user's social security number HETU for the application protected by TAMEb. In fact in this solution anyone with a cellphone equipped with a mobile certificate can access the service since having mobile certificate capability on a mobile device means it must have been registered along with the Finnish social security number (HETU). All users are also authenticated with the same credentials in TAMEb and the final goal of the software is to deliver the HETU safely to the end application.

The purpose of this work is to create a specification according to the FiCom-implementation guideline and use this specification to create a working test-application to demonstrate that the concept works. The specification plays a big part in this work as it will inform IBM about the external requirements for a fully functional MSS system such as the deals with operators and application providers. The final solution would be a ready interface for the mobile identification service so TAMEb clients could just make a deal with an operator and start using it effortlessly with only minor configurations. [2.]

1.2 Tools & IDE

The software is programmed in Java and Rational Application Developer was used for programming which is an Eclipse based integrated development environment (IDE). TAMEb would have supported the C language too but Java was chosen because of the developer's professional capabilities.

The main reason for using RAD as the IDE instead of open source alternatives such as Eclipse is the support for WebSphere Application Server. This way the testing of the software is possible while programming and a lot of time is saved. Otherwise the software would have to be imported from the IDE, transferred to the server's OS (Linux in this case) and then deployed to the server, which is the final stage of testing.

The final testing part is done on virtual Linux where TAMEb 6.1.1 is installed. The Linux operating system in question is Red Hat.

1.2.1 WebSphere Application Server

At the beginning of the project the two server choices which were considered were Apache Tomcat and WebSphere application server (WAS). WAS is quite a heavyweight software application server and might be a little too resource consuming for a small test application like this. It could have been easier to use the light Tomcat to build the test application but as the finished solution is meant to be used on WAS it was decided that WAS it is. This is also why the IDE was chosen. The Rational Application Developer (RAD) supports WAS and this made the testing while programming more comfortable.

At first WAS 6.0 was used for testing but the libraries used in the project had been compiled with a newer version of Java compiler so a Java Runtime (JRE) update was necessary. This could have been done with a Java fixpack for WAS 6.0 but since the installation package for "WAS 7.0 for Rational application developer" was so easily obtainable the new server was installed to the IDE. Java versions are always updated in the new WAS versions so this fixed the library problem.

As illustrated in figure 2 the WebSphere application server provides a graphical interface called administrative console for handling the configuration of WAS. This includes the security configurations for Secure Socket Layer (SSL) too so long command line SSL configuration commands were not necessary. It required some time to learn how to use it but shortly after it became an invaluable asset. All the configurations of the SSL key stores and certificates can be done here. More detailed information about SSL can be found in the later chapters.

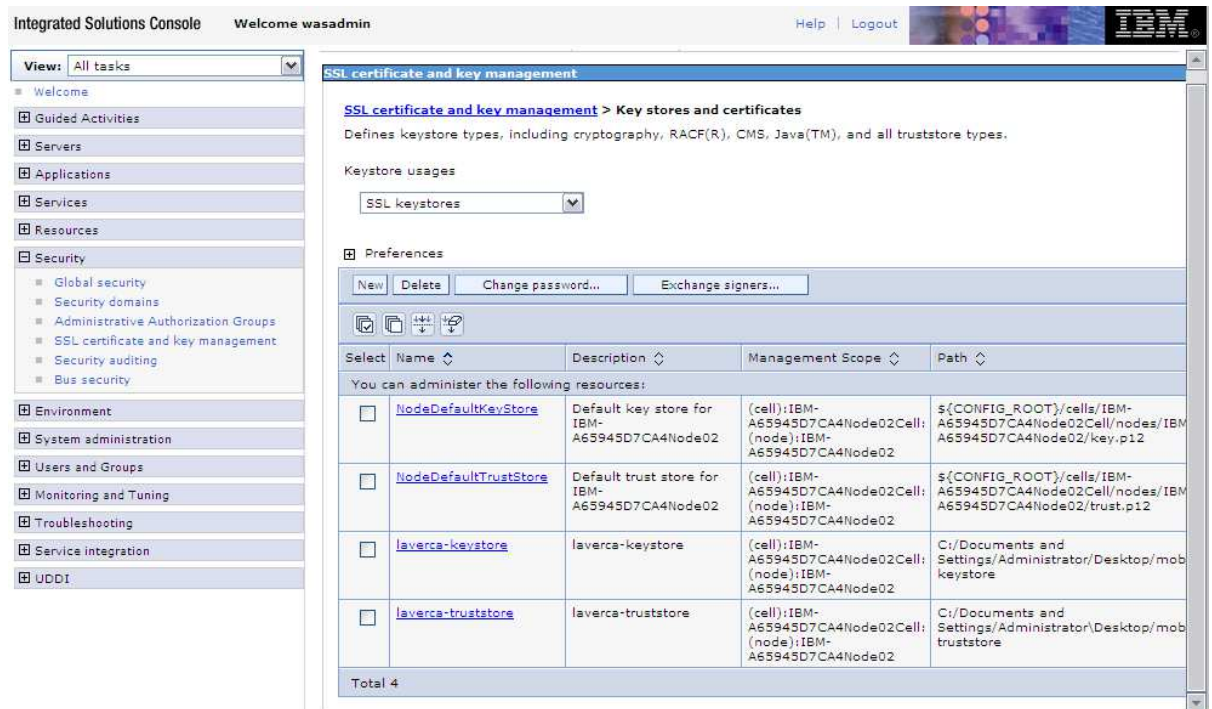


Figure 2: WAS administrative console

After building the program in the IDE, a .war file can be exported from the IDE and the .war file can be deployed to the WAS. The .war contains all the necessary files for the software to work.

1.3 TAMEb

Tivoli Access Manager is an IBM Security Systems product family that uses the same engine for authorization and authentication. The family consists of Tivoli Access Manager for e-business (TAMEb) and Tivoli Access Manager for Operating Systems (TAMOS).

Tivoli Access Manager for e-business is used for single sign-on purposes for web applications. This means that TAMEb is in charge of managing user identification and authentication to protect resources. The idea of single sign-on is that there may be several resources which are somehow related but still isolated. With single sign-on the user can login only once and then have access to all these resources. [3.]

TAMeb and TAMOS both have two core components and the rest of the services are built on these two components. The components are a user registry and a service for authorization. The service is made of an authorization engine that makes the authorization decisions and an authorization database.

In this work TAMeb is installed on 32bit Linux but it supports multiple other operating systems as well.

TAMeb is built of several components and in this chapter a brief explanation will be given of the parts that are related to the integration.

1.3.1 GSKit

Global Security Kit (GSKit) is the component that enables SSL encryption between Tivoli Access Manager and the supported registry servers. It also provides a tool called iKeyman which can be used to easily manage the SSL key pairs, certificate requests and key stores. GSKit is required before installing most of the other TAMeb components, and even if it was not, the MSS service is based on SSL as well so GSKit is required.

1.3.2 LDAP

The Lightweight Directory Access Protocol (LDAP) user registry is provided by IBM Tivoli Directory Server. Tivoli Directory Server is one of the main components of TAMeb and it uses IBM DB2 database as a data store.

1.3.3 WebSEAL

WebSEAL is an HTTP reverse proxy server which applies the security policies to the resources which are meant to be protected. It is a resource manager which is in charge of the protection of the resources assigned under TAMeb. WebSEAL is the most important part of the TAMeb and TAMeb is often called unofficially just WebSEAL. [3;4.]

Figure 3 below gives an example of how non-customized WebSEAL can be used for authentication:

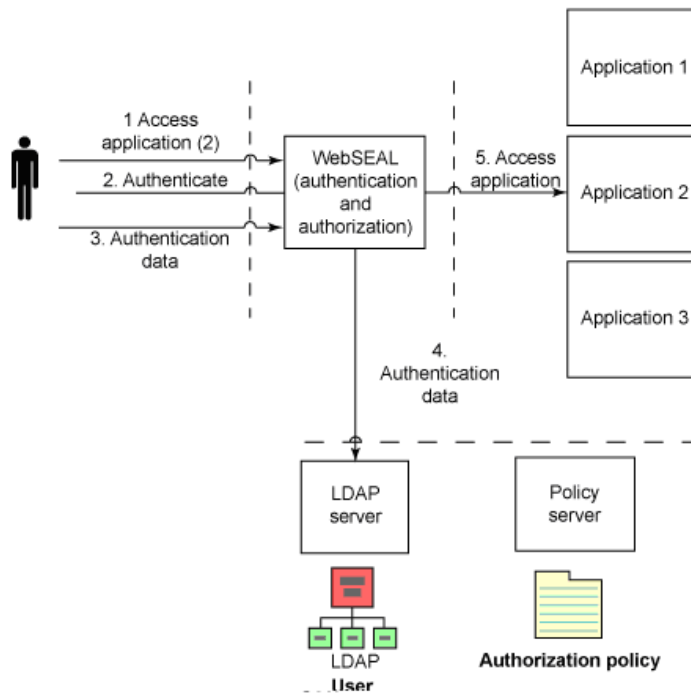


Figure 3: WebSEAL authentication [5.]

The following steps explain the steps in the figure and are directly from the same source. [5.]

1. The user attempts to access application 2.
2. WebSEAL consults its security policy to determine if the user should authenticate. It first checks the access control list (ACL) to see if the user needs to authenticate, followed by an (optional) (Protected Object Policy (POP)). If authentication is required, WebSEAL prompts the user to authenticate.
3. The user provides authentication information
4. WebSEAL checks the authentication information. In this case WebSEAL is checking against the LDAP (user and password)
5. If user is successfully authenticated, and is also authorized to access the resource, WebSEAL forwards the original request to Application 2.

WebSEAL can be customized in many ways and this was just one example of how it can be used. More detailed information of the WebSEAL can be found in the manual.

1.3.4 EAI

EAI stands for external authentication interface and as the name implies it is used to allow TAMEb to trust external authentication. WebSEAL creates the user credentials according to the results of this external authentication.

EAI plays a big role in this work so it deserves a little extra attention. The following graph will help in understanding how EAI works:

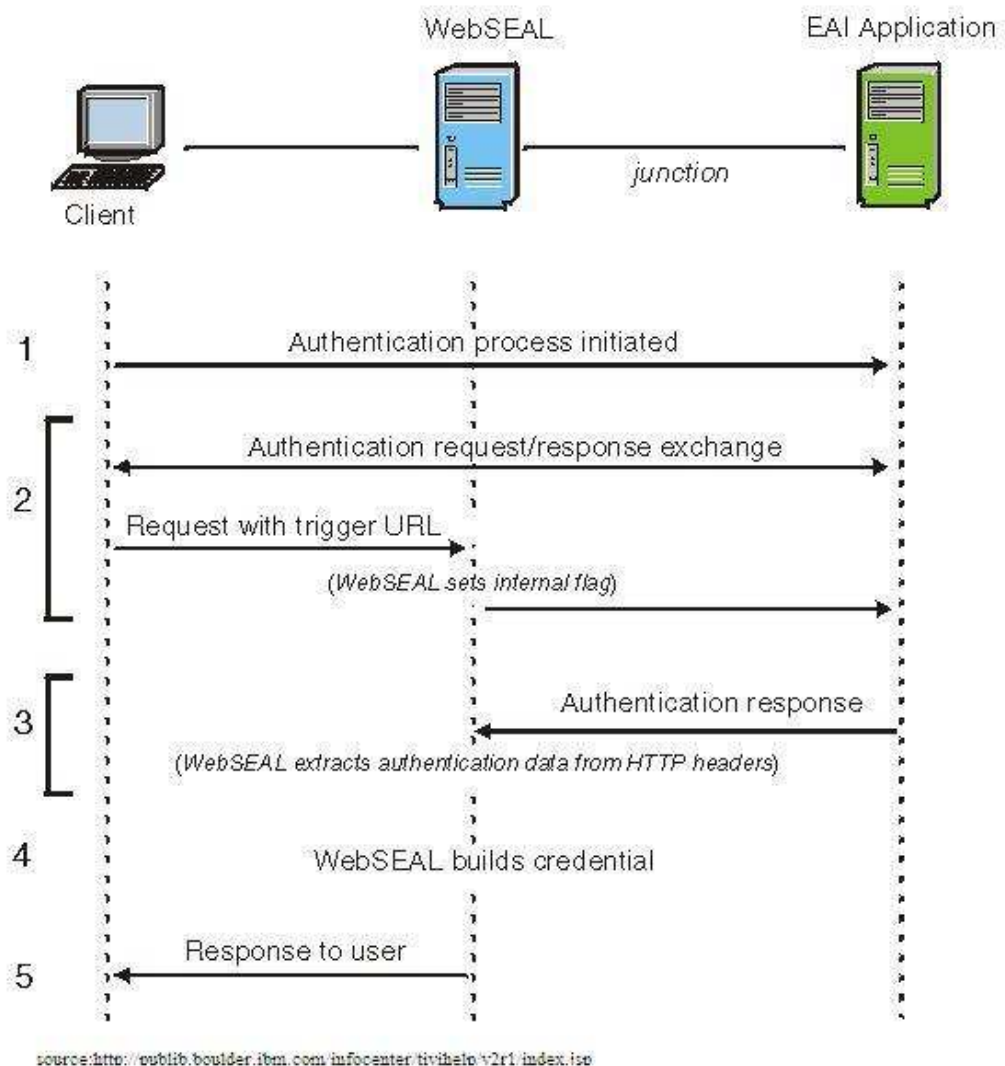


Figure 4: External authentication process [3]

- 1) As can be seen in figure 4, the authentication process initiates when a user without authentication requests a protected resource.

WebSEAL intercepts this request and redirects the user to a login.html page which has a login-form and a submit button which works as a link to the EAI application. The user then types the login information on the form and sends the information to the EAI application by pressing the submit button.

- 2) Next is the process of exchanging requests and responses between the EAI application and the client. The authentication may require several exchanges and the exchanges all go through WebSEAL. In these exchanges the EAI application gathers all the necessary information for authentication.

WebSEAL checks every one of these requests for a specified URL called the trigger URL. The trigger URL is configured in the WebSEAL configuration file and once it is found in the request it causes WebSEAL to look for authentication information in the following response.

After the EAI application has gotten all the necessary information, it authenticates the user and delivers the data to WebSEAL. The data is located in HTTP headers.

- 3) The response is received by WebSEAL along with the authentication data.
- 4) WebSEAL builds credentials according to the data.
- 5) A response is sent to the user which may be a "login success!" page or a redirection to the requested resource.

This is a typical way the EAI is used. There are plenty of other variations how to initiate the process, how the exchange goes and how the final response acts. [3.]

2 About Mobile Signature Service

The current mobile signature service is based on standards created by the European Telecommunication Standards Institute (ETSI). These standards, however, are based on older technology created by Mobile Electronic Signature Consortium known as the mSign.

The consortium released an XML based interface in the year 2000 which defined the protocol for service providers to acquire digital signature from the mobile device users. The consortium was originally made of 35 companies including Siemens and German mobile communication companies. Afterwards a mobile signature, also known as the MoSign project, was started by a group of the following companies: Deutsche bank, Ericsson, Materna, Microsoft, Sema group and Siemens. At the time the mobile signature was aimed at the WAP browser with a smart card reader. This project lead four German banks to announce that they would utilize the discoveries of the project and create a single standard for electronic signatures. Later on this lead into MSS specifications of ETSI which define the mobile signature service's XML interface and mobile signature roaming. Services working under this specification are found in multiple countries under jurisdiction of ETSI such as Germany, French, Hungary, Turkey and recently Finland as well. While every country in the Europe has accepted the law for electronic signatures it is not available in the United States. [6;7;8.]

2.1 General

MSS is an authentication mechanism which is used to add strong authentication for resources via personal mobile devices. It is important to understand that this doesn't limit the possible applications to just mobile applications but any applications requiring the user's consent to proceed with any kinds of transactions. Here is a list of possible solutions according to the mobile ID technology provider Valimo Wireless [9]:

- Secure online bank log-in and transactions
- Secure eCommerce purchases
- Sell and buy shares with non-repudiation
- Sign online credit and loan applications
- Sign corporate transactions

- Make ATM withdrawals without an ATM card
- Access secure eGovernment services such as tax payments, permits and voting
- Remotely access health records
- Conveniently access corporate networks (VPN)
- Sign documents such as PDF files and email
- Anonymous age verifying for restricted access
- Secure mobile contactless (NFC) payment account registration
- Top-up mobile wallets and other mobile applications

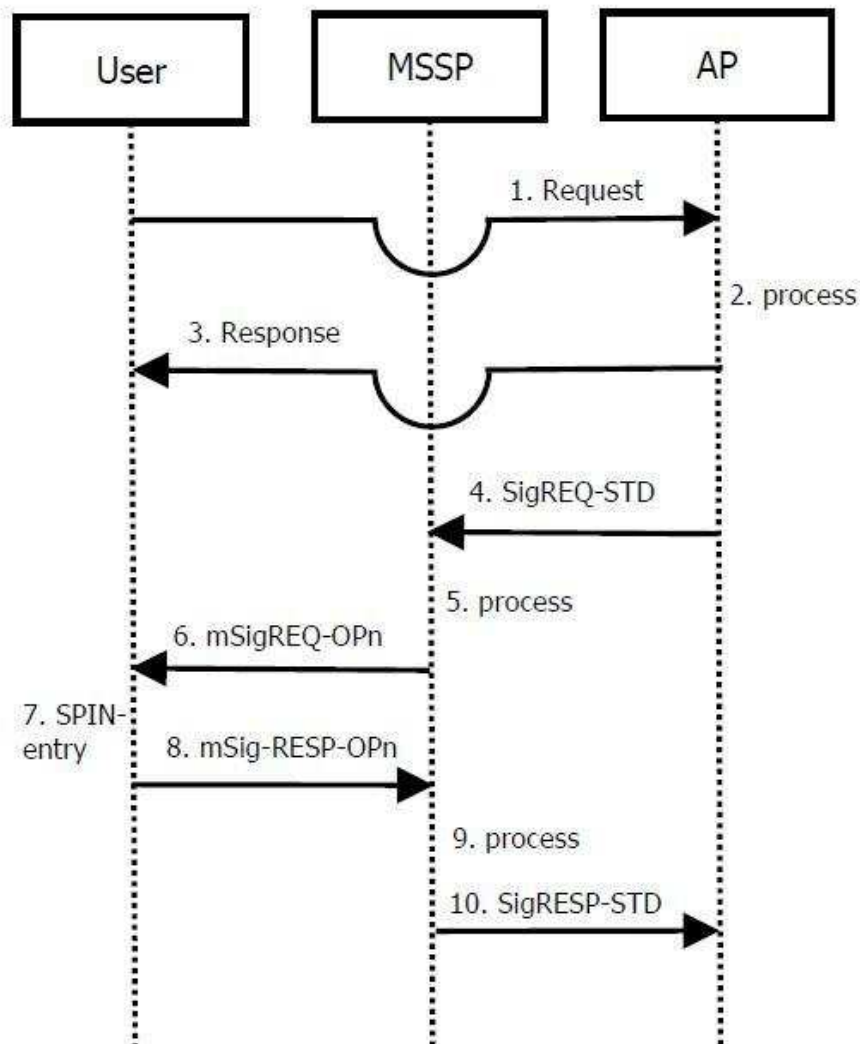
In this context the strong authentication is achieved by the process which requires the user to have a mobile device, registered mobile certificate on his/her SIM-card and a personal SPIN-code which is needed for authentication. After all these conditions are met, the operator can authenticate the user. This fulfills the requirements for strong authentication.

The etymology of the mobile signature service has its own complications. The standards and specifications provided by ETSI tend to use the "Mobile Signature Service" but the other version is Mobile certificate which translates to "Mobiilivarmenne" or Mobile ID. It is necessary to understand the difference which is that MSS is the actual service provided by the operators while Mobile certificate is the actual certificate on the user's mobile device that serves as an electronic identification. Essentially people tend to use all terms but still talk about the same service which can cause confusion.

The technology behind the mobile identification is provided by a company called Valimo Wireless which is a Finnish company founded in 2000 and bought by the Gemalto group in 2010. Valimo produces Valimo Mobile ID, and the ID is used around the world in different kinds of solutions. The Finnish "Mobiilivarmenne" was launched as a co-operation between Valimo and Elisa with Elisa using Valimo's Mobile ID technology. Valimo's mobile ID is a market leader all around the world in terms of installations and users. Valimo Wireless got partners all around the globe as well and the partner/customer list includes Ericsson, Elisa, TeliaSonera, Tieto, Hewlett-Packard, Mobitel Slovenia, Experian (providing MSS in France) and Telefonica-Spain so they have resellers covering most of Europe. [9.]

The mobile signature service which in Finland is called "Mobiilivarmenne" is the Finnish operators' competitor for TUPAS authentication which has dominated the field of strong authentication. The basic idea is fairly simple: The user tries to access a resource (some information or software or anything on the website), and if the resource is protected, the user is prompted for authentication. Now in addition to normal authentication with a username and password, the user will get a message to his/her mobile device asking to confirm that the user really is who s/he claims to be. [10.]

The following sequence diagram demonstrates a typical MSS use case scenario based on the ETSI standards.



source: http://docbox.etsi.org/EC_Files/EC_Files/ts_102204v010104p.pdf

Figure 5: MSS sequence diagram [11]

The steps in the diagram are as follows [11]:

- 1) User sends a request to AP to use a service
- 2) AP processes the request
- 3) User is informed about the MSS
- 4) Standard signature request is sent to the mobile signature method of the mobile signature service provider (MSSP).
- 5) Standard signature request is processed by MSSP

- 6) MSSP sends the mobile signature request to the user
- 7) User's mobile device shows the signature request and creates a digital signature after the user's confirmation
- 8) The mobile device returns the mobile signature response to MSSP
- 9) MSSP processes the response
- 10) MSSP sends the mobile signature response back to AP (with or without the signature)

This of course is just one scenario and there can be plenty of other variations depending on the service providers. For example, there can be external routing entities which complicate the process.

2.2 Business Possibilities

One important question about mobile identification is naturally where the money comes from. In the case of identification this becomes an issue since the TUPAS authentication is provided by the banks while the mobile identification is obviously provided by the operators. Currently, TUPAS authentication is dominating the field but it is starting to become outdated and alternatives or even replacements are becoming a necessity in the near future. If the mobile identification was to replace the TUPAS completely, the revenue would go from the banks to the operators. Figure 6 is of TUPAS usage and it shows the constant growth of the online identification business. Although the graph only shows the usage of TUPAS identification and not other means of online identification, Tupas is still the most used online identification method and the graph can be used to demonstrate the growth of online identification usage. [12.]

Tupas Identifications and Online Payments

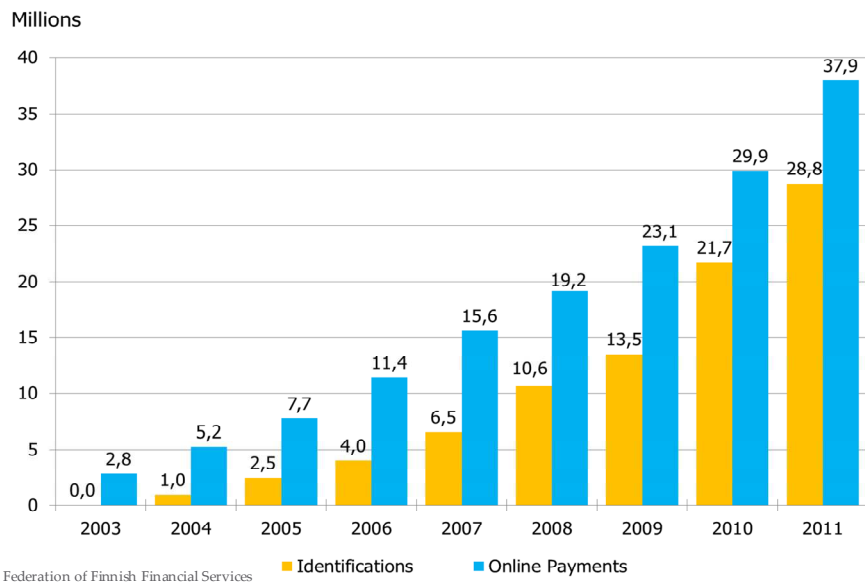


Figure 6: TUPAS usage [12]

This also brings in the question of safety and this is what banks need to be extra careful about. The technologies behind TUPAS and mobile identification are supposed to be equally safe because of the SSL technology which is used in the communications. In addition, the personal mobile device and one time password cards are almost equal in the level of security. This does not remove the fact, however, that the bank would have to trust an outside party to control the identification and authentication. The terms of the mobile signature service state that every party included in the transaction is responsible for the security of its own system. In case the authentication would somehow fail and a user would have access to someone else's information it would be the responsibility of the operator. This does not make it any better either in the bank's point of view because the bank must have trust in its services or the business fails.

Even with these facts in mind there are banks that are interested in using the mobile identification so it is important to find out why. There are a few key points to this. Of course one reason is that the revenue from the TUPAS authentication actually is not that big for the banks according to IBM specialists who work with the banks. While

they do bring some revenue, they do come with maintenance cost expenses. Outsourcing the authentication could be considered as eliminating these expenses and the maintenance resources could be used elsewhere. The bigger concern is the actual security and good usability that makes the online transactions easy enough, and this definitely is what mobile identification is about. It is easy to use and it is fast.

Another point is that the required one time password cards have resulted in negative feedback since some clients of the banks are corporations that have dozens of bank transactions per day and run out of passwords very frequently. Of course this could prove to be a problem if there was a delay with the delivery of the one time password cards. With the mobile identification this would not be a problem anymore.

The third point is the possibility of mobile signatures. This would be a major upgrade to the current system and would save tons of office paper. Forms, contracts and such could be signed digitally and the necessary information would not have to be printed anymore. In the worst cases when considering loans, there are people who need a loan, people who guarantee the loan, people who grant the loan, and everyone gets identical pieces of paper to be signed and plenty of copies are required. Mobile signature would decrease the need for paper drastically since the signature could be made digitally. The system could just send the signature request to each party's mobile device to get the party's consent. These are just a few of the possibilities that the mobile signature would provide. [13.]

2.3 SATU

The mobile signature service response can hold multiple fields of information about the user such as the name, phone number and street address, but the ones that require extra attention are the social security number HETU, which can be included in the response if there is a separate contract for it, and the electronic identity SATU. HETU and SATU are used for identifying the users.

The primary information for identifying a mobile certificate user is SATU which is an electronic identity given for Finnish citizens and foreigners who are staying permanently in Finland. SATU has been created because of the growing numbers of internet services and because the traditional social security number has been shown to

be too vulnerable for abuse on the internet. The reason why SATU is safer is that the actual number does not include any personal information about the person in question such as the age or gender.

The process of acquiring the mobile certificate feature on a mobile device requires that the user is identified with a passport or similar identification. This is because the information is also used to get the SATU from the Population Register Centre by the operators, and this adds one more layer of extra security for the system. [14;15.]

2.4 ETSI-standard & FiCom-Implementation Guideline

As the electronic commerce has been growing rapidly the necessity of appropriate information security has been growing just as much. Electronic commerce utilizes new ways to do business and this is the reason why the safety of electronic interactions becomes an issue.

The purpose of an electronic signature is to have an electronic alternative for good old hand-written signature and for it to have same kind level of trust. The ETSI standard mentions that the electronic signatures can only be as good as the technology and processes used to create them. This is why standards such as ETSI are necessary to create "a common level of confidence and acceptance". [11.]

One issue about mobile signature is that it requires equipment with some amount of computational resources usually a smartcard or such. As stated in the ETSI standard it is unlikely that consumers would like to get additional equipment such as card readers or anything like that. This is where mobile phones come into the game since most of the people in European countries carry them so the people automatically have the necessary smartcards too. The term for electronic signatures is *mobile signatures* when dealing with mobile devices. The fact that it is called mobile signature makes it even more attractive for business since it boosts the existing technology and opens the doors to one billion mobile phone users across 179 countries.

In addition to ETSI's definition of mobile signature, the standard specifies mobile signature design criteria and the necessary technology for implementation. It also

provides use cases of typical solutions and services that could utilize the mobile signature service. It also shows how the signature process goes.

Other points that ETSI specifies are the interfaces that users see. The point of this is to create a unified user experience for any MSS user. Figure 7 demonstrates the user experience.

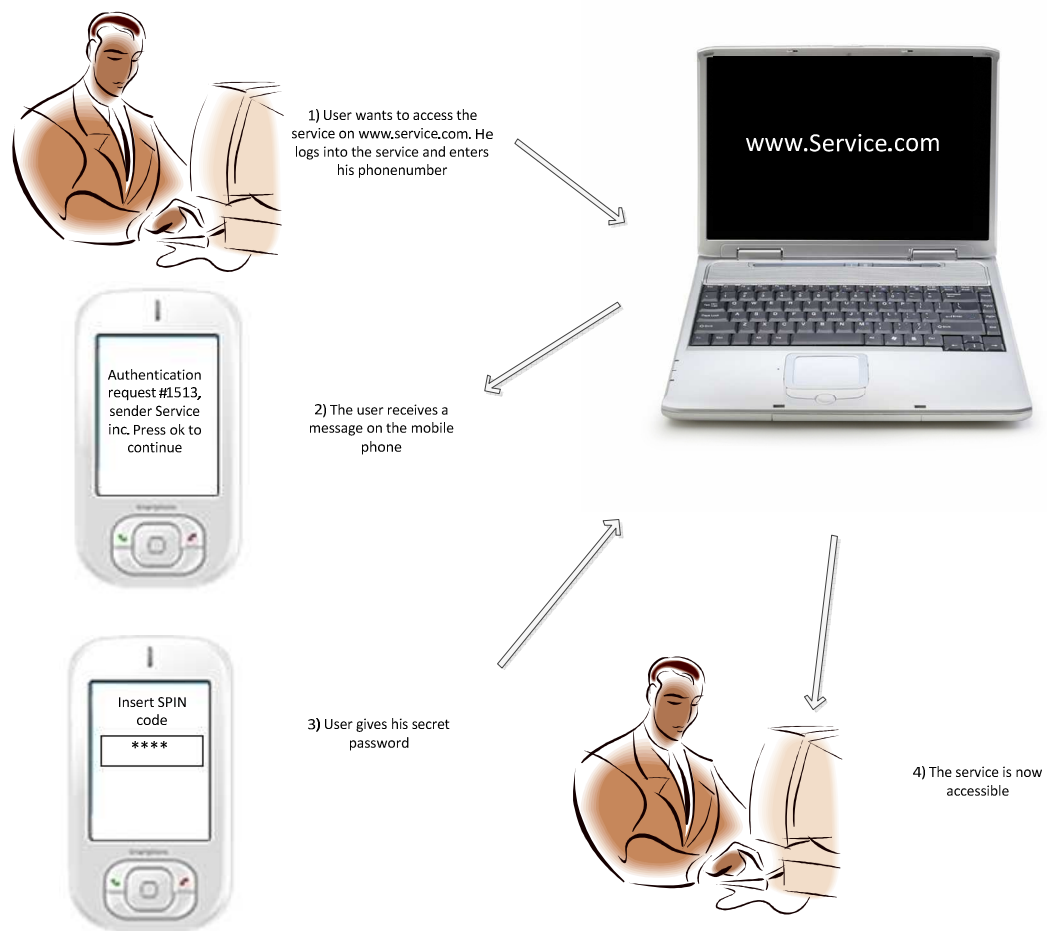


Figure 7: User experience

FiCom documentation is a Finnish interpretation of the ETSI standard and it is made by FiCom ry. Mobile Signature in Finland is a service implemented on the SIM cards and it is provided by Elisa, Sonera and DNA which are all members of the FiCom ry. It works with nearly any cellphone and it does not require a specific operator. [11;16.]

ETSI also provides the necessary libraries for handling the MSS process. The `org.etsi.uri.TS102204.v1_1_2` library provides the necessary functionality in a Java environment. The functions are explained in greater detail in the solution section.

The FiCom documentation states how the service should be used and implemented by the Finnish application providers (AP). It is the main document the operators will refer to while the implementation is in progress. The documentation goes into a great detail about the mobile signature, architecture, information security, user experience and mobile signature service provider (MSSP) interface. It also provides a list of the error messages helping with the debugging process. It is based on the the ETSI TS102204 and TS 102207 standards which specify the application provider interface for mobile signatures, mobile user identification and the mobile signature roaming between different operators' networks. Figure 8 shows how the network roaming works.

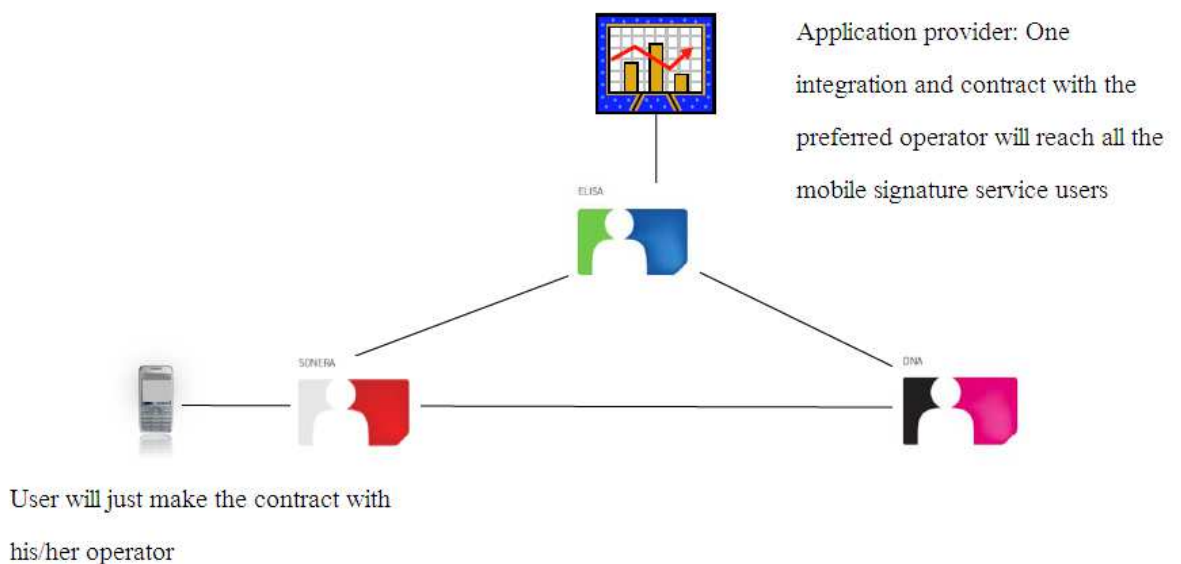


Figure 8: Network roaming [17.]

The most important parts about the FiCom implementation guideline are mentioned in the briefing of the document. Here are the keypoints of the briefing:

- Supported messaging modes: asynchronous client-server and synchronous client-server (not recommended).
- Strong authentication between all parties included in the messaging.

- Application provider ID (AP_ID) and application provider name shown on the mobile device will be agreed on a contract between the application provider and the operator. After these the application provider password (AP_PWD) will be created and the AP will own the ID even if the service is moved from the current operator to another.
- Supported message formats specified in the ETSI standard: MSS_SignatureReq, MSS_SignatureResp, MSS_StatusReq, MSS_StatusResp, MSS_ReceiptReq, MSS_ReceiptResp.
- The FiCom guideline does not specify MSS service registration messages and the registration process is considered to be an internal matter of the mobile signature service provider.
- XML-signed service messages are not supported.
- The identification of the user and the user's operator is done with the phone number.
- Supported character sets: UTF-8, GSM, UCS2.
- Services offered by the operator: anonymous authentication, authentication, signature of plain text document, signature of digest content, issuing consent, operator service for authentication.
- The offered signature services all have their own signature profiles.
- Unified user experience such as the one mentioned above with the ETSI standard, mainly concerning the signature requests.
- The digital signatures are in a base encoded64 PKCS#7 or PKSC#1 format
- Additional services: no spam code, transaction ID, language preference.

The first thing mentioned at the beginning is the asynchronous client-server messaging mode. In this mode the signature process consists of service messages: signature request, status request which follows the state of the request process and the status response which is the response to the latter. The receipt request and responses are optional. The signature process consists of multiple sequential HTTP sessions. These

sessions are started by the application provider. According to the FiCom documentation, this mode is useful compared to the synchronous mode because AP receives the reference information for the signature event at an early phase of the event and the service system can use all the resources it has to provide better service reliability.

The documentation recommends that the first status request should be sent 20 seconds after the signature request and this should be repeated every five seconds. This way there will not be too much useless traffic before the final status response which is the end of the signature process.

One of the key purposes of the FiCom implementation guideline is to define the MSS service messages. The messages are sent in SOAP envelopes. SOAP, which stands for Simple Object Access Protocol, is an XML-based communications protocol for web programs that are running on different computers. It is used for exchanging information between services that run on the web. [18.]

These SOAP envelopes include the envelope header (env:Head) and body (env:Body). The header is optional since it is used for the aforementioned XML signatures and as mentioned earlier the FiCom documentation does not support these signatures.

The body of the SOAP envelope, on the other hand is mandatory. It defines the message type of the envelope. As mentioned above the types are the following: MSS_SignatureReq and MSS_SignatureResp and the other MSS_Signature messages. These types have their own specific information attributes and elements. If there are any errors in the process, the system will return a status code for the AP. These codes are SOAP FAULT codes and each status code is defined at the end of the FiCom documentation. Figure 9 shows an example of the message from the FiCom document: [2.]

```

POST /MSS_Signature HTTP/1.0
Host: mss.teliasonera.com
Content-Type: application/soap+xml; charset="utf-8"
Content-Length: ...

<?xml version="1.0"?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <!-- Optional header -->
  <env:Header>
  </env:Header>
  <!-- Mandatory body -->
  <env:Body>
    <!-- WDSL op name -->
    <MSS_Signature>
      <!-- MSS service message -->
      <MSS_SignatureReq ...>
        .
        .
        .
      </MSS_SignatureReq>
    </MSS_Signature>
  </env:Body>
</env:Envelope>

```

Listing 1: Example of message structure

Different message types are all in the same format but they have their unique attributes. At the beginning of the MSS event the MSS_SignatureReq is sent and it holds information that will be referred to later on in the event by other messages. MSS_SignatureReq holds the information about:

- Messaging mode
- Timeout time
- Application provider ID (AP_ID)
- User contact information (phone number)
- Timestamp
- Signed content (data to be signed (DTBS))
- Signature profile which defines the desired service
- Additional services such as a spam prevention code and the user's information for example social security number and electronic ID SATU. Getting the social security number requires a separate agreement with the operator.

These attributes are defined in the service messages as the following example demonstrates:

```
<MobileUser>  
  <MSISDN>+358123456789</MSISDN>  
</MobileUser>
```

Listing 2: Service message attribute example

The MSS_SignatureResp holds the end status of the event, additional service response messages, and if the signature is successful it also holds the digital signature. The MSS signature itself is either a PKCS#1 compliant signature with the user's mobile certificate or a PKCS#7 compliant signature message. The rest of the attributes in the response are the same as the ones in the signature request.

The status requests are sent continuously until the signature response is received and the status responses follow.

2.4.1 Information Security

The MSS process includes using the user's phone number. Afterwards the user identity can be confirmed by SATU or HETU which means information security requires extra attention to prevent any abuse of the user's personal information. The FiCom documentation presents a few different problem cases and possible solutions for them.

Also all the traffic between the solution and the operator's service is protected with SSL. SSL will be covered in more detail in the chapter describing the solution.

2.4.2 Spam Prevention Code

One of the problems mentioned in the FiCom implementation guideline was that if the AP interface only asks for the phone number, the user can try to send the signature request to someone else's mobile device. It is very unlikely that a smart user would sign such a request but such spam messages could still cause some damage for example to the public image of the application provider or the operator. The spam

prevention code would prevent events where a user would send the signature request to someone else's mobile device.

The spam prevention code is asked along with the phone number. It is at least three characters long and consists of both numbers and characters. This code is provided to the user by the operator. [2.]

3 Solution

3.1 High Level Structure

The solution developed in this project merges MSS into the TAMEb system and delivers the MSS results to the application provider's service using TAMEb. This works out with the EAI which provides the option for TAMEb to trust external authentication methods. The point of the testing is to acquire the user's HETU from the operator and deliver it to the end application.

The SSL configurations and the UI of the solution are rather hastily developed but they are not the primary concern. After the client installs TAMEb to protect the web resources, the SSL connections from the website to TAMEb and the website to operator will have to be re-made, so this is an issue for the clients to handle with IBM and the operators. The clients most likely will not use the plain default UI either so they will have to create their own UI to represent their own brand.

The following figure helps to understand the solution on a high level:

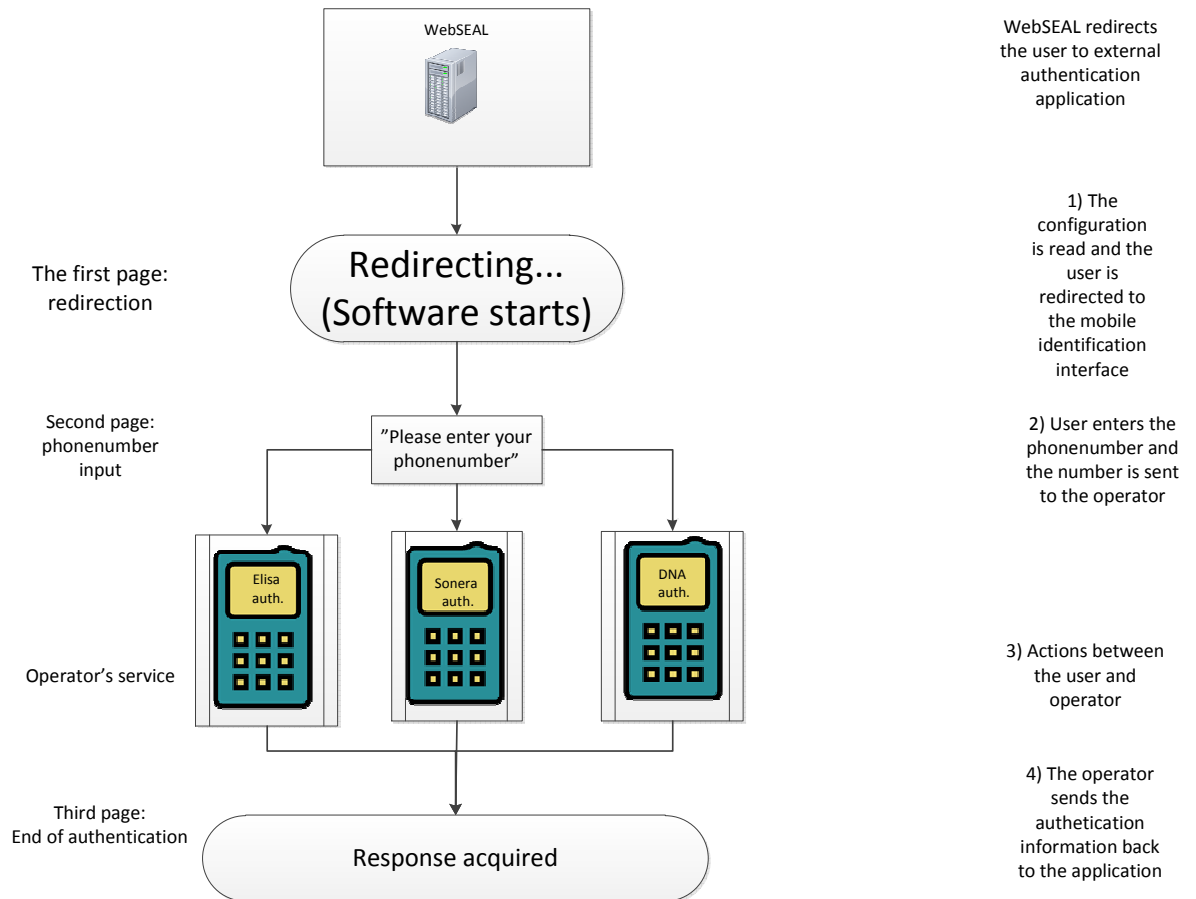


Figure 9: High level structure of the solution

Figure 11 shows the high level structure of the solution. It shows that the user is redirected to the solution by trying to access a protected resource and after the mobile signature is completed the information is sent back to the solution and then back to the WebSEAL. Figure 12 explains the steps in greater detail. Here is the sequence diagram describing the solution:

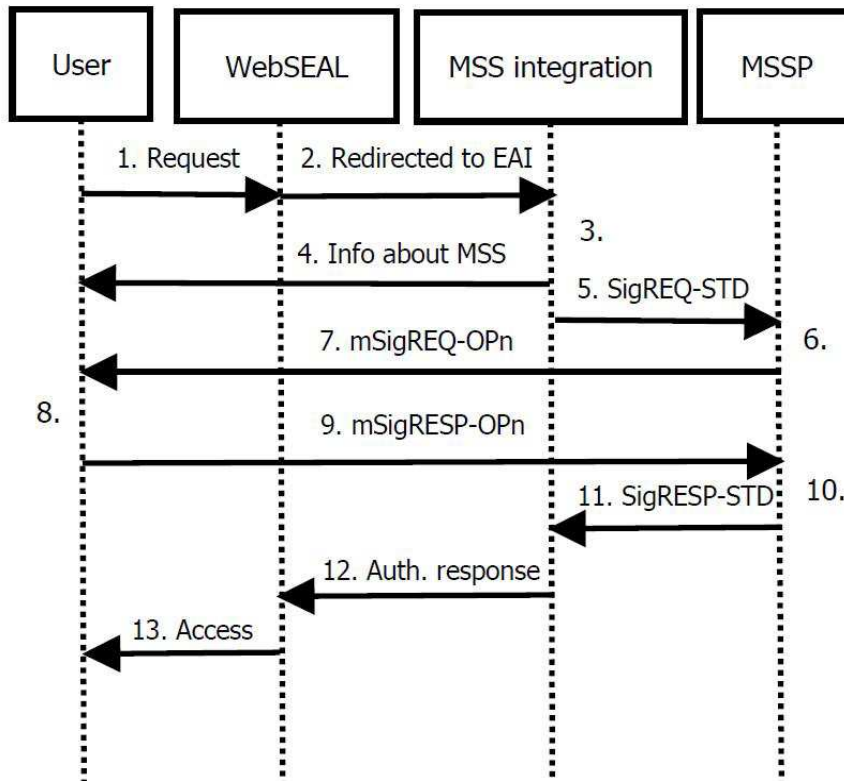


Figure 10: MSS integration sequence diagram

The steps are a combination of the earlier EAI and MSS cases:

- 1.-2. The process starts when the user tries to access a protected resource on a website. The user is not authenticated so WebSEAL takes action, but since EAI is used in this solution, the client is forwarded to external authentication.
3. MSS integration processes the request.
4. The user is informed about the initiation of the MSS.
5. The MSS integration sends a standard signature request to the mobile signature service provider (MSSP).
6. The MSSP processes the request.
7. The MSSP sends a mobile signature request to the user
8. The user enters the SPIN code.
9. The user sends the mobile signature response back to the MSSP.

10. The MSSP processes the response.
11. The MSSP sends the standard signature response back to the MSS integration.
12. The MSS integration sends the authentication information to WebSEAL.
13. WebSEAL builds the credentials and gives access to the user accordingly.

The solution also returns the user's social security number along with the authentication information and WebSEAL delivers it to the end application.

Another viable option would be for the solution to actually create the finished credentials with TAM's API which provides a set of classes for this and sends the credentials back to WebSEAL. Then WebSEAL would just use these credentials and deliver the social security number as it did in the previous case.

3.2 Laverca API

Laverca is an open source API for Java which includes the necessary ETSI and FiCom functionality to make the MSS development easier for application providers (AP). The main point of this API is to make the development process more efficient so that the functionality is easily available if the developers decide to use Java for implementation and the developers will not have to go too deep into the details of the ETSI specifications or SOAP implementations. The Laverca documentation mentions that especially the details of the Client-server asynchronous messaging (C/S async messaging) mode are hidden from the application developers to simplify the developing process.

As the project is open source, the entire source code is available if necessary and a pre-compiled .jar file is available as well. The package includes working samples about how to use the functions and documentation about the project. The operators support this API. Putting up the service is pretty straightforward with it.

The Laverca project is based on a few different documents. These documents define the architecture of Laverca. The documents are the FiCom implementation guideline, ETSI TS102204, SOAP 1.2, HTTP 1.1 and SSL/TLS. With the help of Laverca API these

documents did not have to be studied very carefully to create a working test application. [19.]

Here is a graph of the Laverca architecture.

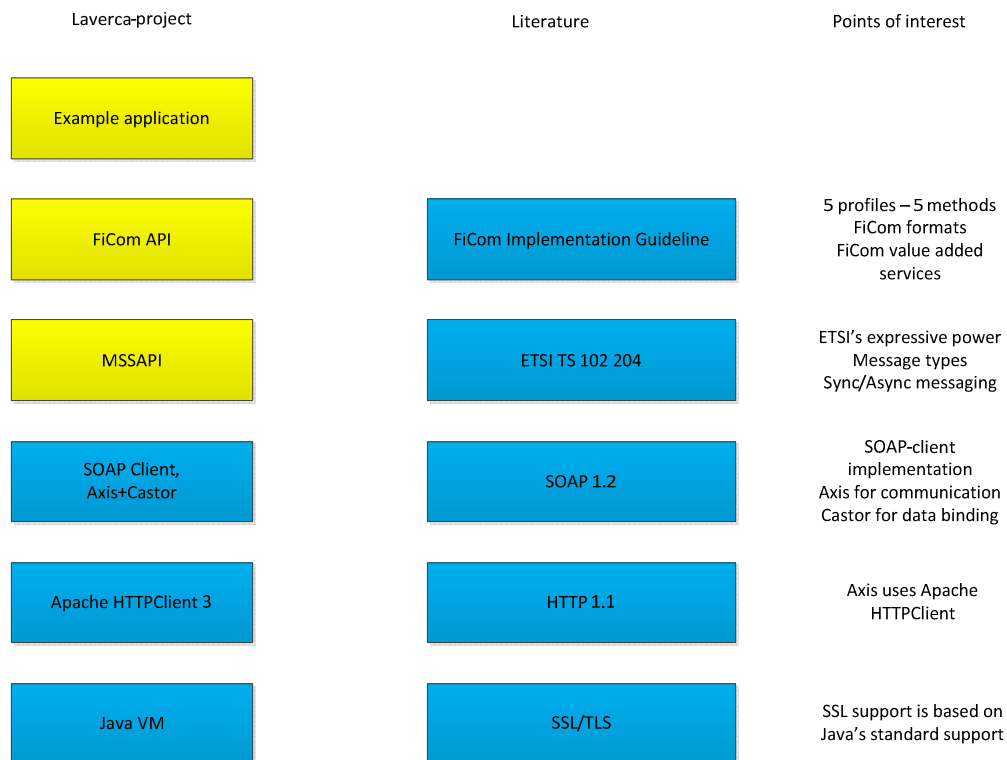


Figure 11: Laverca architecture [19]

For quick development the most interesting part of the architecture is the FiCom API which is also called the FiCom layer. It offers a few functions which are the same as the five signature profiles in the FiCom implementation guideline. The functions are listed as follows [19]:

- Authenticate()
- authenticateAnon()
- signText()
- signData()

- consent()

The reason for this layer is to provide high level functions for the developers and hide the details of the ETSI standards most importantly in the C/S async messaging mode. Here is an example of the FiCom layer source code:

```
import fi.laverca.*;
import org.etsi.uri.TS102204.v1_1_2.Service;
...

String apId = "http://my_ap_id";
String apPwd = "my_ap_pass";
String sigUrl = "https://www.myae.fi/soap/services/MSS_SignaturePort";
String statUrl= "https://www.myae.fi/soap/services/MSS_StatusQueryPort";
String statUrl= "https://www.myae.fi/soap/services/MSS_ReceiptPort";

FiComClient fiComClient = new FiComClient(apId, apPwd, sigUrl, statUrl, recUrl);
.. setup SSL

..
String phoneNumber = .. ask user for it ..
String nospamCode = .. ditto ..
String apTransId = .. generate unique id for this transaction ..
String challenge = .. generate authentication challenge ..

Service noSpamService = FiComAdditionalServices.createNoSpamService(.., false);

try {
    FiComRequest req = fiComClient.authenticate(apTransId,
                                                challenge,
                                                phoneNumber,
                                                noSpamService,
                                                null, // additionalServices,
                                                new FiComResponseHandler() {

            @Override
            public void onResponse(FiComRequest req, FiComResponse resp) {
                log.debug("got resp");
                log.debug(resp.getPkcs7Signature().getSignerCn());
            }

            @Override
            public void onError(FiComRequest req, Throwable throwable) {
                log.debug("got error", throwable);
            }

        });
}
catch (IOException e) {
    log.debug("error establishing connection", e);
}
```

Listing 3: Example of FiCom layer source [19]

As seen in the source code the necessary information which is also used to define the operator such as the application provider ID, password and SOAP tools are all put into variables and the FiCom client is created with these variables. Then the user's phone number and additional services are put into the variables and `fiComClient.authenticate` is called with these values. The response functions are called in a loop which sends the status requests. When the signature response is received, the function is called.

The ETSI layer forms the functionality of Laverca. The ETSI TS 102204 datatypes and SOAP calls are made here. Listing 4 shows an example of how this layer can be used to generate a synchronic signature event.

```
import fi.laverca.*;
...

String apId = "http://my_ap_id";
String apPwd = "my_ap_pass";
String sigUrl = "https://www.myaefi.fi/soap/services/MSS_SignaturePort";
String statUrl = "https://www.myaefi.fi/soap/services/MSS_StatusQueryPort";
.. all the other URLs

EtsiClient etsiClient = new EtsiClient(apId,
                                     apPwd,
                                     aeMsspIdUri,
                                     msspSignatureUrl,
                                     msspStatusUrl,
                                     msspReceiptUrl,
                                     msspRegistrationUrl,
                                     msspProfileUrl,
                                     msspHandshakeUrl);

String apTransId = "A"+System.currentTimeMillis();
String msisdn = "+35847001001";
DTBS dtbs = new DTBS("sign this", "UTF-8");
String dataToBeDisplayed = null;
String signatureProfile = FiComSignatureProfiles.SIGNATURE;
String mss_format = FiComMSS_Formats.PKCS7;
MessagingModeType messagingMode = MessagingModeType.SYNCH;

MSS_SignatureReq sigReq =
    etsiClient.createSignatureRequest(apTransId,
                                     msisdn,
                                     dtbs,
                                     dataToBeDisplayed,
                                     signatureProfile,
                                     mss_format,
                                     messagingMode);

MSS_SignatureResp sigResp = null;
try {
    sigResp = etsiClient.send(sigReq);
}
catch (AxisFault af) {
    log.error("got soap fault", af);
    return;
}
catch (IOException ice) {
    log.error("got IOException ", ice);
    return;
}
}
```

Listing 4: Example of ETSI-layer source code[19.]

The Laverca API uses multiple different open source libraries such as Apache Axis and Castor for SOAP client implementation. Axis also uses Apache HTTPClient. Axis handles the communications and Castor is used for Java-to-XML binding which is required because of the XML based MSS messages. [20; 21.]

The implementation of SSL was done in Laverca's class called `JvmSsl` as shown in listing 5. It is based on the basic Java SSL support. The class is really simple and easy to use:

```
public class JvmSsl {
    /**
     * Sets JVM global SSL settings. This needs to be called only
     * once, before any EtsiClient or FiComClient objects are
     * used. The method tries to open the stores in order to
     * validate the parameters.
     * <p>
     * Note: This is STATIC, and uses global JVM settings.
     *       This can conflict with settings done by some other
     *       code, or by runtime settings.
     *
     * @param trustStore the keystore file containing trusted SSL server certificates.
     * @param trustStorePassword the password to the truststore.
     * @param keyStore the keystore containing your SSL client certificate.
     * @param keyStorePassword the password to the SSL client keystore.
     * @param keyStoreType either "JKS" (Java native) or "PKCS12".
     * @exception IllegalArgumentException if the stores can not be opened.
     */
    public static void setSSL(String trustStore,
                              String trustStorePassword,
                              String keyStore,
                              String keyStorePassword,
                              String keyStoreType)
        throws IllegalArgumentException
    {
        System.setProperty("javax.net.ssl.keyStoreType",    keyStoreType);
        System.setProperty("javax.net.ssl.keyStore",        keyStore);
        System.setProperty("javax.net.ssl.keyStorePassword", keyStorePassword);

        if (trustStore != null) {
            System.setProperty("javax.net.ssl.trustStore",    trustStore);
            System.setProperty("javax.net.ssl.trustStorePassword", trustStorePassword);
        }
    }
}
```

Listing 5: `JvmSsl` class

Figure 12 shows all of the Laverca classes. The classes are not documented and require the user to look into the source code if there is anything unclear about how to use the specific functions of Laverca.

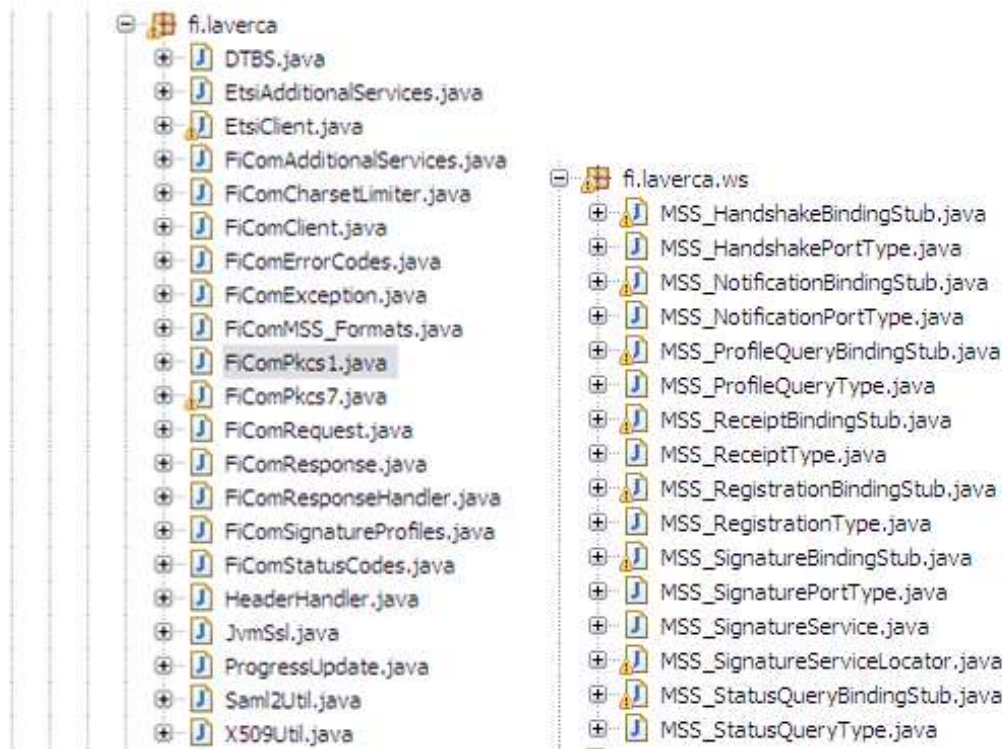


Figure 12: Laverca classes

As it can be seen in the previous figure, the Laverca library provides functions for all the necessary actions mentioned in the FiCom implementation guideline. All the functions for data to be signed, additional services, the FiCom client, ResponseHandling and security are handled by the library.

3.3 Features

The solution includes an interface for DNA, Elisa and Sonera so that the buyer of TAMEb can choose which operator to make a deal with. After the deal with one operator is done, the buyer can still have clients using other operators because the operators have mobile signature roaming deals with each other and the clients are directed from the buyer's service to their respective operators for the signature. The selection of the operator will be done in a configuration file. The configuration file is read when the program is initiated and the configuration will define to which operator the user is guided to.

Secondly, the solution uses the spam prevention code which was introduced earlier. The spam prevention code is a code given to the user by the operator. This feature is provided by the operator, and the required functions are found in the Laverca project.

3.3.1 SSL

The connection between the solution and the operator's service is protected with SSL. While testing, the whole certificate chain was acquired in a zip file, but in the final version the certificates would have to be requested from a certificate authority.

The application requires a few different services from the operator, so three .crt files were required to verify the authenticity of webservices. In addition, there is the .p12 file for encrypting the data to be sent. These values vary a little between operators. Supported key store types are PKCS or JKS.

In the WAS environment the .crt files can be put into the trust store with the administrative console. After logging in, the security section can be chosen on the left-hand side of the page. After that the key and trust stores are found by choosing the SSL certificate and key management and then finally Key stores and certificates. The .p12 file must be imported to the key store.

The key and trust store paths and passwords are given to the application. The Laverca API provides a JvmSsl class which uses Java's standard SSL support. This class has a method for setting up the SSL. In practice this means setting up the SSL identity with environment variables and trusting the operator's SSL certificates.

3.4 Configuration

The solution includes a configuration file for choosing the partner operator. The configuration is straightforward and easy to use. The user can choose which operator to make a deal with and then just make a few simple changes in the configuration files.

The config.properties file is located at the project root directory. There is a ReadMe.txt file in the same directory which states how the solution should be configured as the figure 13 illustrates:

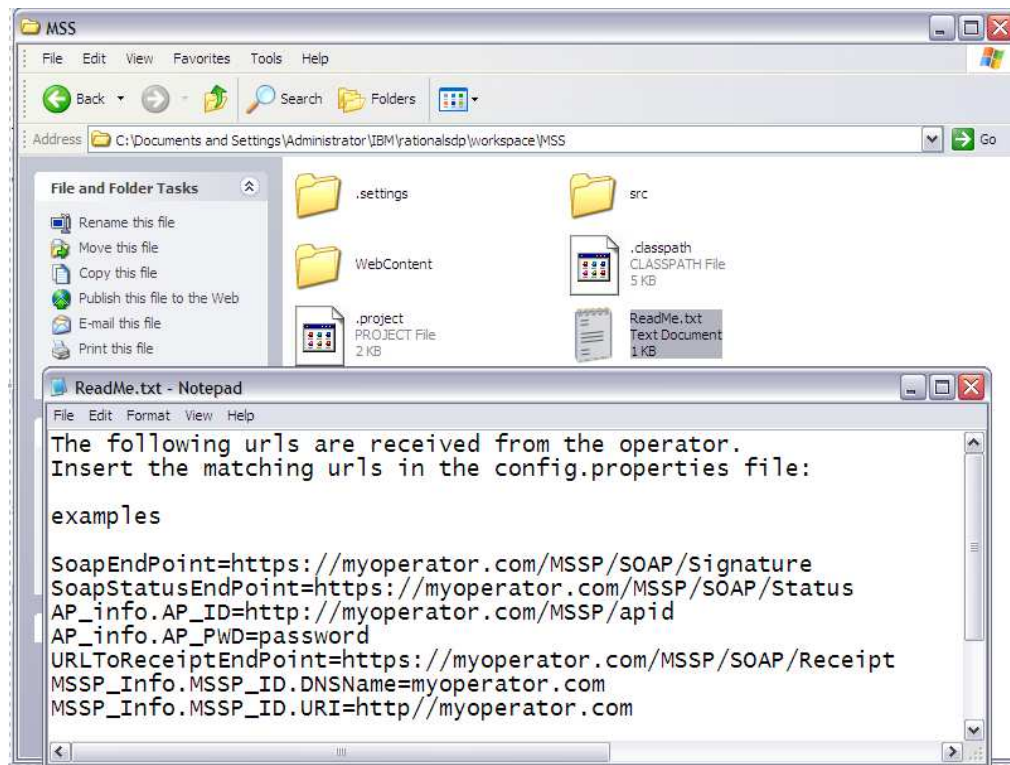


Figure 13: Configuration of ReadMe.txt

The URLs in the ReadMe file are just examples and will not work as they are. The real URLs are acquired from the operators.

3.5 Source Code

The service is a dynamic web project and it uses a servlet as the controller. The project consists of a few .jsp pages, the servlet and a few classes which handle the configuration, SSL setup and the classes for an MSS request and response handling. The servlet itself is quite simple with only a few lines of code as shown in Listing 6. [22.]

```

RequestDispatcher rd =
request.getRequestDispatcher("waitingForResponse.jsp");

rd.forward(request, response);

PrintWriter out = response.getWriter();

SSL ssl = new SSL();

```

```

        ssl.setupSSL();

String phonenumber = request.getParameter("puhelinumero");

        MSSrequest req = new MSSrequest(phonenumber);

        ResponseHandler resp = new ResponseHandler();
        req.sendRequest(resp);

```

Listing 6: Servlet

After the user presses submit on the login.jsp page, the program goes to the doGet-method(above), takes the phonenumber and creates the SSL object and calls the setupSSL method. This sets up the SSL. After that the MSSrequest object is created with the phone number parameter along with the ResponseHandler object. The ResponseHandler object is given to the request object as a parameter in the sendRequest method. This way the servlet has an access to the response parameters.

3.5.1 SSL Class

This class is in charge of setting up the SSL using the fi.laverca.JvmSsl class from the Laverca library as seen in listing 7.

```

public void setupSSL() {

        JvmSsl.setSSL(trustpath, trustpass, keypath,
keypass, keytype);

        System.setProperty("sun.security.ssl.allowUnsafeRenegotiation", "true");}

```

Listing 7: SSL Class

The usage is very simple as it gives functions called setSSL and setProperty. The setSSL method requires the paths to the keystore and truststore as parameters. The certificates are stored in these stores. In addition the passwords to the stores must be

given along with the keytype which in this case is pkcs12. The JvmSsl class uses the javax.net.ssl properties to setup the SSL.

```

public static void setSSL(String trustStore,
                          String trustStorePassword,
                          String keyStore,
                          String keyStorePassword,
                          String keyStoreType)
    throws IllegalArgumentException
    {
        System.setProperty("javax.net.ssl.keyStoreType",
keyStoreType);
        System.setProperty("javax.net.ssl.keyStore",
keyStore);
        System.setProperty("javax.net.ssl.keyStorePassword",
keyStorePassword);

        if (trustStore != null) {
            System.setProperty("javax.net.ssl.trustStore",
trustStore);
            System.setProperty("javax.net.ssl.trustStorePassword",
trustStorePassword);
        }
    }

```

Listing 8: JvmSsl setSSL method

As illustrated in listing 8, the complete setSSL method is found in the JvmSsl class.

3.5.2 MSSrequest Class

This class is in charge of building up the request and sending it. It holds the necessary information which specifies the operator information. The constructor also gets the phone number as a parameter which is used in the sendRequest function. This function is the most important part of the class. The most important parts of the class are shown in listing 9.

```

private static String apId

```

```

private static String apPwd
private static String sigUrl
private static String statUrl
private static String recUrl

public String sendRequest(ResponseHandler resp){

        FiComClient fiComClient = new FiComClient(apId,
apPwd, sigUrl, statUrl,

                                                recUrl);

//EVENT ID
//No spam service
//Phonenumber
//Additional services

try {

        FiComRequest fiReq = fiComClient.call(apTransId,
authnChallenge,
phonenummer,
noSpamService,
eventIdService,
additionalServices,
FiComSignatureProfiles.AUTHENTICATION, //
Authentication profile
        FiComMSS_Formats.PKCS7,
resp);
        FiComResponse fiResp = fiReq.waitForResponse();

```

Listing 9: MSSrequest class

The operator specific variables at the beginning of the class are application provider ID, application provider password, signature URL of SOAP tools, statusport URL and receiptport URL.

3.5.3 ResponseHandler Class

When the request is sent, the Laverca library goes into a loop to wait for a response. The FutureTask class also provides a get function which waits for the response in the loop. This way the servlet can wait until the response is received and then go forward.

The ResponseHandler class provides methods for different cases in the loop which waits for the response.

After the request is sent, the LOOP keeps saying "processing..." until the final response is received or until the software timeouts.

When the final response is received, the OnResponse method is called and the encrypted message is digested. The resp.getPersonIdAttributes(); method is called to get the attributes in the List<PersonAttributes>. After this the log method is called to print the results.

In the final version of the software, the ResponseHandler is created in the servlet, and after the response is received, the ResponseHandler's getHETU() method is called and the HETU is returned to the servlet for processing.

4 Testing & Demonstration

The software was tested along with the software development process in the IDE. Small Java main applications were used in the process to test the features before implementation. As mentioned earlier the mobile signature service itself is based on XML, but with the Laverca library it can be developed with Java. The Laverca library provided all the FiCom specified error messages and the error messages were explained in detail in the FiCom implementation guideline. This made the testing and implementation simple. The functionality of the operators' service was tested with multiple different older phone models.

The test environment included the integrated server on the IDE (Localhost in figure 14) which connected to the IBM Forum Helsinki and went outside through the firewall and this provided the static IP which is also illustrated in figure 14 as 193.xxx.xxx.xxx. Then the service contacted the operator and the operator's firewall was adjusted to trust the static IP address so the solution could gain access to the service. The following graph demonstrates this.

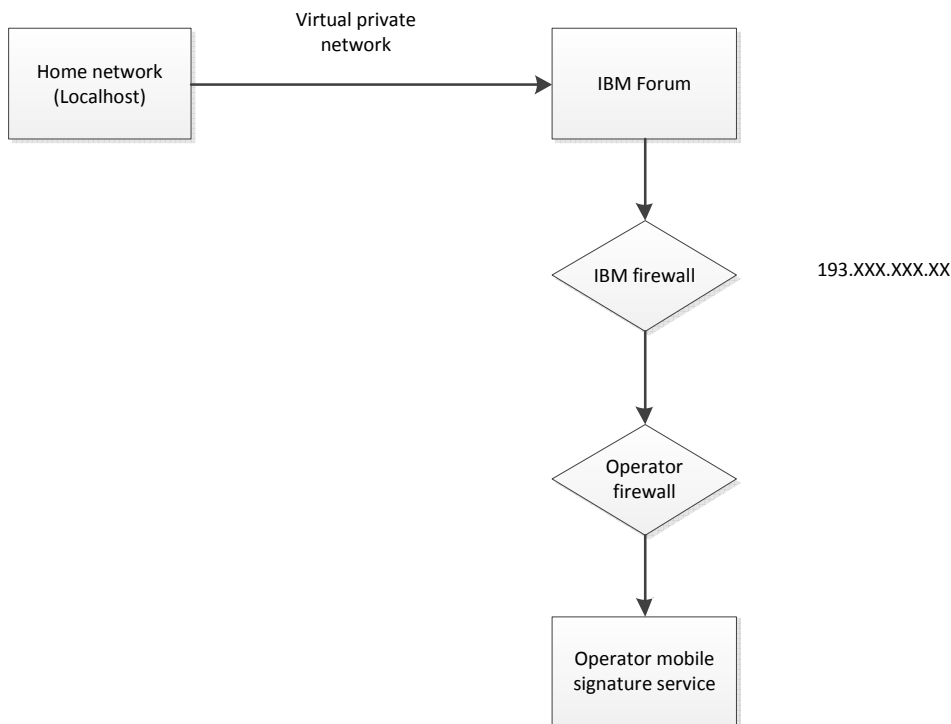
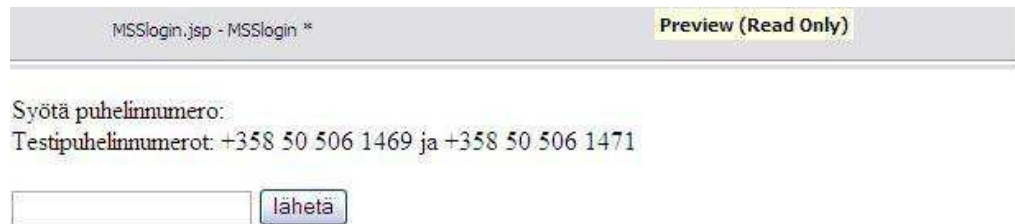


Figure 14: Test environment

Here is a demonstration of the process of the test application

- 1) Webseal intercepts the user when s/he tries to access a protected resource
- 2) TAM guides the user to the EAI application
- 3) The program starts and the user is redirected to the login screen as illustrated in figure 15.



MSSlogin.jsp - MSSlogin * Preview (Read Only)

Syötä puhelinnumero:
 Testipuhelinnumerot: +358 50 506 1469 ja +358 50 506 1471

Figure 15: MSSlogin.jsp

- 4) The user gets the authentication request to his/her phone from the operator as illustrated in figure 16.



Figure 16: Authentication request

- 5) The user replies to the request and the operator sends the response back to the software. As it can be seen in figure 17 the response contains the user's name, SATU, HETU and the original transaction ID. Both responses are from different operators and it can be seen that the responses are almost identical.


```

WebSphere Application Server v7.0 at localhost (WebSphere Application Server v7.0)
[17.4.2012 21:49:32:718 EEST] 0000002a SystemOut      O Processing..
[17.4.2012 21:49:37:703 EEST] 0000002a SystemOut      O Processing..
[17.4.2012 21:49:43:593 EEST] 0000002a SystemOut      O Signature response:
[17.4.2012 21:49:43:828 EEST] 0000002a SystemOut      O Allinen Matti 905060402
[17.4.2012 21:49:43:843 EEST] 0000002a SystemOut      O http://ms.ficom.fi/TS102204/v1.0.0/PersonID#hetu=191166-0658
[17.4.2012 21:49:43:843 EEST] 0000002a SystemOut      O A1334688538140
[17.4.2012 21:49:43:843 EEST] 0000002a SystemOut      O isValid: false
[17.4.2012 21:49:43:843 EEST] 00000026 SystemOut      O debuggausta
[17.4.2012 21:49:43:843 EEST] 00000026 SystemOut      O ServletHETU

```

```

Problems Servers Properties Quick Edit Console Search
WebSphere Application Server v7.0 at localhost (WebSphere Application Server v7.0)
[7.3.2012 13:48:30:859 EET] 00000040 SystemOut      O Processing..
[7.3.2012 13:48:35:828 EET] 00000040 SystemOut      O Processing..
[7.3.2012 13:48:40:859 EET] 00000040 SystemOut      O Processing..
[7.3.2012 13:48:45:843 EET] 00000040 SystemOut      O Processing..
[7.3.2012 13:48:51:484 EET] 00000040 SystemOut      O Signature response:
[7.3.2012 13:48:51:625 EET] 00000040 SystemOut      O Falck Tes Carita Marianne 10002050S
[7.3.2012 13:48:51:625 EET] 00000040 SystemOut      O http://ms.ficom.fi/TS102204/v1.0.0/PersonID#hetu=101092-002A
[7.3.2012 13:48:51:625 EET] 00000040 SystemOut      O A1331120884734
[7.3.2012 13:48:51:625 EET] 00000040 SystemOut      O isValid: false

```

Figure 17: signature response

- 6) After this the software sends the HETU back to TAMeb and the authentication process is done.

5 Comparing the Solutions

In the early stages of this project, there was debate about what kind of integration should be actually made. In the end TAMEb was chosen, and the interface to be used was based on the ETSI standards, but this raised the question about alternative options.

5.1 TAMEb MSS Integration vs Vetuma Integration

One of the ideas was to abandon the MSS integration and start building a Vetuma integration on TAMEb. Vetuma stands for Verkkotunnistautuminen ja –maksaminen (citizen identification and payment services). It is the service of public administration which is used for identifying citizens in web services with bank identification, certificate card or mobile certificate. The reason for this was that the plans were for Vetuma to adopt mobile identification services as well, and by creating a Vetuma integration for TAMEb those services would have been at TAMEb users' disposal [23.].

This idea never became anything more than just an idea but it could have opened some doors while closing others. First of all Vetuma is used widely by the public sector as it is used in suomi.fi website and in the municipality services. It could have provided customers from the public sector. On the other hand, this would limit the mobile identification service to the public sector services and integration for the private sector would have to be implemented anyway. Both integrations might become necessary one day but for now the integrating MSS is a better decision.

5.2 ETSI-interface vs Mobile TUPAS

The ETSI interface has been discussed earlier in this document and it is the interface in which the user can insert a phone number. The ETSI interface is located on the application provider's website, and after the user submits the phone number, the SSL connection between the application provider and the operator is initiated.

Mobile TUPAS, on the other hand, is a service offered by Elisa and mobile identification is part of the existing TUPAS framework on the website, so mobile TUPAS works as one of the banks as shown in figure 18.



Figure 18: Mobile TUPAS

Choosing the mobile identification leads the user to the operator's own specific mobile identification website where the user enters the phone number and the identification process is done directly between the user and the operator. Afterwards the operator sends the authentication information to the application provider.

On the surface the Mobile TUPAS may seem to be a better option than the ETSI interface since the user gives the phone number directly to the operator, but there are some major differences under the surface and both of the solutions are clearly made for different purposes.

One of the key differences is that Mobile TUPAS is strictly for authentication while the ETSI interface can be used for all the FiCom recommended signature profiles, which were the normal and anonymous authentication, mobile signatures and consent. Another difference is that the Mobile TUPAS always requires a browser to view the websites, and the ETSI interface could be integrated on the services that do not require a browser.

The information security technology is also different. Mobile TUPAS relies on the TUPAS security which is protected by Message Authentication Code (MAC) while the ETSI interface relies on the SSL between the servers.

The conclusion is that while the Mobile TUPAS could have some attractive features it is still limited to Elisa and that is unacceptable in this solution since TAMeb clients must be able to make contracts with their partner operators. [13.]

5.3 Alternative Solutions

There are also alternative ways to provide a mobile signature service. One way is to use SMS-OTP which stands for short message service one-time password. In this case the user provides the normal authentication information such as the username and password and in addition provides the software with a one-time password via SMS. Because the authentication requires both a password and the one-time password, it becomes a strong authentication. This could be used in any browser on a PC or mobile devices, which would practically only mean smartphones since the user would have to use the SMS application to send the password.

The SMS-OTP service is also specified by FiCom and it is rated at a lower security rating than the mobile certificate for a few primary reasons. First of all the SMS-OTP method is based on normal unencrypted 7-bit SMS technology because according to FiCom changing this would mean that the mobile devices, their application or SIM cards would require updates or changes, and this would not be acceptable. In addition SMS-OTP does not include the user's electronic identity as the mobile certificate does. [24.]

In my own opinion, there are a few facts that make the mobile certificate superior. The first is the fact that SMS-OTP relies on the normal SMS technology. This already causes more security risks than the mobile certificate since the mobile certificate never exchanges any user information, passwords or pin codes between the user and the operator but rather does the authentication on the phone and just sends a signal to the operator if everything is correct. This does not mean that the user information is easily phished when the user sends an SMS but it still poses a bigger threat than the mobile certificate. Another fact is that the user must always have the one-time passwords at hand if authentication is necessary. This means that the user could just rely on the TUPAS bank authentication and carry along the normal bank one time passwords instead of multiple different one-time password cards. With the mobile certificate the user just needs the mobile device, and remembering one personal PIN

code is enough. Some recognition could be given to SMS-OTP for not requiring nearly any processing power, but this would only matter on a PC browser since ordinary mobile phones do not have the capability for multitasking. Another positive thing is the easy to use interface since nearly everyone with a mobile phone knows how to send an SMS. Using a mobile certificate is not more difficult either though.

Another question is of course the TUPAS authentication. Why should the MSS be used since there already is a working solution that everyone knows how to use? This method has been created by the Federation of Finnish Financial Services and it is used by all major banks in Finland. It has already been a big success in the online services like the tax administration web service and the web service of Social Insurance Institution of Finland. As it has already been mentioned in the previous chapters, TUPAS authentication requires the user to have the one-time password card at hand when authenticating. The mobile signature service only requires the mobile device which most people carry with them all the time and the one code required for identification.

Another fact is that most banks require applications for their services to work on the mobile devices and have more complicated interfaces. The ETSI standards on the other hand specify the unified user experience for mobile signature services and the mobile signature service does not require a browser as the TUPAS authentication does.

Both ways provide strong authentication by requiring a pass code and something the user owns (one time password card for TUPAS users and personal mobile device for MSS users). It is debatable which is actually more secure but in my own opinion this difference is not really noticeable. All in all, MSS provides at least a good alternative for TUPAS in the field of online authentication.

6 Conclusion

The project was a success and the test program worked as intended. Even though the final testing part was not done by the time the documentation's deadline arrived the testing done so far was enough for prove that the concept of integrating MSS on TAMEb works. This already provides more business possibilities for IBM with TAMEb since the original question was whether the Mobile Signature Service could be integrated and used with TAMEb or not. Mobile signature service has the potential to be the next step in the field of online and mobile identification since it is very secure and easy to use. While the mobile market in Europe makes the mobile identification a pretty attractive business, the fact that it has not been adopted in the USA does limit the business possibilities. If it gains a foothold in the Americas, it would gain more possible users and services to use it. A few issues raised their heads, however, while studying and working with the mobile signature service. The following sections provide a summary of the conclusions based on my observation.

6.1 Operator Differences

The service is not too different between the operators but there are a few points which should be raised.

First of all, the project was completed with two operators and the actual test service is free of charge after signing a contract. The test SIM cards were also supplied for free. As the project was programmed with Java, the operators also supplied additional information about the Laverca library which proved to be a really useful addition to Laverca's own documentation. Basically the documentation was a simplified guide about how to call the necessary functions from Laverca.

The security parts were similar for both operators since the SSL connection was not the primary concern in the solution. The operators just supplied the necessary certificates for their services to work. One operator required a static IP from the source of the application to make a firewall exception and another just relied on certificates.

The only real variable between operators is that some operators might need one real mobile signature service equipped SIM card to control the mobile signature service

account which is linked to using the test service. This SIM does not have to be used actually after the initial setup so the costs for the test service still remain minimal.

6.2 MSS Advantages and Disadvantages

Currently, the mobile certificate is only available for people who have activated the mobile certificate with their respective operator with a proper identification such as passport or identity card. In addition, not every representative salesman is authorized to do the process of registering the mobile certificate and these qualified salesmen can only be found in a few chosen operator shops. This greatly reduces the number of possible users. One operator specific problem is also that TeliaSonera renamed its service from Mobiilivarmenne to Sonera ID which causes further confusion. Their website also says that Mobiilivarmenne is a product that has been removed from their portfolio. The mobile signature service itself has not appeared in the media too much either so the group of people who actually know about it is quite limited.

Currently the service is not used very widely among the private sector but it is getting more and more interested customers who want to protect their web resources with the mobile signature service, and in the public sector it has been adopted quite widely. A few bigger customers are the insurance company If, The National Board of Patents and Registration of Finland and the pension insurance company Varma. OP-pohjola has also stated that it is supporting the mobile identification.[16;25.]

One of the advantages of MSS is the fact that it is actually really safe, as it should be. The part where operator sends the request to user's mobile device is actually safer than it looks since when the user enters the SPIN code it is not actually sent anywhere as an SMS but it rather negotiates with the SIM card. This ensures that if the message is somehow intercepted, the SPIN-code will not get into wrong hands. Another important thing about the safety is of course the SSL connection. This means that the sent data is encrypted and with a brute force attack a 128-bit encryption would take 149,745 billion years to crack. [26.]

All in all my personal belief is that the mobile identification might be the next success story in the field of web identification and in the mobile world. The security part is about the same class as it has been so far with the TUPAS authentication but there are

plenty of new possibilities which the mobile identification can be used for. The easy-to-use interface and the availability on nearly any mobile phone make it seem like a good candidate to be an alternative for TUPAS or even a successor.

References

1. ElisaViestinta. Elisa mobile ID [Online]. Finland, 14 February 2012
URL: <http://www.youtube.com/watch?v=5ndOzrESRPg>. Accessed 11 April 2012.
2. FiCom ry. FiCom-implementation guideline [Online]. Finland, 14 January 2012.
URL:
http://www.mobiilivarmenne.fi/documents/MSS_FiCom_Implementation_guideline_2.1.pdf. Accessed 27 April 2012
3. IBM. IBM TAMEb infocenter [Online]. 2005.
URL: <http://publib.boulder.ibm.com/infocenter/tivihelp/v2r1/index.jsp>. Accessed 19 December 2011.
4. Barretto Analisa. What is TAM [Online]? Pune, Maharashtra India, 30 July 2009.
URL:
https://www.ibm.com/developerworks/mydeveloperworks/blogs/tivoli/entry/authentication_and_authorisation1?lang=en. Accessed 19 March 2012.
5. Ashley Paul, Norvill Trevor. How to configure the TAMEb 6.0 EAI to implement complex authentication requirements [Online]. 16 October 2006.
URL: <http://www.ibm.com/developerworks/tivoli/library/t-tamebeaicar/index.html>. Accessed 29 April 2012.
6. Wikipedia. Mobile signature [Online]. January 2008.
URL: en.wikipedia.org/wiki/Mobile_signature. Accessed 10 April 2012.
7. Mobile Marketer. Experian to offer mobile signature services in France [Online]. 19 February 2008.
URL: www.mobilemarketer.com/cms/news/banking-payments/557.html. Accessed 10 April 2012.
8. OTP Bank. Extension of the Mobile Signature service [Online]. Hungary, 2012.
URL:
https://www.otpbank.hu/portal/en/OTPdirektlogin/Security/Extension_Mobile_Signature. Accessed 10 April 2012.
9. Valimo wireless [Online]. Vantaa, Finland 2012.
URL: <http://www.valimo.com/fi>. Accessed 11 April 2012.
10. Laitila Teemu. Mobiilivarmenne saatavissa kaikilta operaattoreilta [Online]. Finland, Puhelinvertailu, 27 June 2011.
URL:
http://www.puhelinvertailu.com/uutiset.cfm/2011/06/27/mobiilivarmenne_saatavilla_kaikilta_operaattoreilta. Accessed 27 March 2012.
11. ETSI. ETSI-Standard TS 102204 [Online]. August 2003.
URL: http://docbox.etsi.org/EC_Files/EC_Files/ts_102204v010104p.pdf. Accessed 19 December 2011.

12. Finanssialan Keskusliitto. Statistics about banks' payment systems [Online]. 25 April 2012.
URL: www.fkl.fi/en/material/statistics/Statistics/Statistics_banks_payment_systems_2002-2011.ppt. Accessed 29 April 2012.
13. Stranden, Harri. 2012. IT architect, IBM Finland, Helsinki. Tapaaminen, 17.4.2012.
14. Väestörekisterikeskus. Sähköisten asiointitunnusten esiasteet luotu koko kansalle [Online]. Finland, 16 March 2003
URL: www.vaestorekisterikeskus.fi/default.aspx?site=3&docid=2717. Accessed 29 March 2012
15. Population Register Centre. Electronic identity and certificates [Online].
URL: www.vaestorekisterikeskus.fi/default.aspx?id=21&docid=33. Accessed 29 March 2012.
16. Mobiilivarmenne [Online]. Finland, 4 May 2012.
URL: <http://www.mobiilivarmenne.fi/fi/index.html>
Accessed 6 May 2012.
17. Maanoja Markus. Mobiilivarmennuksen uudet tuulet [Online]. Finland, Suomi.fi, 25 May 2011.
URL: http://www.suomi.fi/suomifi/tyohuone/yhteiset_palvelut/verkkotunnistaminen_ja_maksaminen_vetuma/Vetuma-asiakastilaisuudet/esitykset/06_Elisa_Varmenne_25052011/Elisa_Varmenne_25052011.pdf. Accessed 25 April 2012.
18. Kyrnin Jennifer. Simple object access protocol [Online]. The United States.
URL: <http://webdesign.about.com/od/soap/a/what-is-xml-soap.htm>. Accessed 6 May 2012.
19. Saura Asko. Laverca documentation [Online]. Finland, 28 June 2011.
URL: <http://sourceforge.net/projects/laverca/>. Accessed 6 May 2012.
20. Axis Development team. Apache axis introductions [Online]. 22 April 2006.
URL: axis.apache.org/axis. Accessed 6 May 2012.
21. Castor Development team. Castor introductions [Online]. 18 April 2012.
URL: www.castor.org. Accessed 6 May 2012.
22. Harju Jukka, Juslin Jukka. Tuloksellinen Java ohjelmointi. Edita publishing: Helsinki; 2006.
23. Citizen identification and payment service Vetuma [Online]. 29 June 2011
URL: www.suomi.fi/suomifi/workspace/shared_services/vetuma/index.html.
Accessed 1 June 2012
24. Arjen tietoyhteiskunnan neuvottelukunta, Sähköisen tunnistautumisen kehittämisryhmä. Mobiilitunnistautumismenetelmät [Online]. Helsinki, Finland, 13 November 2008.

URL: www.arjentietoyhteiskunta.fi/files/185/mobiilitunnistamismenetelmat.pdf
Accessed 10 April 2012.

25. Thomson Reuters ONE OP-Pohjola-ryhmä tukemaan mobiilivarmennetta [Online].
15 November 2011.
URL:
http://www.kauppalehti.fi/5/i/yritykset/lehdisto/thomsonreutersone/tiedote.jsp?oid=20111101/13213441370270&request_ahaa_info=true. Accessed 17 April 2012.
26. Symantec. Secure Sockets Layer (SSL): How It Works [Online]. 2012.
URL: https://www.symantec.com/theme.jsp?themeid=how-ssl-works&inid=vrsn_symc_ssl_How_It_Works. Accessed 27 March 2012.

