MINIMAL-FOOTPRINT MIDDLEWARE FOR THE CREATION OF QUALIFIED SIGNATURES

Martin Centner, Clemens Orthacker
Institute for Applied Information Processing and Communication (IAIK)
Graz University of Technology, Inffeldgasse 16a, Graz, Austria

Wolfgang Bauer
XiTrust Secure Technologies GmbH, Grazbachgasse 67, Graz, Austria

Keywords: Qualified electronic signatures, e-Government.

Abstract: Qualified electronic signatures are recognized as being equivalent to handwritten signatures and are supported by EU legislation. They require a secure signature creation device (SSCD) such as a smart card. Unfortunately, there are neither standard means for the integration of SSCDs with Web applications, nor are the exiting means widely deployed. Web application providers are still faced with a lack of deployment of such means and a lack of integration with standard software. This paper will present a novel approach to address these issues by a middleware that does not require users to install dedicated software for the creation of qualified electronic signatures. The middleware is deployed as a web application and splits the signature creation process into two parts: One part is performed on the server side and the other part (requiring access to functions of the secure signature creation device) is deployed and executed as a lightweight component in the user's browser on demand.

1 INTRODUCTION

Handwritten signatures play an important role in day-to-day business. They are typically used to authenticate the origin of a document and to give evidence of the intent of the signatory with regard to that document. Electronic signatures may provide an adequate equivalent in electronic business. This is also supported by the EU directive on electronic signatures (1999/93/EC, 1999), which defines a legal framework for electronic signatures and has been adopted into national legislation by the EU member states. The required infrastructure is already available in a number of the EU member states and other countries. However, electronic signatures as electronic equivalent to handwritten signatures have not yet reached a widespread use beyond specific application areas such as e-government.

In this paper we present an approach that aims at facilitating the further dissemination of Qualified Signatures. In particular we address the issues of integrating the means required for signature creation with Web applications and the deployment of such means.

The proposed minimal-footprint middleware may be deployed with a web application or as a central service and handles the details of signature creation. It enables a user to create Qualified Signatures as defined by the EU directive without requiring dedicated software to be installed by the user. To access a secure signature creation device (SSCD) as required for Qualified Signatures, a lightweight component is on demand deployed and executed in the user's browser.

The minimal-footprint middleware is a technology neutral concept. To study the concept and to provide an open-source implementation the MOCCA project[1] was initiated by the Austrian Federal Chancellery and the Graz University of Technology. Although other web technology would be conceivable, the MOCCA implementation relies on Java Applets and the PC/SC interface to stay as platform and browser independent as possible. Meanwhile MOCCA has been deployed in production for important e-government services in Austria.

In the following this paper will give a short introduction to Qualified Signatures and their requirements in section 2 and to Secure Signature Creation

[1] http://mocca.egovlabs.gv.at

Centner M., Orthacker C. and Bauer W.
MINIMAL-FOOTPRINT MIDDLEWARE FOR THE CREATION OF QUALIFIED SIGNATURES.
DOI: 10.5220/0002808300640069
In *Proceedings of the 6th International Conference on Web Information Systems and Technology (WEBIST 2010)*, page
ISBN: 978-989-674-025-2

Devices as required for Qualified Signatures in section 3. Section 4 discusses ways of integrating the creation of Qualified Signatures with applications. The minimum-footprint middleware concept and it's reference implementation MOCCA is discussed section 5.

2 QUALIFIED SIGNATURES

The EU directive on electronic signatures (1999/93/EC, 1999) defines special requirements for *advanced electronic signatures*, *secure signature creation devices* (SSCD), *qualified certificates* (QC) and certification service providers issuing qualified certificates. The requirements aim at high level of security.

Advanced electronic signatures based on QC and which are created by a SSCD—commonly referred to as *Qualified Signatures*—are required to

> . . . (a) satisfy the legal requirements of a signature in relation to data in electronic form in the same manner as a handwritten signature satisfies those requirements in relation to paper-based data; and (b) are admissible as evidence in legal proceedings. (1999/93/EC, 1999, Article 5)

Studies by the European Commission (e.g. (IDABC 6485, 2007)) show that corresponding legislation and infrastructure is available in a majority of the EU member states. However, Qualified Signatures have not yet found widespread use beyond specific application areas such as e-government.

(Roßnagel, 2009) studies the reasons for the slow adoption ratio of Qualified Signatures. As electronic signatures do not provide much benefit by themselves, their dissemination is largely dependent on availability of corresponding applications. On the other hand, the effort required for integration of electronic signatures with applications must usually be justified considering the potential usage ratio. Hence, the adoption of electronic signatures currently suffers from a chicken-and-egg problem. The problem is intensified by the fact, that currently there is a significant effort required by application providers for integrating and deploying the means required for the creation of qualified signatures.

3 SECURE SIGNATURE CREATION DEVICES

Almost all SSCDs provided for the creation of Qualified Signatures are based on smart-card technology. They are either implemented as chip-cards in credit-card size with an interface according to ISO/IEC 7810 and ISO/IEC 7816, use a contactless interface according to ISO/IEC 14443 or are directly integrated with a corresponding terminal device to be used as USB token. Therefore almost all SSCDs share at least a common low level interface defined by ISO/IEC 7816 parts 3, 4 and 8. In practice however, this just means that communication with an SSCDs is based on the exchange of Application Protocol Data Units (APDUs). To be able to access the functions of an SSCD a lot of additional information is required. Smart cards implementing ISO/IEC 7816 part 15 try to provide this required information in a standardized way. However only a few SSCDs implement ISO/IEC 7815–15 and even if they do, it remains rather impossible to interface SSCDs in a complete generic way. Therefore, either applications have to know how to access a specific SSCD or need layer of abstraction and another component doing so.

PC/SC[2] has become the de facto standard for integration of smart cards and smart-card readers into mainstream operating systems, with other technologies such as CT-API[3] losing importance. PC/SC allows for communication with smart-cards on the basis of APDUs. On the contrary, a number of competing solutions exist for the abstraction and integration of electronic signatures and other cryptographic functions based on smart cards into operating systems and applications. To name just a few, there are operating system dependent solutions like Microsoft's CSP/CNG[4], Keychain Services[5] in Apple OS X and more operating system independent solutions such as PKCS#11[6]. All these solutions have in common, that they require a module implementing the specifics for each particular SSCD they are going to support. Of course, these modules look quite different for any of the solutions. Thus it is not surprising, that none of them is currently able to support all or even most of the available SSCDs. In fact, there is also a number of SSCDs for which not any such module is available. Additionally, the installation and update of specific modules can also be a challenging task for end users if this is not performed by the operating system or applications automatically.

[2] http://www.pcscworkgroup.com/

[3] CardTerminal Application Programming Interface http://www.tuvit.de/downloads/Tuev-IT/CTAPI11EN.pdf

[4] Crypto Service Provider / Crypto Next Generation http://msdn2.microsoft.com/en-us/library/aa380256.aspx

[5] http://developer.apple.com/mac/library/documentation/Security/Conceptual/Security_Overview/Security_Services/Security_Services.html

[6] http://www.rsa.com/rsalabs/node.asp?id=2133

The EU directive on electronic signatures requires that SSCDs must ensure that “the signature-creation-data used for signature generation can be reliably protected by the legitimate signatory against the use of others” (1999/93/EC, 1999, ANNEX III). Most SSCDs implement this by requiring the user to provide a secret personal identification number (PIN). Any solution accessing the SSCD must therefore also be able to provide the user with a possibility to enter the PIN to authorize the signature. This might be done by an appropriate user dialog or by activating a PIN-pad on the smart-card terminal if available.

4 APPLICATION INTEGRATION

SSCDs implement only basic cryptographic algorithms. Signatures on electronic documents usually require a signature format such as CMS[7] and XMLDSig[8] (or their corresponding formats for Advanced Electronic Signatures, CAdES and XAdES[9]). If applications directly interface SSCDs or access their functions by interfaces discussed in section 3, they also need to implement the processing of the required signature formats. Abstraction of this functionality can be provided by libraries which have to be integrated with the application or can be achieved by the use of a middleware that provides signature creation services.

To enable users to create Qualified Signatures in Web applications, these applications need a way to employ the user’s SSCD. There are however no standard means provided by Web browsers for the creation of electronic signatures. Therefore, middleware solutions have been developed that need to be installed on the user’s machine and allow to be triggered through a Web browser. The middleware then takes over the task of creation a Qualified Signature with the user’s SSCD and returns it back the application. Such solutions are for instance used by the Austrian Citizen Card (Hollosi and Karlinger, 2004),(Rössler, 2008) and the German e-Card-API-Framework (BSI TR-03112, 2008). Of course such middleware may also be accessed by local applications on the user’s machine.

The EU directive on electronic signatures mandates that “Secure signature-creation devices must not alter the data to be signed or prevent such data from being presented to the signatory prior to the signature process” (1999/93/EC, 1999, ANNEX III). Therefore middleware for the creation of Qualified Signatures usually also provides the possibility to view the to-be signed data.

The traditional middleware approach represents a convenient way to integrate Qualified Signatures with applications and especially with Web applications. However, conventional middleware software needs to be installed on the user’s machine as a prerequisite. The minimum-footprint middleware approach eliminates this requirement by providing the required components for accessing the user’s SSCD on demand.

5 MINIMAL-FOOTPRINT MIDDLEWARE

The general concept of the approach is to split the signature creation process into two parts, use a small component executed in the user’s browser to perform the part that needs access the functions of the SSCD and deploy this component on demand. Everything that may be performed securely on the server is kept out of this component, to allow for a minimal-footprint on the user’s machine. From the application’s point of view however, there is just a conventional middleware taking over the task of creating an electronic signature.

The Austrian Federal Chancellery and the Graz University of Technology have initiated the project “MOCCA” in 2008 with the aim to provide an open source implementation of a minimal-footprint middleware.

5.1 SSCD Access

In the context of the MOCCA project several technologies for implementation of the browser component and accessing the SSCD have been assessed. The goal was to support the mainstream operating systems (Windows, Mac OS X and Linux) and at least the three most used Web browsers (Internet Explorer, Firefox and Safari). Firefox and Internet Explorer would principally allow access to SSCD functions using scripting technologies such as Java-Script and VB-Script. However, this requires appropriate modules for integration of the SSCD with the operating system (Internet Explorer: CSP) and the browser (Firefox: PKCS#11), respectively. The appropriate modules have to be installed and configured by the user. As an alternative browser plug-in technologies

[7] Cryptographic Message Syntax (CMS) – IETF RFC 3852 http://tools.ietf.org/html/rfc3852

[8] XML Signature Syntax and Processing – W3C Recommendation http://www.w3.org/TR/xmldsig-core/

[9] CMS Advanced Electronic Signatures – ETSI TS 101 733 and XML Advanced Electronic Signatures – ETSI TS 101 903 http://www.etsi.org/WebSite/Technologies/ElectronicSignature.aspx

such as Flash, Silverlight and Java Applets where evaluated. A possibility for accessing an SSCD using Flash or Silverlight in the available versions was not found.

Java Applets provide operating system and browser independent access to smart cards via the Java Smart Card I/O API using the PC/SC interface. The targeted operating systems provide out of the box support for PC/SC and for many of the smart-card readers on the market. Therefore, a user only needs to plug a supported smart-card reader and is not required to install additional software.

Statistics show, that more than 50% of internet users are using a browser with a Java-Plugin in the relevant version[10]. For those not having a Java-Plugin installed, Java provides an easy installation procedure that could also be triggered by the middleware directly. As Java is general technology not only required for the creation of electronic signatures and already available on more than the half of the users' machines, the requirement not needing to install dedicated software is considered to be met. Therefore, Java Applets where chosen for the implementation of the browser component of MOCCA.

5.2 Architecture

The architecture of MOCCA is designed to be as technology neutral and open as possible. This should allow for future adaptations if required. It is organized into four different layers as shown in figure 1.
For integration with applications it was chosen to implement the Security-Layer interface (SL) and a transport binding of HTTP(S) as defined by the Austrian Citizen Card specification. SL is an open concept comparable with OASIS-DSS[11] and not restricted to Austrian citizen cards. Additional interfaces (such as OASIS-DSS) could easily be implemented by extending or adapting the two uppermost layers.

The Security-Layer interface allows for the creation of CMS and XAdES signatures. Currently, MOCCA only implements the creation of XAdES signatures. The corresponding SL request consists of elements specifying the signature key to be used, the data to be signed and the XML document the created signature should finally be embedded in. There is great flexibility in specifying how the middleware should retrieve and transform the data for signing and whether the data should be embedded into the signature or referenced externally.

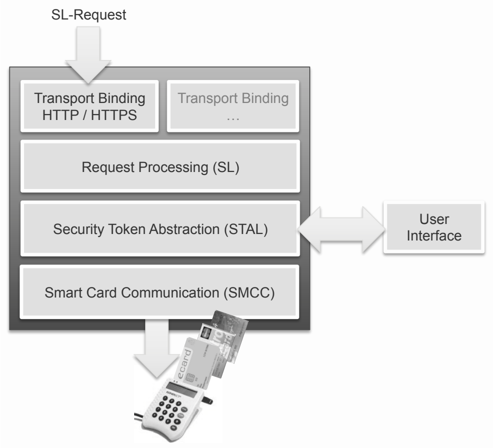
SL-Request
Transport Binding HTTP / HTTPS
Transport Binding …
Request Processing (SL)
Security Token Abstraction (STAL)
User Interface
Smart Card Communication (SMCC)

Figure 1: Layers of MOCCA.

Transport Binding Layer. A transport binding of HTTP/HTTPS is provided to enable an application to access the middleware through the browser by using simple Web forms carrying an SL request.

Request Processing Layer (SL). performs all the steps required for the creation of an XAdES signature except for the calculation of the signature value.

Security Token Abstraction Layer (STAL). consists of a client and a server part. The server part offers a Web service. The client part is included in the Java Applet. The client connects back to the Web service of the server when the Applet has been started. The Web service is then used to send commands from the server to the client part of the STAL. [12]

Smart Card Communication Layer (SMCC). is also included in the Java Applet and called by the STAL to handle communication with SSCDs and smart-card readers via PC/SC.
All XML processing required for the creation of a XAdES signature is performed in the request processing layer. This allows for very simple STAL commands. Currently, the STAL supports only two basic commands. One for retrieving certificates and other information stored on a SSCD and one for the creation of a signature using the SSCD. Additionally, the STAL may call back to the Request Processing layer for retrieving the data referenced by the to-be signed

[10] Source: http://riastats.com/

[11] OASIS – Digital Signature Service http://www.oasis-open.org/committees/dss/

[12] A standard SOAP based Web service with swapped roles of server and client has been chosen in favor of a reversed SOAP (PAOS) binding. This allows for a standard Java client implementation and for non-Java clients (e.g. Java Script, Flash, etc.) to connect to the server. This may become relevant if other technologies would allow SSCD access in the future and should be integrated.

signature. This is used to enable the user to retrieve and view the data to-be signed as explained below.

5.3 Security Model

A Java Applet may be signed to ensure it's integrity and to allow the user to verify that it is from a reliable source. If a Java Applet also needs access to local resources it must be signed. The first time a signed Applet is loaded by the Java-Plugin the user needs to give consent for executing it and may choose to always trust the issuer of the Applet.

It was chosen to encapsulate all security relevant parts of the process of creating a Qualified Signature within the signed Java Applet. The Applet may be signed by a trusted third party and different service instances may provide the same signed Applet.[13] For the purpose of creating a Qualified Signature, users therefore only need to trust the issuer of the signed Applet. This is considered to be equivalent with the requirement to trust the issuer of other conventionally installed software used for the creation of Qualified Signatures (e.g. locally installed middleware-software, PKCS#11 modules, etc.).

The entire XML processing is performed outside the Java Applet, to keep it as small and simple as possible. The Applet only receives the *SignedInfo* part of the XAdES signature, which contains the references and hash values of the to-be signed data. However, the Applet allows the user to retrieve and view the to-be signed data. When it retrieves this data it calculates the hash value and compares it with the hash value of the corresponding reference in the *SignedInfo*. That way, the Applet can ensure the data retrieved for viewing is actually the data going to be signed. When the user decides not view the to-be signed data, there is also no need to compare the hash values.

The security relevant communication with smart cards and smart-card readers is handled entirely within the Applet. The Applet is relying on PC/SC communication only. Therefore, it needs to know how to employ the functions of a particular SSCD. To keep the footprint of the Applet to a minimum, it does not rely on additional modules such as PKCS#11 that would need to be deployed with the Applet. The SSCD abstraction is therefore implemented directly by the SMCC layer of the Applet.

5.4 State of Implementation and Findings

So far the official MOCCA release supports, the Austrian health-insurance cards and other Austrian Citizen Cards. A number of SCCDs from other EU countries has already been integrated in the context of the pan european project on electronic identities “STORK”[14]. This includes the Belgian “Belbic” and smart-cards form Italy, Portugal and Estonia. Support for these and other smart-cards is going to be integrated with the official release.

Along with the creation of Qualified Signatures, MOCCA also supports the requests required for authentication using the Austrian Citizen Card (Rössler and Leitold, 2005) (Rössler, 2008). MOCCA has already been deployed in production for authentication and the creation of Qualified Signatures for a number of e-government services in Austria. This includes, the Austrian online tax declaration service *FinanzOnline*[15], which is the most used e-government service in Austria. Additionally it has been successfully used in the past student union election in Austria, which was the very first lawful election in Austria using remote e-voting facilities.

6 CONCLUSIONS

The presented minimal-footprint middleware represents a concept to leverage the use of Qualified Signatures in Web applications. It enables users to create Qualified Signatures without requiring them to install dedicated software as a prerequisite. At the same time it encapsulates the details of the signature creation process and hides them from the application. This allows for an easy integration with applications.

The authors have proven the concept by an open-source implementation “MOCCA”, which is now an important building block of Austrian e-government services.

The principle concept is technology neutral—for the implementation of MOCCA a Java Applets and the PC/SC interface have been chosen. These two technologies are available across browsers and operating system platforms.

There are other solutions for the creation of electronic signatures available on the market, which also use Java Applet technology. However, to the best of our knowledge, none of these solutions is implemented as middleware. The middleware concept,

[13] The Applet provided with the official MOCCA distribution is actually signed by the Secure Information Technology Center – Austria (A-SIT), which is the Austrian confirmation body for electronic signature solutions.

[14] http://www.eid-stork.eu/

[15] https://finanzonline.bmf.gv.at

would yet allow to switch technologies without affecting application integration. If technologies should emerge in the future that would allow for an even better handling of the SSCD integration, it would suffice to adapt just the middleware. The minimal-footprint middleware concept therefore has all the advantages of a conventional middleware without the need to be installed by the user.

REFERENCES

1999/93/EC (1999). *Directive 1999/93/EC of the European Parliament and of the Council of 13 December 1999 on a Community framework for electronic signatures*. European Parliament and Council.

BSI TR-03112 (2008). *BSI - Technische Richtlinie: eCard-API-Framework (BSI TR-03112).* Bundesamt fr Sicherheit in der Informationstechnik.

Hollosi, A. and Karlinger, G. (2004). The Austrian Citizen Card. AG Bürgerkarte. http://www.buergerkarte.at/konzept/securitylayer/spezifikation/20040514/introduction/Introduction.en.html.

IDABC 6485 (2007). *Preliminary Study on Mutual Recognition of eSignatures for eGovernment applications, Report*. European Commission / European eGovernement services (IDABC).

Leitold, H., Hollosi, A., and Posch, R. (2002). Security architecture of the austrian citizen card concept. In *Computer Security Applications Conference, 2002. Proceedings. 18th Annual*, pages 391–400.

Rössler, T. (2008). Giving an interoperable e-ID solution: Using foreign e-IDs in Austrian e-Government. *Computer Law & Security Report*, 24(5):447 – 453.

Rössler, T. and Leitold, H. (2005). Identifikationsmodell der österreichischen Bürgerkarte. In *Proceedings of the D-A-CH Security Conference 2005*, University of Technology Darmstadt, Germany.

Roßnagel, H. (2006). On diffusion and confusion – why electronic signatures have failed. *Lecture Notes in Computer Science – Trust and Privacy in Digital Business*, pages 71–80.

Roßnagel, H. (2009). *Mobile qualifizierte elektronische Signaturen*. Datenschutz und Datensicherheit.