A secure, affordable infrastructure for electronic signature applications

Overview of a server based, user-friendly approach.

Lothar Fritsch

Eighteen months after the EU directive on electronic signatures passed legislation, certification service providers (CSPs) around Europe are handing out personal smartcards containing qualified certificates to be used as electronic signature devices. Naturally, the focus of eCommerce application development is the integration of electronic signatures in trade, finance and government affairs. While the first applications that entered the market were small, specialized pieces of software usable only by enthusiasts, now there are concepts for interoperable, ergonomic and secure infrastructures for electronic signatures enabling a high throughput. This article presents a security infrastructure designed for economic, ergonomic and secure signing of web based documents.

Lothar Fritsch
Product Manager for IT-Security and Internet Payment at fun communications GmbH in Karlsruhe, responsible for electronic contracting frameworks and signature applications.
http://www.fun.de
E-Mail: lothar.fritsch@fun.de

Many business cases on the web require non-repudiation and authentication. Business processes in finance, insurance, medicine or eGovernment usually are based on interactive forms. This is an ideal environment for electronic signatures. To create an infrastructure suiting these business needs as well as meeting the user’s privacy needs, an application-independent architecture is required. The immediate consequences of this approach are:

- the application must handle a multitude of business cases and related forms (and thus, various document formats)
- the application will deal with smartcards, card readers and certificates from many CSPs
- the application will be used by all kinds of users (who are, in particular, more or less experienced and conscious of IT security)
- the application should be available on many computing platforms to ensure broad acceptance on the market.

Furthermore, economic aspects for vendors and operators are a crucial factor. In particular, the maintenance of clients for multiple platforms causes a high cost of ownership. Thus, the first obvious design decision is to create a server-based, thin-client infrastructure using Java technology. The reasons for this are the mobility of Java applets on the one hand, and the portability of Java server applications on the other hand. Thus, the infrastructure will be used for the sake of user acceptance the standard web browsers with Java virtual machines on the client, while the signature server will control much of the processes.

The second crucial design decision is the neutrality concerning the use of the signed documents. A vendor of a complex signature product is dependent on a large number of instances sold on the market. Thus, specialization concerning the particular application of signature technology is unaffordable to a software house. Following this, a strict abstraction of the business process from the signature mechanism is required, resulting in the need to delegate the workflow management to an external systems. Compatibility to standard applications and standard software\(^1\) ensures the possibility to connect the signature process delivered by the signature infrastructure to specialized application products. On the other hand, independence from particular application software is required to ensure flexibility for the operator of a signature application – the flexibility to exchange the application software for the business processes.

As a result, the basic infrastructure is an electronic signature application server designed as a „service module” for business applications. From the business application’s view, the signature server is a transaction server for contract transactions. This server provides the abstraction of the whole signature process to the business process. Diagram 1 shows the basic infrastructure.

Security Infrastructure

The handling of various forms – and in particular their content – on-line requires attention. Particularly in the finance business or in eGovernment, personal data is filled into forms. Privacy laws as well as

\(^1\) E.g., ERP or CRM systems and standard office software.
Signature application framework

Diagram 1: Framework and roles

the protection of business secrets both lead to the need for a security infrastructure fulfilling these needs in a reliable way. Strong focus should be put on the unobservability of the signed document content by unauthorized parties — through encryption through and access control.

Another important element of the security infrastructure is the prevention of document falsification — including both the "secure viewer" as well as the reliable document transport topics.

This section presents a design for such an infrastructure.

Unobservability and Access Control

While there is SSL for the transmission of encrypted and somehow authenticated web pages, a reliable security for a secure signature framework is more complicated. We suggest to use a smartcard containing the qualified electronic signature certificate as well as a certificate for authentication. The advantage is the employment of the smartcard's physical security for Authentication purposes.

For the sake of transmission security, we advise the employment of a certified secure channel instead of SSL. SSL is a complex protocol without any certification of reliability or strength of the functions [3]. Furthermore, cryptographic keys in SSL can authenticate a computer, but not a person.

Diagram 2 shows a Java based security infrastructure used to authenticate the person signing or accessing a restricted web space.

Instead of the simple http data transfer, a security applet is in control of the http protocol. At the application server, a security server controls the access of the client. Two different processes are in charge: a localhost proxy applet for secure transfer, and the authentication applet using its own communication channel to signal the presence of a smartcard to the security server using a secure challenge-response protocol. The security server keeps a session context connecting the two processes to each other. As soon as the smartcard is removed from the card reader, the session context turns invalid, thus further access to the restricted web space will be denied. Instead of SSL, a formally certified security protocol, e.g. the Transport/S 2.0, (E4/high) [6] can be used to reliably transmit content and authentication data.

Document Safety

In this section, a closer look at the handling of the documents to be signed is taken. At first, we observe the path of a document, e.g. a form, from the business application to the signer and back to the relying party.

Please note that in this description, the transparent presence of the security server described above is assumed to protect the interaction between the user and the signature server.

Interactions

Signing a Document

The process starts in step (1) in diagram 3. The user interacts with the business application, e.g. by filling in a form. Submitting the form to the business server, the server can validate the form contents. In step (2.), the business application instructs the signature server to gain a signature form the user. The document is handed over to the signature server as well as a session ID. At the same time, the user's web browser is handed over an URL generated by the signature server, causing the browser to connect to the signature server using its own session ID (step (3.)). The 4th step delivers the signature applet and the document to be signed to the client web browser. Using the applet's document viewer, the user inspects the document delivered by the signature server, creates the signature using his qualified certificate and sends the resulting signed document back to the signature server (step (5.)). Optionally, the signed document can be stored to a local archive.

Now, the signature server performs various actions to ensure the validity of the signed document:

- a comparison of the document sent to the client with the document received;

2 For code integrity reasons, the applet must be a signed applet. Optionally, the user may install a manual copy of the applet from a CD to his web-browser.
A secure, affordable infrastructure for electronic signature applications

Diagram 3: the complete signature infrastructure (the underlying http tunneling security infrastructure in diagram 2 is omitted).

- a verification of the electronic signature data, using the public key from the user’s qualified certificate;
- access to the on-line directory of the CSP that issued the user’s qualified certificate to ensure the certificate’s legal validity. This may also uncover attributes restricting the validity of the certificate;
- optionally connect to a trusted time stamping service to add a reliable date and time to the document.

Step (6.) finally delivers the result to the business application: either the signed document with status information coded into a standard data format\(^1\), or an error code describing the problem that occurred. Problems range from a simple time out during the user interaction with the signature server up to certificate invalidity or manipulated documents.

**Document Integrity**

So far, three parties are involved in handling the document to be signed and the signed result. How can be ensured the user is presented with his original document? Still, we assume data transport and physical integrity to be based upon the security server infrastructure. At first, the user fills in the document to be signed, returning it to the business application. The business application may format the document contents before presenting the document and user ID to the signature server\(^4\).

1. Such a format is e.g. PKCS#7.
2. Several commercially available servers for the handling of on-line forms reformat the submitted information into a new document.

The user then is presented with the resulting document to be signed. Here, the user is able to verify the document’s contents. Following the creation of the signature, the signature server compares the document to the document that was originally sent to the user. These two processes create a workflow where both the user and the signature server double-check the document’s contents.

Naturally, the next topic to be reviewed is the document format security and the secure viewer for the users.

**Document Format Security**

There has been much discussion about trustworthy software and related security problems at the user interface of signature applications. So far, all certified signing components in Germany rely on simple text formats for secure document viewing. While this approach is simple enough to handle the certification of client Software, it does rarely meet the requirements of business processes. Documents to be signed for industry applications may include:

- the company’s corporate identity design including graphics;
- complex data formats like CAD files;
- forms using various styles, formatting and graphical elements.

Thus, a broad acceptance of electronically signed documents on the market will be linked to the availability of standard document formats for signed documents. The failure to handle such documents will cause the failure of commercial success of signature application systems.

How can secure document viewing be handled? The approach of software certification is far too complex for application software like word processors or CAD systems. Certification for established document creation software, e.g. the systems established on personal computers, seems like a hopeless effort after a decade of software engineering has been done on these systems\(^3\). On the other hand, quick market acceptance of signature application must be reached by offering the established document formats to the user.

How can this dilemma be resolved? My suggestion is the usage of open document formats so the signature server can verify the document formats. The idea is illustrated using Adobe’s PDF document format and Acrobat Reader\(^\text{TM}\) viewer.

As PDF is an open format with publicly available specification, the signature server is enabled to accept „data format verification plug-ins”\(^2\). In the case of PDF, the plug-in is a parser for PDF documents implementing the open specification. The plug-in ensures a proper document following the specification, and in particular will reject any document containing dynamic content like input fields or JavaScript. After verification using the plug-in, the signature server transmits a ‘clean’ document to the client.

The client will use Acrobat Viewer as the document viewer. Naturally, Acrobat Viewer is not a secure viewer. A way out of this dilemma is a vendor’s self-declaration. The EU directive for electronic signatures requires a self-declaration from CSPs that will be held responsible for damages inflicted by malfunction. A similar declaration for viewer software, requiring vendors to declare standard conformity and the absence of unwanted functions in their software will ensure viewing software quality.

Unlike the viewer, a format filter software is simple enough to be formally certified, resulting in a considerably short time to-market at competitive economic conditions.

Other security problems of the viewer, like the lack of security of operating systems and PC user interfaces cannot be tackled by signature application manufacturers. The treatment of this class of security threats is far out of the scope of this article. A possible resolution is the usage of

---

\(^{3}\) see also [5],[6] and [7] for an impression of the complexity of software certification.
secure signature environments like certified security wallets [8], as there is little hope of fast availability of accepted, secure PC operating systems.

Meanwhile, some level of security can only be reached by employing rather strict measures of IT security: firewalls, eMail and Web content verification, software integrity checks and personal firewalls on the PC used as a signing workstation along with protected disk space.

**User Ergonomy**

One of the principal design decisions of the signature infrastructure was the ease of use by the signing party. From fun communication's experience in building online banking systems, we observed that

- users resent extra installation efforts for new technologies;
- the total cost of ownership for corporations increase dramatically with every extra step of installation a user is required to do;
- users value mobility, resulting in a faster acceptance of web based services on the market (compared to client-based services);
- users dislike large instruction manuals and training efforts.
- users have the habit to neglect security for comfort.

Some study of user ergonomy of a signature framework these effects has been done in [1].

The main principles of a user-friendly electronic signature framework should be:

- the absence of installation efforts and software maintenance work;
- interoperability of several PKIs;
- transparent use of secure signature creation devices from many CSPs;
- transparent use of various document formats and viewers;
- user-friendly updates of the signing software and viewers;
- automatic and comfortable update of CA certificates, revocation lists and other information concerning the operation of CSPs.

Java technology on the signature server allows to avoid installation efforts as well as update efforts on the user's side. These installations are done by the server's vendor or its operators. Naturally, access to the signature server may only be granted to trustworthy staff following a strict security policy.

The installation of new PKIs and CSP certificates is performed at the server as well. This may need the installation of new applet interfaces to deal with smartcard readers and particular smartcard interfaces. An optimal solution provides a plug-in interface for the new PKI „drivers”. In analogy to PC hardware components, at some time in the near future such „drivers” could even be provided by the CSP itself. Another promising approach is interoperability standards for PKIs such as the ISIS-MTT standard support by CSPs in Germany. ISIS-MTT specifies an infrastructure for interoperable certificate structures, directories and other elements of PKI administration. Details can be found in [2].

Revocation lists concerning CSP certificates are handled on the server, too. Here, the regular installation of revocation lists can be done more or less automatically.

An open issue is the installation of new document formats. On the server, using the filter plug-in together with an update of the applets, new document formats can easily be integrated. But this is only sufficient in the cases where the applet is able to present the document for viewing.

In other cases, such as handling complex documents, e.g. from DTP or CAD software, the user still will be required to go through an additional installation process. Although I’m convinced this procedure is acceptable to the users. As mentioned in the introduction, we deal with various levels of user know-how and motivation. It should suffice for „home users” for most of their on-line business cases to rely upon HTML forms, signed text and widely available formats like PDF. Those users who require complex document creation and viewing systems are either well-experienced or professional users who are motivated and own the knowledge for software installation, e.g. by using their system administrator.

**Conclusion**

This article proposes an infrastructure that was developed considering the evolution of electronic banking from host-based terminal applications to client software to web based applications. Examining this evolution, the infrastructure described above is the state of the art in networked business systems. On the web, user acceptance is crucial, which we reach by providing easy of use as well as a security structure that protects privacy of the document content against eavesdroppers.

Furthermore, economic constraints are considered by the abstraction from a particular application so the flexible use of the infrastructure is possible.

Interoperability is the key for wide acceptance on the market – with the „driver” concept and standards like ISIS-MTT, a wide range of PKIs can be support.

Therefore, we expect large signature applications to occur on the European market soon. We believe this infrastructure design to be an important submission to the further development of electronic signature applications.

**Bibliography**


