USENIX Association

Proceedings of the 5th Smart Card Research and Advanced Application Conference

San Jose, California, USA
November 21–22, 2002

Smart Cards in Interaction: Towards Trustworthy Digital Signatures

Roger Kilian-Kehr Joachim Posegga

SAP AG Corporate Research, CEC Karlsruhe
Vincenz-Priessnitz-Str. 1, D-76131 Karlsruhe, Germany
{*roger.kilian-kehr, joachim.posegga*}*@sap.com*

Abstract

We present approaches to raise the security level in the process of electronic signature creation by shifting as many tasks as possible involved in digitally signing data into a tamper-resistant and trustworthy smart card. We describe the fundamental technical principles our approach is based on, illustrate resulting design options, and compare the security of our approach with traditional electronic signature scenarios.

Keywords: *electronic signatures, smart cards.*

1 Introduction

The cryptographic underpinnings of electronic signatures such as mathematical one-way functions or public key cryptography are well understood, and practically secure algorithms and key lengths are widely established. From this perspective, electronically signing documents is a straightforward undertaking.

The actual procedure for digitally signing a document or a transaction, however, is a complex scenario in practice which involves numerous issues beyond cryptography: Since a person who wants to create a digital signature will usually not carry out the relevant computation by herself, she needs to delegate this to some application running on a platform (device) that can perform such computations. The security level of the overall signature creation process therefore depends on the security of several other, non-cryptographic factors, e.g. the environment where the document/data presentation takes place, the security of the communication channel to a user, or the security properties of the environment where the cryptographic computations are carried out.

Consider a scenario where a user signs a document displayed in a Web browser on a PC; at best, this involves a smart card, where the signing key is stored, and the cryptographic algorithm to encrypt the document hash (and other relevant data) is executed within the card. An attacker who wants to trick the user into signing a fake document would likely not attack the smart card, but the environment within it is used (i.e.: the OS, the driver of the smart card reader, the signing application, the Web browser, etc.).

The "added value" of a smart card is such a scenario is, largely, that it makes it hard to compromise the cryptographic key, but the card contributes little to the actual trustworthiness of an individual digital signature: The card is used as a tamperproof device that executes a fixed computational function, i.e., it reads a data block, encrypts it, and it returns the result. The card itself, however, does not interact directly with the user (card holder), but through a mediator like a PC or a mobile phone. But these devices are usually a lot less secure that a typical smart card.

This problem is, in theory, easy to solve: Raise the security level and require a closed, trustworthy system for applying electronic signatures. Unfortunately, this solution is extremely hard to roll out in practice, both because it is expensive and since dedicated hardware, which would be required, simply does not fit into today's computing world.

We propose to take another direction, and build upon execution platforms that are provided by today's smart cards, in particular SIMs and USIMs used for GSM and UMTS. Such cards offer functionality beyond the "hardwired", secure token that smart cards are mostly figured: Besides holding a secret key and performing cryptographic algorithms, GSM SIMs and UMTS USIMs include application platforms (e.g. [7, 15]), that allow programs that run inside these smart cards to use services of its host. A mobile phone hosting such a SIM provides I/O and networking capabilities to the SIM over standardized

protocols [6, 3, 4, 5, 1, 2]. As a result, applications running inside a SIM can actively initiate and control user interaction, communicate over the network, etc.

Our paper discusses the various options for enhancing the security of the process of signing a document by involving a secure execution platform in such a smart card; essentially we investigate the following question underlying such an approach:

> *How much in terms of security can be obtained, if as much functionality as possible is shifted from untrusted components into a trustworthy platform available in a tamper-resistant device?*

Overall, our research provides means for increasing the trustworthiness of digital signature by imposing less assumptions on the integrity of a card terminal that classic approaches do.

Paper Outline

The main research contribution of our paper is given within Sect. 2. After introducing some notational conventions with standard digital signatures in Sect. 2.1, we investigate basic, on-card hash computation in Sect. 2.2. Section 2.3 extends this by involving a trusted third party. A third approach integrating the identity of the document's originator into the signature protocol is presented in Sect. 2.4. Although all the approaches are vulnerable to so-called "conspiracy attacks" they represent significant improvements in the overall security of an electronic signature creation process.

Based on the results of the previous approaches Sect. 2.5 proposes to digitally sign user interactions triggered by scripts that run inside smart cards to enable the comfortable, application-driven creation of electronic signatures on mobile devices.

Section 3 compares our work to related approaches, and we finally wrap-up our work in Sect. 4.

2 Smarter Signing with Smart Cards

This section explores several options for implementing the process of digitally signing documents by taking advantage of secure application platforms in smart cards: We discuss the security benefits of moving more and more of the required computation into the secure environment of a card.

As the starting point, consider the "traditional" procedure, where smart cards are used as crypto tokens holding a secret key and providing an implementation of cryptographic algorithms.

2.1 Basic Electronic Signature Protocol

The most important roles in scenarios for electronic signature creation are the signer S owning a public key pair (S_S, P_S), the document to be signed D, the signature creation application A, a document viewer V interacting with the signer, a smart card C, and the originator O of the document D. The basic protocol is as follows:

$$\begin{array}{lrl} (1) & O \to A: & \{D\} \\ (2) & A \to V,S: & \{D\} \\ (3) & S,V \to A: & accept/reject \\ (4) & A \to C: & \{h(D)\} \\ (5) & C \to A \to O: & \{sig_{S_S}\big(h(D)\big)\} \end{array}$$

Here, (1) denotes the document transfer from the originator to the signature creation application, (2) the document presentation, (3) the signer's interaction/choice, (4) the hash computation, and (5) the signing process.

The above procedure can be improved w.r.t. security when moving some of these individual steps partially into the secure environment of a smart card. First we consider on-card hash computation.

2.2 Electronic Signatures with On-Card Hash Computation

The computation of the hash function is certainly a possible target for an attacker who wants to manipulate a signing procedure; but performing the hash computation inside a trusted device such as a smart card itself is not a panacea: it is important how the document presentation *and* hash computation is done in the overall signature protocol. Consider for example the following case:

$$\begin{array}{lrl} (1) & A \to C: & \{D\} \\ (2) & C \to A: & \{sig_{S_S}\big(h(D)\big)\} \end{array}$$

In this case, (1) denotes the document transfer to the smart card and (2) the document signing process. From a security point of view an intruder I who is in control of A can easily exchange document D with another document D' which is subsequently sent to the card, hashed, and finally signed. Hence, compared with the basic protocol, no additional benefit can be gained from moving a hash computation into a card in a straightforward way.

On-Card Hash Computation Protocol

Assuming a scenario in which the signature creation application A is located in a security module C, and the viewer in a (less trustworthy) terminal a possible protocol is as follows:

$$\begin{array}{lrl} (1) & O \to A: & \{D\} \\ (2) & A \to V,S: & \{D\} \\ (3) & S,V \to A: & accept/reject \\ (4) & C,A \to O: & \{sig_{S_S}\big(h(D)\big)\} \end{array}$$

Here, (1) denotes the document transfer to the application being hosted by the smart card, (2) the document presentation, (3) the user's choice, and (4) the hash and signature computation in the card.

Assuming end-to-end secure communication between O and A/C, an intruder is not able to control the hash computation anymore. Only the document presentation and the user's *accept/response* could be manipulated, although the intruder controlling V cannot gain anything from such manipulation, except by mounting the following attack.

A Conspiracy Attack on On-Card Hash Computation

- The intruder I and the originator O cooperate.
- O sends the document D', i.e., the document which the attackers want to be signed by S.
- Upon invocation of V, I presents a fake document D, which S might accept for signing.
- In the card, D' is signed and sent back to O.

Hence, an attack is still possible, if the intruder subverting V and the originator O of the document directly cooperate. Although this attack is of general importance, practically, it means that it is not sufficient anymore to attack the user's terminal only, but also to manage to actively send a faked document which the user subsequently signs.

As a consequence, shifting the hash computation in the above manner to a tamper-resistant device seems to give a substantial improvement in the overall security of the signature creation process.

2.3 Electronic Signatures Assisted by a Trusted Third Party

On-card hash computation is often not feasible, e.g. due to the limited bandwith one can use when communicating with a smart card. The process of computing hashes can, however, also be delegated to a trusted third party T as the following protocol outlines. A URL url_D is used to denote some resource where D can be fetched from. The trusted third party T then computes D's hash on behalf of A and signs it. A just forwards the URL to the document viewer V and the further protocol steps are the same as in the on-card hash computation protocol (cf. Sect. 2.2).

$$
\begin{array}{lrl}
(1) & O \rightarrow A: & \{url_D\} \\
(2) & A \rightarrow T: & \{url_D\} \\
(3) & T \rightarrow A: & \{sig_T(h(D))\} \\
(4) & A \rightarrow V, S: & \{url_D\} \\
(5) & S, V \rightarrow A: & accept/reject \\
(6) & A \rightarrow C: & \{sig_T(h(D))\} \\
(7) & C \rightarrow A \rightarrow O: & \{sig_{S_S}(h(D))\}
\end{array}
$$

In this protocol, (1) denotes the transmission of the URL under which the document to be signed is located to the application, (2) passing the URL to the TTP, (3) TTP fetches document and computes the hash, (4) represents the document presentation to the user, (5) the user's choice, (6) pass-through of the TTP's signature to the card and verification the the signature, and (7) the final signature computation by the smart card.

Similar to the on-card hash computation protocol, it is vulnerable to a conspiracy attack as described above.

2.4 Electronic Signatures with Recipient Addressing

Looking at the traditional signature creation protocol it becomes obvious that authenticity of a document sender is not of particular concern. In electronic business processes, however, signatures are often used to provide the technical means for contracts between two parties. Although the identities of the contract partners are usually somehow denoted in the document D, this is by no means cryptographically protected.

To improve the signature process further, we include the cryptographic identity of the document originator into the signature process. In particular we propose the following protocol which is based on the on-card hash computation protocol (cf. Sect. 2.2) and the public key pair (S_O, P_O) of the originator O denoted by id_O:

$$
\begin{array}{lrl}
(1) & O \rightarrow A: & \{D, sig_{S_O}(D)\} \\
(2) & A \rightarrow V, S: & \{D, id_O\} \\
(3) & S, V \rightarrow A: & accept/reject \\
(4) & C, A \rightarrow O: & \{sig_{S_S}(h(D), sig_{S_O}(D))\}
\end{array}
$$

Here, (1) denotes the document and signature transmission, (2) the presentation of the document and the identity of the originator, (3) the user's choice, and (4) the final hash and signature computation.

This protocol now achieves that an electronic signature is created over both – the cryptographic hash of the document and the identity of the recipient or originator of the signature.

To assess the advantages of this approach consider that in a traditional signature attack scenario an intruder could "hijack" the signing process of an arbitrary document D_O with its intended recipient O to

infiltrate another document D' to be signed. The intruder I could then claim that the user has signed this document which is likely of advantage to the intruder. In the above protocol, however, the intruder I is not able to obtain a signature $sig_{S_S}\big(h(D'), sig_{S_I}(D')\big)$ since the signature $sig_{S_I}(D')$ cannot be generated. At best $sig_{S_S}\big(h(D'), sig_{S_O}(D')\big)$ could be obtained, but leading to a contradiction between the information available in D' denoting I as the recipient and the envelope signature sig_{S_O}. Therefore, we argue that linking the document and the recipient in the signature gives advantages to standard electronic signature creation.

Basically, the same conspiracy attack presented in the on-card hash computation in Section 2.2 can be mounted in the recipient addressing scheme. Again, if originator O and intruder I cooperate, the user is not able to distinguish that signature creation occurs with a document that she does not intend to sign.

2.5 Electronic Signatures on Interactions

We have so far considered electronic signatures on standard clients, e.g. desktop PCs. One of the most problematic issues with electronic signatures on mobile devices is the fact that such signatures are computed over complex documents. In particular this means that according to current signature laws, e.g. those in Germany, the document must be presented to the user who then either accepts or rejects the subsequent signature creation. Hence, a document to be signed must be presented as a whole in a suitably rendered fashion, which is often difficult on small, mobile devices. The problem of encoding and subsequently displaying a document in a reproducible and standardized way has been extensively discussed by Scheibelhofer [13]. In his approach he uses XML style sheets defining mappings to a possibly certified rendering engine.

To tackle this presentation problem, we consider not only the presentation of a document but also the way the document is created. We argue that a document is often the result of some kind of interaction between a service provider, e.g. who offers goods, and a client who selects goods to buy. Finally, after all selections are made, a document containing the complete list of goods is presented and signed accordingly.

If such an interaction "document" is encoded as an executable script, the execution of the script is *deterministic* as long as all *non-deterministic* input which is received from "outside" the script such as user input, random number generator, persistent variables, etc. is recorded. A "document" over which the signature is computed is then comprised of

(a) the executed script,

(b) the persistent state used during the computation,

(c) all user input,

(d) all messages received from other communication channels,

(e) the current time and progress of execution,

(f) some platform characteristics such as version numbers, serial numbers, etc.

The signature can be easily verified by executing the script in a simulated environment using the recorded and signed input values. Thus, a signed document in this sense is not intended to be human-readable, but rather meant to record and log the interaction that happened between a service provider and a user.

A Smart Card Platform for Mobile Code

More concretely, we propose to use a secure platform for the execution of (remote) code in a smart card which functions as follows:

- The smart card implements an interpreter for mobile code written in a domain-specific language optimally supporting the intended application domain.
- A client such as a service provider sends messages containing so-called *scripts* written in the domain-specific language the card-resident interpreter understands.
- The card's runtime platform executes the script, handles user interaction, and sends back the responses to the client.
- The platform implements key management facilities in order to provide end-to-end security between the client and the smart card.

Such a platform must be *secure* in the sense that neither the mobile code nor the user is able to harm the platform's integrity. Furthermore, the platform gives certain guarantees to both – code and user – that the scripts are executed as intended and no information leakage or secret storage manipulation can occur by malicious code or an external attacker.

Thus, the platform acts as a *trusted computing base* running in a tamper-resistant device protecting the user from the code and vice versa.

```
1  script {
2
3     provider "bidbiz.com";
4     name     "bidbiz auction client";
5     id       "20011223/24357";
6     options  signed-interaction;
7
8     implementation {
9        playtone;
10       push("News from bidbiz.com:\nBid in auction #3576 (Antique watch): EUR 63.");
11       display;
12
13       push(mark);
14       push("Place new bid?");
15       push("New bid...");
16       push("Cancel");
17       select;              <-- User selects option: (int,'1')
18
19       push( 2 );
20       eq?;
21       if (true) goto end;
22
23 enter:
24       push("Enter new bid (>EUR 63):");
25       input;               <-- User inputs new bid amount: (string,'70')
26       dup();
27       push(63);
28       le?;
29       if (true) goto end:
30       playtone;
31       push("Please enter a bid greater than EUR 63.");
32       display;
33       goto enter;
34
35 end:  sign-interaction;
36       response;
37       exit;
38    }
39 }
```

Figure 1. Mobile auction client with interaction signatures

Example: Mobile Auctions

For illustration purposes we provide an example illustrating our approach in the domain of mobile auctions (Fig. 1). This example is based on the one presented in [8], however, it has been extended to support the creation of signatures on user interaction.

The given example illustrates the use of the stack-based domain-specific language we use to write our scripts without going into full detail. An in-depth description of the language can be found in [10].

A script starts with header information about the name of the script and its provider (lines 3–5). Line 6 denotes that the script's execution should be implicitly signed by the interpreter. The `implementation` part (line 7) contains the actual program.

Lines 9–12 demonstrate how to display an initial message about the latest news of the online auction. Lines 14–18 show how the arguments for a user selection (primitive `select`) are pushed onto the stack marked by the initial marker set in line 14. After the selection has been performed the arguments including the marker are removed from the stack and the number of the selected item is available on the stack.

Lines 20–22 check, whether the subscriber selected item no. 2 (i.e. "Cancel") in which case a jump to the label '`end`' is performed. Otherwise an input dialog is opened in lines 25–26 and the input from the subscriber is returned on the topmost stack position and duplicated in line 27. Then the entered amount is checked in lines 28–30, whether its is greater than 64. Otherwise a text is displayed in lines 31–33 and execution resumes to the input dialogue (label '`enter`').

Finally, in line 36, the whole recorded execution of the

script is signed and a signature object containing all the relevant information about the script's execution including the signature is pushed onto the stack. The signature object is then sent back to the originator in line 37 and execution terminates.

During execution the runtime environment collects the non-deterministic input from the various sources into a log $L = \{i_1, \ldots, i_n\}$ of inputs i_j. In the above example execution thus yields

$$L = \{(\texttt{int}, \texttt{'1'}), (\texttt{string}, \texttt{'70'})\},$$

i.e., for each input we record the type information and the data. The overall interactive log of an execution of script P with the identifier id_P is computed and returned together with additional platform information R to the original sender S as follows:

$$C \rightarrow S: \quad \left\{id_P, L, R, sig_{S_C}\big(hash(id_P, P, L, R)\big)\right\}.$$

The receiver must be able to verify the authenticity of the signature by simulating the execution of the script according to the log L. Based on this simulation, the interaction of the script and the user can be replayed and the user's choices and inputs can be examined to take appropriate action.

The execution of the script should occur in a transactional context, i.e. if for some reason the execution is terminated, no signature is created.

Summary

Using signatures on runtime execution audits combined with recording user interactions as a means to implement non-repudiation is, to the best of our knowledge, a novel approach. We consider this approach particularly useful for our application domain for the following reasons:

- Due to the lack of user input and output facilities, performing all possible executions within the trust domain of the smart card is from a security point of view desirable.
- All interaction which leaves the trust boundary of the smart card is reduced to the bare minimum, i.e. to user interactions only.
- The approach is very flexible, since it offers scripts a full control over the way signatures are built, how encryption is performed, and how interaction takes place. As such it is able to offer applications means to implement security policies as needed.

Thus, our approach allows to take full advantage of the smart card as an open platform for running security-critical applications in the tamper-resistant context of the physical device. More precisely, it represents one instance of the on-card hash computation approach as presented in Sect. 2.2. Furthermore, it can be easily extended to also support the third-party assisted approach in Sect. 2.3 and the recipient addressing approach in Sect. 2.4 assuming available key management facilities as described in [10].

3 Related Work

The main contribution of our research is in increasing the trustworthiness of digital signatures by building as much as possible on the security properties of execution platforms in smart cards.

Alternatively, one can try to enhance the trustworthiness of devices; [11, 12] discuss portable end-user devices (POBs) and security modules and define a number of requirements to be made for such devices. They observe that trustworthy POBs do not exist and conclude that therefore the development of secure applications should concentrate on protocols and procedures. A related approach is, e.g. described in [9] that comprises two different devices, a PDA and a smart card, that together implement a security-sensitive application: the smart card does not perform its task without the PDA and the PDA cannot perform the task without the help of the smart card.

One of the key ideas of this paper is documenting user interaction involved with digital signatures; a suitable, lightweight scripting language suitable for on-the-fly download to smart cards has been proposed in [8, 10].

The actual runtime execution monitoring using an execution log has been investigated by Vigna as a means to protect the execution of mobile agents in hostile environments [14]. The sender of a mobile agent can use this signed execution log to verify whether the agent has been tampered with while executing on a remote agent platform.

4 Conclusion

Starting from the observation that the process of electronic signature creation is still vulnerable in many practical settings, we have proposed three protocol variants that aim at shifting functionality from untrusted components into a smart card.

The first option considers on-card hash computation combined with end-to-end secure transfer of the document to be signed from the originator to the smart card. Another approach uses a trusted third party to perform resource-intensive computation of the document's hash outside the card. The third approach is characterized by the integration of identity of the document's originator

into the protocol eliminating further attacks. However, so-called "conspiracy attacks" in which an intruder and an originator cooperate are still, yet less easily, mountable.

Based on the new protocols a novel approach for the creation of electronic signatures based on a runtime execution platform for smart cards has been presented. This approach is able to include the different protocol options presented and is especially suited for use in mobile settings characterized by the limited device capabilities in terms of user input and output. We have illustrated our approach with an example in the domain of mobile auctions – an application that is ideally suited to be run on mobile phones.

Generally, we suggest that GSM SIMs and UMTS USIMs might be ideal candidates for hosting such a smart card platform. Our results demonstrate that improvements in the electronic signature creation process are feasible if the environment the creation takes place is suitably taken into consideration.

References

[1] 3rd Generation Partnership Project. *3GPP TS 31.112 V5.0.0 (2001-09) Technical Specification 3rd Generation Partnership Project; Technical Specification Group Terminals; USIM Application Toolkit (USAT) Interpreter Architecture Description (Release 5)*, Sept. 2001. Available at *http://www.3gpp.org*.

[2] 3rd Generation Partnership Project. *3GPP TS 31.113 V5.0.0 (2001-09) Technical Specification 3rd Generation Partnership Project; Technical Specification Group Terminals; USAT Interpreter Byte Codes (Release 5)*, Sept. 2001. Available at *http://www.3gpp.org*.

[3] European Telecommunication Standardization Institution (ETSI). *Digital cellular telecommunications system (Phase 2+); Security Mechanisms for the SIM application toolkit; Stage 2 (GSM 03.48 version 8.1.0 Release 99)*, Nov. 1999.

[4] European Telecommunication Standardization Institution (ETSI), Sophia Antipolis, France. *Digital cellular telecommunications system (Phase 2+, Release 98): Subscriber Identity Module Application Programming Interface (SIM API); Service description; Stage 2 (GSM 02.19 version 7.0.0 Release 1998)*, 2000.

[5] European Telecommunication Standardization Institution (ETSI), Sophia Antipolis, France. *Digital cellular telecommunications system (Phase 2+): Subscriber Identity Module Application Programming Interface (SIM API); SIM API for Java Card; Stage 2 (GSM 03.19 version 7.1.0, Release 1998)*, 2000.

[6] European Telecommunications Standard Institute. *Digital cellular telecommunications system (Phase 2+); Specification of the SIM Application Toolkit for the Subscriber Identity Module – Mobile Equipment (SIM–ME) interface (GSM 11.14)*, 1998.

[7] Java Card Technology. Specifications are available at *http://java.sun.com/products/javacard/*.

[8] R. Kehr and H. Mieves. SIMspeak: Towards an open and secure platform for GSM SIMs. In I. Attali and T. Jensen, editors, *Proceedings of International Conference on Research in Smart Cards: Smart Card Programming and Security, E-smart 2001, Cannes, France,*, volume 2140 of *Lecture Notes in Computer Science*. Springer-Verlag, September, 19–21 2001.

[9] R. Kehr, J. Posegga, and H. Vogt. PCA: Jini-based Personal Card Assistant. In R. Baumgart, editor, *Proceedings of Secure Networking – CQRE [Secure]'99, Düsseldorf, Germany*, volume 1740 of *Lecture Notes in Computer Science*, pages 64–75. Springer-Verlag, November 30 – December 2, 1999.

[10] R. Kilian-Kehr. *Mobile Security with Smartcards*. PhD thesis, Darmstadt University of Technology, May 2002.

[11] A. Pfitzmann, B. Pfitzmann, M. Schunter, and M. Waidner. Vertrauenswürdiger Entwurf portabler Benutzerendgeräte und Sicherheitsmodule. In *Proceedings of Verläßliche Informationssysteme VIS'95*, pages 329–350. Vieweg, 1995.

[12] A. Pfitzmann, B. Pfitzmann, M. Schunter, and M. Waidner. Mobile user devices and security modules: Design for trustworthiness. Technical Report RZ 2784 (#89262), IBM Research Division, Zurich, May 1996.

[13] K. Scheibelhofer. What you see is what you sign. In *Proceedings of IFIP conference on Communications and Multimedia Security (CMS '2001), Darmstadt, Germany*, May 21–22, 2001.

[14] G. Vigna. Cryptographic traces for mobile agents. In G. Vigna, editor, *Mobile Agents and Security*, volume 1419 of *Lecture Notes in Computer Science*. Springer-Verlag, June 1998.

[15] Windows for Smartcards. *www.microsoft.com/smartcard/*.