Trust Relations in a Digital Signature System Based on a Smart Card

[Published in *Proc. of Cartes 2000*, pp. 429–449, Paris, France, October 24-26, 2000.]

Jean-Luc Giraud and Ludovic Rousseau

Gemplus, Cryptography & Security Group
B.P. 100, 13881 Gémenos Cedex, France
{jean-luc.giraud, ludovic.rousseau}@gemplus.com

Abstract. In this paper we present different ways for an attacker to tamper with a digital signature scheme. The paper focuses on smartcard-based signature schemes because they offer the best ratio of cost over security. The risks of compromise or attack for each part of the system used to produce the signature are assessed and possible solutions for the problems illustrated are suggested. In fact, the smartcard is often the most secure link in the chain of trust involved in digital signature systems.

Keywords. Trust, security, smart card, electronic signature.

1 Introduction

Online commerce on the Internet is expected to generate considerable profit in the near future. People are expected to order goods and services from their computer instead of going to the retail shop. This new type of commerce where the buyer and the vendor are not physically present at the moment of the transaction creates new ways of cheating the system. In order to reach the expected volume of transactions, Electronic commerce systems will be developed to replace the current paradigm of a credit card number over an SSL link.

Electronic commerce transactions are basically an agreement between a customer and a merchant to exchange goods and money. This amounts to the signing of a "contract" between the two parties. With the advances of modern cryptology, strong solutions exist to provide secure signature systems. The entire security of the system will then rely on the secrecy of a value (called a key) and keeping this key secret is thought to be the main challenge of this technology. The use of smartcards seems to be an elegant and easy way to solve this problem, but, as we will see, the secrecy of the key is far from being the only issue in a digital signature system.

The first section highlights the different properties that an ideal digital signature system should have. We then discuss how they are implemented in real

systems and where/how an attacker can try to subvert the system. Finally, we offer some ideas on how to solve these problems.

2 Ideal Digital Signature Systems

The development of business to business or customer to business electronic commerce will require a legal framework to specify liability. In many countries, laws are currently being passed in Europe or in the United States stating that electronic signatures can be used as valid proof for an agreement or a transaction.

For these reasons, it is important to consider the available technical means to generate signatures in order to choose the most reliable ones.

2.1 Properties of a signature

Usually, a signature is expected to be [Sch96]:

- authentic: a valid signature implies that the signer deliberately signed the associated message,
- unforgeable: only the signer can give a valid signature for the associated message,
- not reusable: the signature of a document can not be used on another document,
- impossible to repudiate: the signer can not deny having signed a document that has a valid signature.

In addition, one would expect that the signed document is unalterable: the signature should not be valid if the message is modified.

2.2 Building blocks for a digital signature system

We will now see what we have to put in our digital signature system to fulfill the requirements listed in 2.1.

Providing authenticity In order to be able to prove that the signer agreed to sign the document, the digital signature system must be such that it can only be activated by the genuine signer. This basically means that the system has to authenticate the user prior to any signature computation. There are classically four categories of authentication techniques, which rely on:

- Something the signer knows (e.g. PIN codes, password,...)
- Something the signer is (e.g. biometric characteristics,...)
- Something the signer knows how to do (e.g. the way of writing,...)
- Something the signer owns (e.g. a hardware token, a smart card,...)

The first solution is the simplest and most widely used one. The second one is probably more secure but currently relies on rather expensive hardware and is therefore not adapted to a large-scale system. With the generalization of touch-pads on PDAs and laptops, the third solution might soon be applicable on a wide scale. Hardware tokens will be considered as part of the signature systems instead of a means to authenticate the signer.

The issue of authenticity brings about the problem of the identity of the signer: in a digital world, there is no obvious way to be sure who signed a document. Even in the really powerful paradigm of public key cryptography, there is no built-in way of making sure that a public key you relieved along with the document belongs to your old friend Alice or the untrustworthy Eve. A solution to this issue is the use of trusted third party and more precisely a Certification Authority (CA). The CA will prove in a secure way to the receiver of the signature the link between the element of proof the signer gives with his signature (e.g. his public key) and the signer's identity.

Unforgeable signatures With modern cryptography, providing unforgeable signatures in an ideal (e.g. mathematical) signature system is not a problem anymore. This issue has been studied for a long time and a great breakthrough happened in 1976 with the discovery of public key cryptography [DH76]. Public key cryptography is probably the most elegant solution available. After the work of Diffie and Hellman, many public key cryptosystems have been devised and the most popular is RSA which was invented in 1978 [RSA78]. With this algorithm, provided that the signer's private key remains secret, there is no way to forge his signature.

Not reusable signatures A signature computed on one document should not be of any use to produce a valid signature of another document. With RSA, if an attacker is given C_1, the signature of document M_1 and C_2, the signature of document M_2, he can easily generate the valid signature for document $M_1 \times M_2$ by just computing $C_1 \times C_2$. To protect the system from such an attack, padding schemes have to be applied and the padded message will be signed. In real implementations, a hash function will also be involved to reduce the amount of information to sign: instead of signing the whole message, the system will only sign the output of the hash function which is a "fingerprint" of the original text.

Non repudiation With a public key cryptosystem repudiation is not possible in an ideal signature system provided that:

- the signer authentication mechanism can not be by-passed,
- there is no way to subvert the CA,
- the signer's private key can not be compromised.

If the first point is true, than a signer won't be able to claim that he did not authenticate for this signature computation. If the second one is true, than

nobody can generate fake certificates and pretend to be some else. If the signer's private key can not be compromised, no attacker would have been able to compute a valid signature. There is then no way for signature to be computed without the agreement of the signer.

Non-modifiable message Using a modern hash function allows to protect the signature system from post-signature modification of the message.

3 Real Digital Signature Systems

Real systems attempting to match the properties of the ideal schemes described in 2 usually include the following entities:

- a signer (or card-holder)
- a verifier (retailer)
- a smartcard (or another type of secure token)
- a smartcard reader
- a Personal Computer (PC) which in turn includes:
  - an Operating System (OS)
    - serial or USB port driver
    - smartcard driver
    - cryptographic library
  - one or more applications (Mail User Agent, browser,...)
- a Public Key Infrastructure (PKI)
  - one or more Certification Authority

For a typical Internet electronic commerce transaction, the signer is the customer at his home, the smartcard and its reader are his own and so is the PC. He manages the OS on his own. The verifier is the retailer and the entity managing the PKI could be a bank.

For purchases in a store, the signer has his own card but the reader and the PC belongs to the verifier (and might actually be a payment terminal). The PKI manager would also be a bank.

The following section lists which part of the system are involved to get the properties given in 2 and what the trust relationships are. Potential attacks are given in the subsequent section.

3.1 Providing authenticity

As stated in 2.2, this property will require two functionalities:

- Signer authentication
- Identity verification

The signer will be authenticated with some secret information: a PIN code, some biometric information or his way of writing his signature. In classical systems, terminals get the signer's authenticator (i.e. secret information) from a central server and compare it with the one that has just been entered. To prevent any eavesdropping, the central server might cipher the authenticator with a key specific to each terminal. One major drawback is that this kind of system does not scale very well: in very large applications (ATM,...) any entity willing to authenticate the signer has to fetch the value from the central site which requires time-consuming communication. Smartcards offer a much more elegant solution: the signer holds a tamper-proof device capable of authenticating him before allowing any signature computation.

The signer's secret is going to be processed by at least the reader and also by the keyboard and computer most of the time.

The signer's computer will be used to display the message to sign. The signer will agree to sign according to what he reads on the screen.

The trust relationships are as follows:

- the signer trusts the smartcard to be tamper resistant enough to safely store his authenticator. He also trusts the reader and sometimes the computer to not memorize his secret,
- the signer trusts that the smartcard does not allow any signature computation without prior authentication,
- the signer trusts that the document he reads on his computer and agrees to sign is really what is given to the smartcard.

Once the signer is authenticated, the cryptographic mechanisms involved in the signature process can be activated. In a standard system, they will include a public-key algorithm, a padding scheme and a hash function. In order to link the public key in use with the identity of the signer, a Certification Authority (CA) is required. The CA signs a certificate with his private key, binding the signer's identity and public key. The signer's certificate is stored on the smartcard. The public key of the CA is supposed to be known by all parties in the system.

Here the trust assumption is that the CA's private key can not be used by anybody but the CA itself.

3.2 Unforgeable signatures

As mentioned in 2.2, the main issue here is to keep the private key of the signer secret. Private key are usually too long for the signer to remember, so a first solution is to keep it ciphered on the hard-drive/floppy disk of the computer. The deciphering key should only be known by the signer. At any rate, the private key of the signer is deciphered at some point on the computer and the level of security of standard OSs is far from being sufficient to prevent attempts to read it unciphered in the computer memory. A much safer solution is to use a smartcard to store the private key *and* compute the signature. The system is more secure and flexible since the signer always has his card with him.

In the smartcard-based system, the signer trusts that the smartcard can securely store his private key, otherwise, he would not use it.

3.3 Not reusable signatures

Even in a real system, using a strong hash function (SHA-1,...) and padding schemes is sufficient to block any attempt to reuse a signature.

3.4 Non repudiation

The parts of the system involved in non repudiation are similar to those used in 3.1, but the trust relationships are dual: the signature verifier trusts that the system and its sub-parts only allow the legitimate signer to sign a document. This way, the verifier will be able to prove that the signer agreed upon signing in case of repudiation.

- the signature verifier trusts that the smartcard is tamper-resistant enough so that it can't reveal the signer's authenticator,
- the verifier trusts the system (smartcard excluded) to not memorize the signer's authenticator,
- the verifier trusts that the smartcard does not allow any signature computation without prior authentication of the signer,
- the verifier trusts that the document the signer read on his computer and agreed to sign is really what was given to the smartcard.

3.5 Non-modifiable message

This property is obtained by using a strong hash function. It should be implemented in the smartcard but is currently most of the time computed by the computer.

4 Attacks on Real Digital Signature Systems

This section shows how the fundamental properties of signatures can be violated in a real digital signature system. The way each trust assumption can be broken is analyzed.

All entities in the system have different interests and might use all possible means to cheat it:

- signers will revocate their cards and then sign orders right after, hoping that the delay in the transaction process will enable them to validate the transaction before denying having ever done it
- verifiers (retailers) will try to get a document signed without the signer being aware of it or even better, they will try to get the secret key of their customer to place unwanted orders
- outsiders might attack the CA's private key in order to create valid certificates and use them in the system
- CAs might want to deny having received a revocation order if they have interest in it.
- ...

4.1 Attacks on authenticity

Revealing the signer's authenticator There are mainly two ways of getting the value of the signer's authenticator:

- breaking the smartcard
- capturing it on its path from the keyboard to the card

Attacks on the card The signer's authenticator is usually a static value stored in the card. Possible attacks usually are:

- brute force attack:
  challenging the card with all possible values until the right one. This would be particularly efficient on a PIN code. Unfortunately, smartcards use a ratification counter to allow a maximum number of wrong presentations. This attacks is therefore inefficient.
- power analysis [Koc98], [KJJ99]:
  these attacks are not efficient on static verifications. Further more, they usually need the averaging of many power consumption waveforms and the ratification mechanism will prevent the attacker from getting enough.
- invasive attacks [AK96], [KK99]:
  an attacker with access to the proper hardware and a good knowledge of VLSI design would eventually retrieve the value of the authenticator. The time required to implement this attack would probably more than sufficient for the signer to realize that his card has been stolen. He would therefore revocate his certificate before the attacker could use the information he got.

Eavesdropping A much easier way to get the value of the signer's authenticator is to eavesdrop on the communication channel between the input device used by the signer (e.g. the keyboard) and his smartcard. For instance, for a PIN code verification, the signer types his PIN on the keyboard, the PC's Operating System gets the key codes and transmits them to the smartcard reader which finally sends it to the card.

Tampering with the keyboard or the smartcard reader is a type of attack that can happen at a retail shop where a malicious shop owner can modify his terminal to capture the PIN code of all his customers. An accomplice can then steal the card and use it. This scenario is rather dangerous for the shop owner because the system usually has a back-end consolidation system which can usually detect that a lot of stolen cards have been used in this particular shop before they were robbed.

For transactions over the Internet, where the signer is in a trusted environment (his owns his keyboard and smartcard reader), the PC's OS is clearly the weakest point in the chain. It usually is poorly designed and when it is not, security mechanisms are disabled by default to provide maximum user friendliness. Many viruses and Trojan horses can be uploaded to the PC through e-mail, web-browsers,... The GemPC420 [Gem99] smartcard reader can thwart this attack: it is connected between the keyboard and the PC and when a PIN code

verification is requested, the keyboard input is captured by the reader. The OS never has access to the PIN code value. A LED allows the signer to see when the reader is in capture mode.

Signing without authentication In order to minimize interaction with the signer, the authenticator is only requested by the card once after it is powered-on. From then on, the signature primitive can be used as often as necessary. When the card is used on an untrusted terminal, after the signer has been authenticated, the terminal can request the signature of many documents without the signer's consent. The notion of untrusted terminal extends to the signer's own computer as seen in 4.1.

A prudent protocol would require the signer's authentication prior to each signature computation request.

Document modification The document the signer agreed to sign can be modified before it is sent to the card. So the terminal or the PC has to be trusted. Tampering with the terminal might requires expertise and it is possible to build tamper-evident terminals.

In a PC, the Operating System is responsible for communications between the application and the smarcard reader. Modifying the OS's functionalities is a very powerful way to subvert the signature system. One way of doing is to modify the software packages that a signer might download from the internet to be able to modify the behavior of the OS. Solutions like using Gnu Privacy Guard [Gnu99] in the Debian GNU/Linux system to sign packages exist [Col99].

Another type of attack is to install a Trojan horse on a PC which can be really straightforward [CER98a], [CER98b].

A great enhancement of the security of PC's OS seems to be necessary to achieve strong levels of security in digital signatures. Global solutions can be an integrity checker at the kernel level as in [Igl99] or enhanced security mechanisms in[Mic99].

Attacking the PKI The final attack on authenticity is when the signer pretends to be somebody he is not. In order to do this, he has to forge a false certificate with another person's identity and a public-key/private-key pair he knows. To perform this, he needs to know the private key of the Certification Authority which seems highly difficult in a well designed PKI.

4.2 Forging signatures

In a good public key cryptosystem (RSA with proper key size,...), you need the signer's private key in order to be able to forge his signature. In our system, it is stored in the smartcard. As in 4.1, the secret material can be retrieved by

- power analysis [Koc98], [KJJ99]:

this class of attack, and DPA in particular, are really efficient. Nevertheless, smartcard manufacturer have improved their cards to thwart these attacks or at least make them much more difficult. An important drawback of power analysis is that the attacker has to have access to the physical token to conduct the attack. This would probably be noticed by the signer.
- invasive attacks [AK96], [KK99]:
  an expert with access to the expensive hardware required for this attack would probably be able to get the value of the secret key. Once again, the signer would probably realize that his card was stolen before the attacker can use the secret material he retrieved.

4.3 Signature repudiation

Non-repudiation is probably the most difficult property to have in a real signature system. Applicable attacks are those listed in 4.1. But the defending side is in a much less favorable position because it is in the signer's interest to reveal his secret material. A possible solution is to warn the signer that any signature given before a repudiation of his card will be considered as valid. This way of working is already applied in many countries implementing smartcard-based payment schemes. The signer then has a clear disadvantage: the party in charge of revocation could pretend to have never got the repudiation order or at least to have received it later than the signer claims. Systems where the verifier and the Certification Authority are one entity could be particularly unfair towards the signer. Therefore, in a fair system, the CA should always be a Trusted Third Party.

4.4 Post signature message modification

The issue of potential message modification after a signature has been extensively studied in the field of cryptography. Very good solutions are available with modern hash functions and padding schemes. For instance, a system using SHA-1 and PCKS#1-V2 with RSA would not allow to modify a signed message without invalidating the signature.

5 Conclusion

Current transactions on the Internet use a credit card number over an SSL link. The fraud rate of this system is much higher than for normal credit card transactions and honest customers might be distressed to see their credit card numbers used by malicious third parties. New solutions will have to be developed to protect the customer, the retailer and the bank. Smartcard-based systems seem to offer an excellent ratio of security over price and security over flexibility. As illustrated in this paper, the trust relationships are complicated and natural assumptions might not hold. For instance, the signer's PC can't be trusted, even

though it is physically protected: logical attacks might prove much more profitable to a malicious person to get an arbitrary document signed. There has been a lot of work on the tamper resistance of smartcard and lots of publicity on recently discovered attacks, but the smartcard is far from being the weakest link in a signature system at the moment. Furthermore, chip and smartcard manufacturers continuously improve the security level of their products. Electronic commerce is probably going to be one of the most challenging security development ever since all transactions are conducted without the signer actually seeing the verifier. Even if solutions to most of these issues exist already, there is still some work to do to build very strong systems. Enhancement of the performances in security of standard Operating Systems seems to be the blocking point for the moment, but the new generation already has promising features.

References

[AK96] Ross Anderson and Markus Kuhn. Tamper Resistance — a Cautionary Note. In *Second USENIX Workshop on Electronic Commerce Proceedings*, pages 1–11. USENIX Press, 1996.

[CER98a] CERT. <http://www.cert.org/vul_notes/VN-98.07.backorifice.html>, 1998.

[CER98b] CERT. <http://www.cert.org/vul_notes/VN-98.06.ms_jscript.html>, 1998.

[Col99] Ben Collins. Experimental dpkg available. <http://www.debian.org/Lists-Archives/debian-devel-9910/msg02053.html>, October 1999.

[DH76] Whitfield Diffie and Martin E. Hellman. New Directions in Cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, November 1976.

[Gem99] Gemplus. GemPC420. See URL <http://www.gemplus.com/products/hardware/gcr420.htm>, 1999.

[Gnu99] GnuPG team. The GNU Privacy Guard. <http://www.gnupg.org/>, 1999.

[Igl99] Pietro Iglio. TrustedBox: a Kernel-Level Integrity Checker. In *Fifteenth Annual Computer Security Applications Conference*, pages 189–198, Phoenix, Arizona, USA, December 1999.

[KJJ99] Paul Kocher, Joshua Jaff, and Benjamin Jun. Differential Power Analysis: Leaking Secrets. In *Advances in Cryptology – CRYPTO'99 Proceedings*. Springer-Verlag, 1999.

[KK99] Oliver Kömmerling and Markus G. Kuhn. Design principles for tamper-resistant smart card processors. In *USENIX Workshop on Smart Card Technology*, pages 9–20. USENIX Press, May 1999.

[Koc98] Paul Kocher. Differential Power Analysis. Technical report, CRI, 1998. Available at <http://www.cryptography.com/dpa/>.

[Mic99] Microsoft. Windows 2000 reliability and availability improvements. See URL <http://www.microsoft.com/windows2000/library/howitworks/management/relavail.asp>, 1999.

[RSA78] Ron L. Rivest, Adi Shamir, and Leonard Adleman. A Method for Obtaining Digital Signatures and Public Key Cryptosystems. *Communications of the ACM*, 21(2):120–126, February 1978.

[Sch96] Bruce Schneier. *Applied Cryptography*. John Wiley & Sons, Inc., 2nd edition, 1996.