Mitigating the Untrusted Terminal Problem Using Conditional Signatures

István Zsolt BERTA Levente BUTTYÁN István VAJDA
Laboratory of Cryptography and Systems Security,
Department of Telecommunications, Budapest University of Technology and Economics
{istvan.berta, levente.buttyan, istvan.vajda}@crysys.hit.bme.hu

Abstract

We study the problem of how a user at an untrusted terminal can generate digital signatures with the help of a smart card. This problem may arise in many practical applications; an example would be a user generating an electronic check at a merchant's terminal in a shop. The danger is that the terminal can obtain a signature from the card on an arbitrarily chosen document, that is different from the one displayed on the screen and confirmed by the user. We propose a solution to this problem which is based on a new concept called conditional signature. This leads to a new paradigm where digital signatures are not considered as non-repudiable proofs, at least until a short deadline.
Keywords: *untrusted terminal, conditional signature, message authentication, electronic signature, smart card*

1 Introduction

We consider electronic commerce applications, where a user – a sole human being – wishes to make business with a remote partner. If sensitive data travels through an insecure network, it should be protected e.g. by cryptographic algorithms. When a protocol participant is supposed to be a human, it is implicitly assumed that she uses a *terminal* (e.g., a PC), which stores cryptographic keys, performs cryptographic computations, and handles network connections on behalf of her. It is also implicitly assumed that the terminal is trusted by the user for behaving as expected, and in particular for not compromising the security of the user (e.g., by leaking her keys).

Unfortunately, most terminals cannot be called 'trusted'. Either because the party operating the terminal is not trusted by the user, or the user cannot be convinced that the terminal does not have hidden features. Moreover, it is often very hard to check if the hardware or software of the machine has been tampered with.

Although most systems use smart cards as security measures, cards lack user interface, which enables new attacks, that are impossible in case of traditional computers. While smart cards can produce strong digital signatures, they cannot verify that the message they sign was not altered by a malicious terminal. This is why Schneier and Schostack [13] called smart cards 'handicapped computers'.

2 Related work

Terminal identification is perhaps the most basic problem addressed in the literature. In this most simple model, terminals are categorized into two main groups: trusted terminals and untrusted terminals. Users trust terminals in the former group completely, while they want to avoid terminals in the latter one. In order to help users to distinguish between trusted and untrusted terminals, terminals are required to authenticate themselves before they are used. It is assumed that only trusted terminals are able to authenticate themselves correctly. Thus, trusted terminals have to be tamper resistant, otherwise tampered terminals could serve an attacker while still being able to authenticate themselves. Unfortunately, it is very hard to identify tampered terminals. Thompson [15] demonstrates that it is impossible to verify the integrity of a terminal, without a separate trusted machine that has direct access to the terminal's 'hard drive'.

Asokan et al. [2] and Rank and Effing [10] show a simple protocol, that – using smart cards and one-time passwords – enables the identification of fake terminals. Asokan et al. propose the use of distance bounding protocols to prevent the attacks of tampered terminals.

This paper relies on the work of Asokan et al. by assuming, that there is a group of trusted terminals that the user is able to identify. However, our solution also enables the user to use untrusted terminals and make certain sensitive operations (like digital signatures) on them.

The problem of man-in-the-middle attacks of untrusted terminals was addressed by Abadi et al. [1] first, by analyzing the dangers of delegation of rights to a terminal. They envision a smart card that has peripherals to communicate directly with the user, and they show secure protocols for such a device. Unfortunately, after more than 10 years of technical development, such a smart card is still not a feasible assumption.

The solution of Clarke et al. [5] uses a super-smart card, a device equipped with a digital camera, which is connected to the network while continuously monitoring the screen of the terminal. Although this device is currently technically infeasible, this solution would enable authentic communication without requiring the user to perform any calculations. However, it is hard to make such a complex device tamper resistant.

According to Rivest [11] there is a fundamental conflict between having a secure device and having a 'reasonable customizable user interface' that supports downloading of applications. He suggests, that digital signature should not be considered non-repudiable proof, but simply plausible evidence. Thus users should be given well-defined possibilities for repudiating such signatures.

In contrast to solutions based on super-smart cards, the one presented in paper does not require the card to have a user interface or any special peripherial, but can be implemented using smart cards that exist today.

Stabell-Kulo et al. [14] proposed a protocol for sending authentic messages from untrusted terminals. Their solution gains authenticity by encryption using a one-time-pad together with a monoalphabetic substitution table. In contrast to the work of Stabell-Kulo et al., the solution presented in this paper does not require the user to perform cryptographic operations or to memorize cryptographic keys.

Another approach takes advantage of the fact, that the smart card is physically close to the user. Berta and Vajda [4] propose a solution, where the user can send authentic biometric (audio or video) messages from untrusted terminals. Although the solution of Berta and Vajda are feasible with today's smart cards, it relies on the fuzziness of biometry by assuming that it takes more time for the attacker to counterfeit biometric messages, unlike the solution presented in this paper, which is a purely cryptographic one.

Since smart cards did not solve the problem of untrusted terminals, another idea emerged. Pencil-and-paper cryptography (or human-computer cryptography) tries to give the user methods to protect the secrecy or authenticity of the message without the help of a smart card. Among historical methods (like the book cipher) the one-time-pad can be considered quick and easy enough for the limited computational power of the human. However, in case of long messages the user would need secure storage space for long one-time keys. The solitaire algorithm of Schneier [12] provides strong encryption, and uses a deck of card for keying. Although it is optimized for use by humans, in case of long messages, encryption requires a significant amount of time, so this algorithm is more suitable for secret agents than every-day people. Methods proposed by Naor and Pinkas [8] rely on visual cryptography ([9]), which uses transparencies placed on the computer's screen. Their algorithm relies on a one-time-pad, where the xor operation is accelerated by the fast visual processing of the human being. Matsumoto [7] developed a human identification scheme, that enables challenge and response identification of humans at untrusted terminals. Unfortunately, this scheme can be undermined if the attacker can use human interaction too.

In contrast to solutions based on human-computer cryptography, the one presented in this paper does not require the user to perform any cryptographic operations. It relies on a trusted smart card and assumes that a user has the opportunity to regularly access trusted terminals. The authors present a more detailed review on literature in [3]

3 Model

A user wants to generate digital signatures at untrusted terminals. User U has limited memory and computational power. While U is able to memorize some passwords or PIN codes, she cannot memorize cryptographic keys, neither can she perform cryptographic computations. The private key of U is stored and the signatures are generated by a smart card C in possession of user U.

Card C is a trusted personal microcomputer without direct interfaces towards U. C is connected to the terminal so messages between C and U, must pass through the untrusted terminal. C is manufactured by a trusted manufacturer and and hence, it is assumed to function correctly, so C does not try to leak the private key of U or to use the private key without authorization.

We denote by M the intended recipient of the digital signature generated by C. M could be a service provider, a merchant, another user, etc.

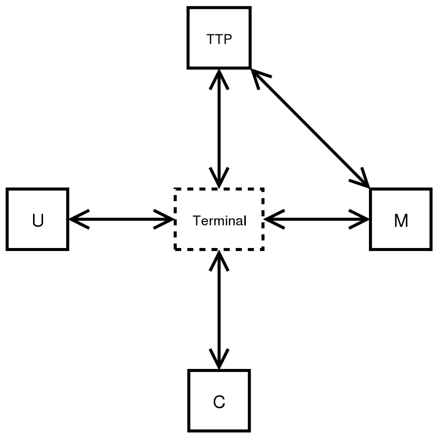

Figure 1. Physical connections

In certain protocols we are going to assume that there is a trusted third party TTP in the system that both U and M trust. Depending on the exact procotol used, TTP may have different functions. (e.g. see Section 5.1) The physical connections defined so far are illustrated in Figure 1.

We assume that the untrusted terminal T in front of U is fully under the control of an attacker. This means that the attacker is able to steal and abuse any PIN code[1] typed in by U on the keyboard of the terminal, to send fake messages to U through the display of the terminal, and to modify messages that U sends to C for signing before passing them on to C. Thus, the attacker can obtain signature from the smart card for an arbitrary message.

However, we assume, that from time to time, U has access to C from trusted terminals too. A trusted terminal could be the home PC of U, or a terminal operated by a trusted organization and believed to be tamper resistant (e.g., an ATM machine). Of course, in order to use a terminal for this purpose, it must be properly authenticated first.

Thus, two different scenarios exist for U: one where the terminal is untrusted, and another one where the terminal is trusted. In the first case, all logical channels between U and the other actors C, M, and TTP are insecure, as they pass through the untrusted terminal, and thus they are controlled by the attacker. In the second case, secure logical channels between U and the other actors C, M, and TTP can be established, since U and the trusted terminal together can be viewed as a computer that can set up cryptographically secured connections with C, M, and TTP. Since C, M, and TTP are able to perform cryptographic computations, they can establish encrypted and authenticated channels between each other, regardless of the terminal in front of U.

4 Generic protocol

In order to detect attacks mounted by the attacker, we propose a framework that allows users to sign messages on untrusted terminals with the help of their smart cards, review the signatures later in a trusted environment, and revoke the fake ones (or authorize only the valid ones). This is made possible by using conditional signatures. In this section, we first introduce the concept of conditional signatures, and then propose a generic protocol that uses conditional signatures.

4.1 Conditional signatures

Conditional signatures were introduced by Lee and Kim [6], who used this concept for solving fair exchange problems without expensive cryptographic primitives like verifiable escrow. A conditional signature of U on a message m is U's ordinary signature $sig_U(m, c)$ on m and a description of a condition c. If $sig_U(m, c)$ is correct and condition c is true, then $sig_U(m, c)$ is considered to be equivalent with $sig_U(m)$, U's ordinary digital signature on m. However, if c is false, then U is not responsible for m. Intuitively, U's conditional signature is U's commitment: 'I signed m, but if c is not true, then my signature on m is not valid.'

[1]Although PIN codes are useful against e.g. card theft, they provide little protection against the threat of untrusted terminals, so their use is not discussed in this paper.

4.2 Core protocol

We have shown, that it is impossible to prevent the terminal from obtaining signature from the card on an aribitrarily choosen document. Therefore, instead of generating an ordinary signature, we propose that C generates a conditional signature such that it is guaranteed that the condition cannot become true before a certain amount of time has passed. This should leave time for the user to move to a trusted terminal for checking the signatures generated by the card, and to enforce that the conditions of the fake signatures can never become true.

Although most smart cards have no internal clock, they may acquire the current time from secure time servers (e.g. Berta and Vajda [4] show a protocol for this problem). Thus, we assume, that C knows the current time.

These thoughts lead to the following generic protocol: (Note, that while steps 1-4 happen at an untrusted terminal, steps 5 and 6 are performed using a secure terminal and via secure channels.)

Step 1: $U \rightarrow T$: m
When U wants to sign message m at an untrusted terminal, she first provides the terminal with m, then she inserts her card C into the terminal's smart card reader.
Step 2: $T \rightarrow C$: m
Step 3: $C \rightarrow T$: $c, sig_U(m, c)$
The card logs m and computes the conditional signature $sig_U(m, c)$ of U on m, where c is a condition that includes (among other things) the deadline t. The conditional signature will not be valid before t, and will become valid after t if and only if the other conditions in c hold.
Step 4: $T \rightarrow M$: $(m, c, sig_U(m, c))$
Step 5: $C \rightarrow U$: M, m, c
Later, but before the deadline t, U reviews the list of messages logged by C at a trusted terminal. This can be done, for instance, on her home PC. Before outputting its log, C authenticates the terminal to be sure that it is a trusted one.
Step 6: For each message m that U intended to sign, U ensures that the condition c becomes true; for the rest of the messages, U ensures that the condition becomes false. This might involve additional steps and further communication with M or TTP. See Section 5 for details.

A third party needs to check if the digital signature $sig_U(m, c)$ of the card is correct and condition c is true in order to verify a conditional signature.

4.3 External logging

There are two problems with the above core protocol: First, C is required to log every message that it signed. However, C is a smart card with limited storage capacity. In some applications (where large messages have to be signed), it may be infeasible for C to store every message. Second, C is required to input the whole message to be signed. Again, if messages are large, then this may be impossible or impractically slow. The first problem can be solved by outsourcing the logging function to an external log server. This log server needs to be trusted by U only, so it may even be the user's home PC if it is online. The second problem can be solved, by letting the terminal compute a hash of the message to be signed; only this hash is passed to the smart card, and the conditional signature is generated on this hash.

In order to include these extentions in the generic protocol only steps 2, 3 and 5 need to be changed. In the description below, LS denotes the log server and it is assumed that C and LS shares a symmetric key $K_{C,LS}$.

Step 1: $U \rightarrow T$: m (Same as before.)
Step 2: $T \rightarrow C$: $h(m)$ (where h is a hash function)
Step 3.1: $C \rightarrow T$: $LS, C, \{C, n, h(m)\}_{K_{C,LS}}$ (where n is a sequence number maintained by C)
Step 3.2: $T \rightarrow LS$: $(C, m, \{C, n, h(m)\}_{K_{C,LS}})$ to LS.
Step 3.3: LS decrypts $\{C, n, h(m)\}_{K_{C,LS}}$ and verifies if the decryption was successful by checking that the first field of the decrypted message is C, and by verifying that n has not been used so far by C. LS also computes the hash of m and compares the result with $h(m)$ received in the encrypted part of the message. If any of the verifications above is unsuccessful, then LS aborts the protocol. Otherwise, LS logs (C, n, m), and sends an acknowledgement:
$LS \rightarrow M$: $mac_{K_{C,LS}}(LS, C, n, h(m))$
Step 3.4: $T \rightarrow C$: $mac_{K_{C,LS}}(LS, C, n, h(m))$
Step 3.5: C verifies the acknowledgement, and if it is correct, then it continues the core protocol.
$C \rightarrow T$: $(c, sig_U(h(m), c))$
Step 4: $T \rightarrow M$: $(m, c, sig_U(h(m), c))$
Step 5: Later, but before t, U downloads the logged messages from LS and reviews them at a trusted terminal. Before this operation takes place, the terminal is authenticated in order to be sure that it is a trusted one.
Step 6: Same as before: U ensures that only those conditions become true where she intended to sign the corresponding messages.

5 Finalizing signatures

As we have seen in the previous section, it is always required that condition c is not true before a given deadline t. This is an inherent requirement in our scheme, which gives the user some time to move to a trusted terminal and review the signatures generated so far by her smart card. However, various approaches are possible to define what happens after t, or more precisely, how the user can enforce the truth value of c after t. In this section, we discuss some of these approaches.

First of all, we note that in most of the applications, it is desirable that the status of a digital signature does not vary in time. In our scheme, this is not fully supported, since every signature is invalid until t, and then it may become valid. There is a good reason to allow this, namely to mitigate the untrusted terminal problem. We note, however, that the schemes that we propose below guarantee that after t, the status of the signature becomes stable. In particular, once the user reviewed and accepted a signature, it cannot be revoked.

It seems to be a good idea to define a default truth value for c after t that cannot be changed later, because this ensures that the status of each signature will indeed become stable after t independently of the negligence of the involved parties. In other words, if the user does not take any steps until t to confirm or to revoke a signature, then the status of the signature will take the default value at t, and the user can no longer do anything about it. Depending on the default truth value, we can distinguish between two classes of protocols. Protocols in the first class support the *default accept* approach, where a signature automatically becomes valid after t (and remains valid forever) unless it is explicitly revoked by the user before t. Unfortunately, all of these protocols require a TTP. Protocols in the second class support the *default deny* approach, where a signature remains invalid after t (and forever), unless it is explicitly confirmed by the user before t. Protocols of this class might not be suitable in certain applications, because users may tend to forget to confirm conditional signatures, which means that they revoke them.

In the following two subsections, we present two protocols and corresponding conditions in order to illustrate the finalization of the status of signatures.

5.1 'Simple deadline' condition

In this scheme, condition c is true, if deadline t has passed and TTP countersigned the conditional signature. This scheme follows the default accept approach, but a similar scheme can be constructed for the default deny approach too.

U signs message m at an untrusted terminal:
Step 1: $U \rightarrow T$: m
Step 2: $T \rightarrow C$: m
Step 3: $C \rightarrow T$: $t, TTP, sig_U(m, t, TTP)$
Step 4: $T \rightarrow M$: $t, TTP, sig_U(m, t, TTP)$

U reviews signed messages from a trusted terminal:
Step 5: $C \rightarrow U$: M, m, t, TTP
Step 6: If U did not intend to sign message m:
$U \rightarrow TTP$: *'I revoke my signature* $sig_U(m, t, TTP)$*'*.
Otherwise, U does not need to act.
Step 7: $M \rightarrow TTP$: $t, TTP, sig_U(m, t, TTP)$
Step 8: If U did not revoke the signature before t, then:
$TTP \rightarrow M$: $sig_{TTP}(sig_U(m, t, TTP))$

A third party needs to check if the digital signatures $sig_U(m, t, TTP)$ and $sig_{TTP}(sig_U(m, t, TTP))$ are correct in order to verify the conditional signature.

5.2 'Trapdoor function' condition

Our previous protocol had two very expensive operations: It required TTP to compute a digital signature for every signature of every user, and two signatures needed to be verified in order to verify the conditional signature. Our next protocol follows the default deny approach. Here, condition c contains $h(r)$, the hash of nonce value r. Condition c is true if r is also presented.

U signs message m at an untrusted terminal:
Step 1: $U \rightarrow T$: m
Step 2: $T \rightarrow C$: m
Step 3: $C \rightarrow T$: $h(r), sig_U(m, h(r))$
Step 4: $T \rightarrow M$: $h(r), sig_U(m, h(r))$

U reviews signed messages from a trusted terminal:
Step 5: $C \rightarrow U$: $M, m, h(r), r$
Step 6: If U intended to sign message m then:
$U \rightarrow M$: r (Otherwise, U does not need to act.)

A third party needs to check if the digital signature $sig_U(m, h(r))$ is correct and r is presented in order to verify the conditional signature.

This protocol is efficient, it does not require any help from TTP. However, it does not fully support the requirement, that the value of condition c cannot be changed after a certain deadline. Although U is able to change the value of c from false to true any time, she cannot do it vice versa. Help from TTP is required if we would like to freeze the value of c after a deadline. Help from TTP is also needed if we would like to construct a similar scheme that follows the default accept approach. In this case TTP needs to be involved, in order to send r to M, unless U revokes the signature before a certain deadline.

6 Conclusion and future work

We studied the problem of how a user at an untrusted terminal can generate digital signatures using her smart card without allowing the terminal to obtain the signature of the user on an arbitrarily chosen message. Our solution is based on a new concept called conditional signature, leading to a new paradigm where digital signatures are not considered as non-repudiable proofs, at least until a short deadline.

One common principle can be identified in all works on the untrusted terminal problem: a system is not suitable for providing security services (like encryption or digital signature) without a trusted hardware component having a secure channel towards the user of the system. In our solution, the trusted hardware component is the smart card of the user, and the secure channel is provided in an off-line manner through a trusted terminal such as the home PC of the user.

In our future work, we intend to further analyze and optimize the protocols presented in this paper. We also want to address privacy problems that arise in our proposal. In particular, in our current schemes, the TTP can link the issuer of a signature with its intended recipient, which may not be desirable in some applications. We also plan to examine if a user is able to identify the terminals that mount attacks on her, and if she is able to prove the attack to a third party.

References

[1] M. Abadi, M. Burrows, C. Kaufman, and B. Lampson. Authentication and Delegation with Smart-cards. Theoretical Aspects of Computer Software: Proc. of the International Conference TACS'91, Springer, Berlin, Heidelberg, 1992.

[2] N. Asokan, H. Debar, M. Steiner, and M. Waidner. Authenticating Public Terminals. Computer Networks, 1999, 1999.

[3] I. Z. Berta, L. Buttyán, and I. Vajda. Mitigating the untrusted terminal problem using conditional signatures. CrySyS Lab Technical Report, http://www.crysys.hu/publications/fi les/BertaBV2004condsig.pdf, 2003.

[4] I. Z. Berta and I. Vajda. Documents from Malicious Terminals. SPIE Microtechnologies for the New Millenium 2003, Bioengineered and Bioinspired Systems, Maspalomas, Spain, 2003.

[5] D. Clarke, B. Gassend, T. Kotwal, M. Burnside, M. v. Dijk, S. Devadas, and R. Rivest. The Untrusted Computer Problem and Camera-Based Authentication, 2002.

[6] B. Lee and K. Kim. Fair Exchange of Digital Signatures using Conditional Signature. SCIS 2002, Symposium on Cryptography and Information Security, 2002.

[7] T. Matsumoto. Human-Computer cryptography: An attempt. In ACM Conference on Computer and Communications Security, pp 68-75, 1996.

[8] M. Naor and B. Pinkas. Visual Authentication and Identifi - cation. Lecture Notes in Computer Science, volume 1294, 1997.

[9] M. Naor and A. Shamir. Visual Cryptography. Lecture Notes in Computer Science, vol 950, pp 1–12, 1995, http://citeseer.nj.nec.com/naor95visual.html, 1995.

[10] W. Rankl and W. Effi ng. Smart Card Handbook. John Wiley & Sons, 2nd edition, ISBN: 0471988758, 1997.

[11] R. Rivest. Issues in Cryptography. Computers, Freedom, Privacy 2001 Conference http://theory.lcs.mit.edu/~rivest/Rivest-IssuesInCryptography.pdf, 2001.

[12] B. Schneier. The Solitaire Encryption Algorithm. http://www.counterpane.com/solitaire.htm, 1999.

[13] B. Schneier and A. Shostack. Breaking up is Hard to do: Modelling security threats for smart cards. USENIX Workshop on Smart Card Technology, Chicago, Illinois, USA, http://www.counterpane.com/smart-card-threats.html, 1999.

[14] T. Stabell-Kulo, R. Arild, and P. Myrvang. Providing Authentication to Messages Signed with a Smart Card in Hostile Environments. Usenix Workshop on Smart Card Technology, Chicago, Illinois, USA, May 10-11, 1999., 1999.

[15] K. Thompson. Reflections on Trusting Trust. Communication of the ACM, Vol 29. No. 8, August, 1984 pp 761-763, 1984.