PRIVACY PROTECTING PROTOCOLS FOR REVOKABLE DIGITAL SIGNATURES

István Zsolt BERTA
Laboratory of Cryptography and Systems Security,
Department of Telecommunications,
Budapest University of Technology and Economics
istvan.berta@crysys.hit.bme.hu

Levente BUTTYÁN
Laboratory of Cryptography and Systems Security,
Department of Telecommunications,
Budapest University of Technology and Economics
buttyan@hit.bme.hu

István VAJDA
Laboratory of Cryptography and Systems Security,
Department of Telecommunications,
Budapest University of Technology and Economics
vajda@hit.bme.hu

Abstract Consider an application where a human user has to digitally sign a message. It is usually assumed that she has a trusted computer at her disposal, however, this assumption does not hold in several practical cases, especially if the user is mobile. Smart cards have been proposed to solve this problem, but they do not have a user interface, therefore the user still needs a (potentially untrusted) terminal to authorize the card to produce digital signatures. In order to mitigate this problem, we proposed a solution based on conditional signatures to provide a framework for the repudiation of unintended signatures. Our previous solution relies on a trusted third party who is able to link the issuer of the signature with the intended recipient, which may lead to severe privacy problems. In this paper we extend our framework and propose protocols that allow the user to retain her privacy with respect to this trusted third party.

1. Introduction

Cryptographic protocols are often described informally in terms of message passing between hypothetic principals usually called *Alice* and *Bob*. These principals perform cryptographic computations and communicate through a network. 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, many terminals (especially public ones) should not be trusted by the user. For instance, a terminal that is installed at a public place (e.g., a PC in a hotel or in an airport lounge), or a terminal operated by an untrusted principal (e.g., a POS terminal of an unknown merchant in a foreign country) fall into this category. In the former case, anyone can access the terminal and install malicious software on it that alters its behavior; in the latter case, the operator of the terminal can easily do the same.

In order to defend against untrusted terminals, the user may carry a trusted computer with her everywhere. A convenient solution is a smart card, because it is small and users are already used to carry them in their pockets. However, smart cards lack user interface, which enables various new attacks, that are not possible in case of traditional computers. This is why Schneier and Schostack [Schneier and Shostack, 1999] called smart cards 'handicapped computers'. A typical example for such an attack is the following man-in-the middle attack: If the user wishes to compute a digital signature using her smart card, the terminal can send a different message to the card, a message the user would never want to sign. Unfortunately, the user at an untrusted terminal cannot prevent this attack, so the malicious terminal may obtain the user's digital signature for an arbitrary message.

In [Berta et al., 2004], we proposed a solution that could be used to solve or at least alleviate the above problem. Our solution relies on a concept called *conditional signature* [Lee and Kim, 2002]. In our protocol, whenever the user wants to sign a message at an untrusted terminal, the card adds a condition to it before the message is signed. Thus, the terminal does not obtain the user's regular signature, but it obtains her conditional signature (a signature on the message together with the condition). A conditional signature is equivalent with a regular signature if and only if the condition is fulfilled and the signature is valid. In our model, we assume that from time to time the user has access to trusted terminals too. Whenever she approaches a trusted terminal, she may review messages she signed at untrusted ones, and – depending on the exact protocol – she may fulfill or prevent their appropriate conditions, and thus con-

firm or revoke the signatures. If a user does not confirm or revoke a conditional signature before a certain deadline, the condition obtains a default truth value. In a nutshell, our solution is a framework for the controlled revocation of digital signatures.

From a practical point of view, those conditional signature protocols seem to be most usable, that make signatures automatically confirmed, unless they are explicitly revoked by the user. We assume, that most terminals – though untrusted – do not actually mount attacks on the user. Similarly, most users are not criminals and do not repudiate messages they did intend to sign. Protocols that make signatures automatically confirmed allow well-behaving merchants (terminals) to do business successfully even with those users who tend to forget to confirm their signatures.

Unfortunately, all of these protocols that automatically confirm signatures unless revoked require a trusted third party to operate. (see Section 3.4) In these protocols, the trusted third party is able to link the user with the recipient of the digital signature, which may lead to severe privacy problems in some applications. The user may trust the TTP for assisting her in revoking unintended signatures, but she may not want the TTP to know when, where and on what messages she generated signatures (especially intended ones). Requiring the TTP to be trusted only for the revocation of unintended signatures makes users less reluctant to use its services, and thus allow more service providers to get into the signature revocation service provider business. If the TTP would need to know less about the user, perhaps more organizations could fulfill this task, and conditional signature schemes protecting the user from the attacks of malicious terminals could become more widespread.

In this paper, we propose protocols based on smart cards and conditional signatures, that allow the user to retain her privacy (and often provide even unconditional privacy) with respect to the trusted third party.

The organization of the paper is the following: In Section 2, we report on related work. In Section 3, we define our system model and present our solution based on conditional signatures. In Section 4 we show three protocols that extend our previous solution by allowing the user to retain her privacy with respect to the trusted third party. Finally, in Section 5, we conclude the paper.

2. Related work

2.1 Untrusted terminals

Terminal identification is perhaps the most basic problem addressed in the literature. In this most simple model, terminals are categorized into two main groups: terminals trusted by the user and untrusted terminals. Naturally, trusted terminals have to be tamper resistant, otherwise tampered terminals could serve an attacker while still being able to authenticate themselves. Asokan et al.

[Asokan et al., 1999] and Rank and Effing [Rankl and Effing, 1997] show a simple protocol, that – using smart cards and one-time passwords – enables the authentication of terminals.

Other papers consider the terminal untrusted, and propose solutions for its secure usage. The problem of man-in-the-middle attacks of untrusted terminals was addressed by Abadi et al. [Abadi et al., 1992] first, who analyzed the dangers of delegation of rights to a terminal. They examined, what additional peripherals a card needs to have to solve the above problem. Clarke et al. [Clarke et al., 2002] also propose a solution based on a super smart card. Other authors tried to come up with solutions that are workable with cards that exist today. [Stabell-Kulo et al., 1999], [Berta and Vajda, 2003]

According to Rivest [Rivest, 2001] 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 signatures should not be considered non-repudiable proofs, but simply plausible evidence. Thus, users should be given well-defined possibilities for repudiating such signatures.

Since smart cards did not solve the problem of untrusted terminals, other ideas 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. The most notable solutions based on human-computer cryptography are [Schneier, 1999] and [Matsumoto, 1996], and those, that rely on visual cryptography ([Naor and Shamir, 1995], [Naor and Pinkas, 1997]). Unfortunately, most of these solutions are rather awkward, and require the user to perform complex or unusual operations.

None of the above solutions try to protect the privacy of the user.

2.2 Anonymous payment systems

In case of any protocol that protects privacy, anonymous communication is a key element. Mixes yield a typical solution for anonymous communication, they were introduced in [Chaum, 1981]. A Syverson et al. [Syverson et al., 1997] describe a famous solution that uses a network of mixes for communication on the WWW. A simpler solution is [Anonymizer Inc., 1999], that provides a proxy for its users.

Electronic payment systems require a balance between anonymity and traceability. Chaum introduced an anonymous payment system (the late DigiCash), it is based on blind signatures. [Chaum, 1982] The foundations of some other famous anonymous payment systems are introduced in [Brands, 1994] and [Franklin and Yung, 1992]. Jakobsson et al. [Jakobsson and Raïhi, 1998] also propose an electronic payment system in which anonymity is based on a mix-network.

The trusted third party in this paper is in a position very similar to that of a bank or trustee in the above papers. However, in contrast to the above papers, we discuss the privacy of a user who is only able to perform cryptographic operations using her smart card beyond the untrusted terminal. We also rely on the existence of methods for anonymous communication.

3. A protocol without privacy

3.1 Model and assumptions

We consider a system with human users who want to generate digital signatures at untrusted terminals. *User U is a human* and has limited memory and computational power. She is able to memorize some passwords or PIN codes, but cannot memorize cryptographic keys, neither can she perform cryptographic computations. For this reason, the private key of U is stored and the signatures are generated by a smart card C in possession of user U.

Essentially, *smart card C* is a trusted personal microcomputer without direct interfaces towards U. C is connected to the terminal in front of U, and all messages between C and U, must pass through the terminal. Smart card C is assumed to be trustworthy and tamper-resistant. The card is *tamper-resistant*, so it is impossible to alter its behavior, reverse engineer it or extract information from it. Card C is *trustworthy*, because it is manufactured by a trusted manufacturer. Since smart cards undergo extremely rigorous evaluation and certification, and not even the manufacturer can alter their behavior after issuance, we consider this assumption to be justified. Thus, if properly authenticated (e.g. by a challenge and response method), smart card C is assumed to be a trusted party.

We assume that the *untrusted terminal* T in front of U is fully under the control of an attacker, who may have installed all kinds of malicious software on the terminal before U started to use it. Since T is able to modify messages that U sends to C for signing, the attacker can obtain a 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. Such a trusted terminal could be the home PC of U, but it can also be 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.

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. It is a well-spread scenario that terminal T is operated by service provider M. So, we assume, that M and T may cooperate against the user.

We are going to assume that there is *trusted third party* TTP in the system that both U and M trust. However, user U would still like to retain her privacy with

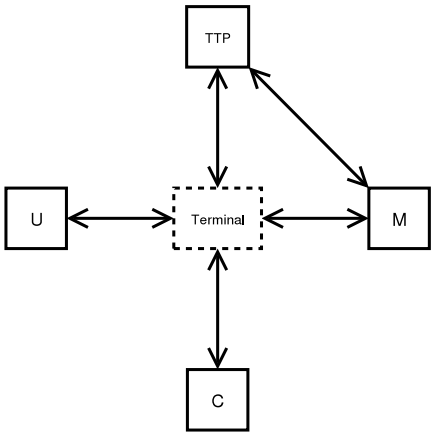

Figure 1. Physical connections

respect to TTP. This means, that while U trust TTP only for the revocation of unintended signatures, but U would like to prevent TTP from knowing where, when and what messages she signed. Thus, she would like to prevent TTP from knowing, which merchants or service providers she does business with. TTP follows the given protocols, and does not try to cheat by breaking into the terminal or intercepting messages for other parties. Neither does TTP collaborate with T or M to reveal the identity of the user.

A note on the background of this latter assumption: The identity of the user can neither be hidden from M (otherwise M would not be able to verify the signature of U), nor from T (because U is physically present at T who can even take a photograph of her). This means, in case TTP collaborates with T or M, the user is unable to retain her privacy with respect to TTP.

The physical connections defined so far are illustrated in Figure 1.

3.2 Conditional signatures

Conditional signatures were introduced by Lee and Kim [Lee and Kim, 2002], 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."*

We proposed the use of conditional signatures against malicious terminals in [Berta et al., 2004]. Since it seems to be impossible to prevent the terminal from obtaining signature from the card on an arbitrarily chosen document, instead of generating an ordinary signature, we proposed that C generates a conditional signature.

In our proposed schemes, it is guaranteed that the condition cannot become true before a certain amount of time has passed. This leaves time for the user to access a trusted terminal for checking the signatures generated by the card, and to ensure that the conditions of the fake signatures can never become true.

3.3 The protocol

In this subsection we present a protocol we proposed in [Berta et al., 2004], that allows the user to repudiate signatures computed at untrusted terminals, but does not allow her to retain her privacy with respect to TTP.

User U approaches an untrusted terminal:

Step 1: $U \rightarrow T$: m
U types message m using the keyboard of the terminal.

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. In the most simple case, condition c is the following string:
"My signature on m is valid if and only if deadline t has passed and TTP countersigned this conditional signature."

Step 4: $T \rightarrow M$: $(m, c, sig_U(m, c))$

Later, at a trusted terminal:

Step 5: $C \rightarrow U$: M, m, c
Before deadline t, U reviews the list of messages logged by C at a trusted terminal. This can be done, for instance, by U returning to her home and inserting C into the smart card reader of her home PC. Before outputting its log, C authenticates the terminal to be sure that it is a trusted one.

If user U did not intend to sign m, then:

Step 6: $U \rightarrow TTP$: *I revoke my signature* $sig_U(m, c)$.

Otherwise, the signature becomes confirmed after deadline t:

Step 7: $M \rightarrow TTP$: $c, sig_U(m, c)$

If deadline t has passed and user U did not revoke her signature in Step 6:

Step 8: $TTP \rightarrow M$: $sig_{TTP}(sig_U(m, c))$
TTP countersigns the conditional signature.

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.

We assume, that the card is able to log messages. While smart cards have severe memory limitations, it is possible for them to outsource logging to an external log server. In [Berta et al., 2004] we show a protocol that solves this problem.

However, the above protocol does require the user to reveal her identity to TTP. Naturally, the easiest way to protect the privacy of the user from TTP would be to eliminate TTP from the protocol. In the next subsection, we examine the possibility of this option.

3.4 Need for a trusted third party

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 any conditional signature scheme must 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 user U does not confirm or revoke the signature before deadline t, then its status will take a default value and U 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 deny* approach, where a signature remains invalid after t (and forever), unless it is explicitly confirmed by the user before t. Such protocols may operate without TTP, we demonstrate an example for this in [Berta et al., 2004]. However, protocols of this class might not be suitable in certain applications, because users may tend to forget to confirm conditional signatures, which means that the signatures are automatically revoked. Protocols in the second class support the *default accept* approach, where a signature automatically becomes valid after t (and remains so forever) unless it is explicitly revoked by the user before t. The majority of terminals do not mount attacks on the user, so forgetting to review a signature rarely has severe consequences. We consider it a critical issue, not to prevent terminals from doing business with those users who tend to forget to confirm signatures. In these protocols it is the interest of the user to review signatures made on untrusted terminals, so she is less likely to forget this important protocol step.

While protocols supporting the *default accept* approach seem more practical, all of them require a TTP. In these protocols, after Step 3, the conditional signature is in the hands of the untrusted terminal (or the untrusted merchant). Condition c (and the signature of the user) will become valid after a certain deadline automatically, unless U revokes it. If U has to revoke it at T or M, either of T or M could simply repudiate the receipt of such a revocation. While smart card C is also a trusted party, C is not online continuously. Thus, it is impossible for C to maintain a signature revocation list that is trusted and can be checked by all other parties. Thus, TTP is needed to enforce a default value for the signature and to handle revocation.

It seems that all of the practical protocols require the help of a TTP. However, if the TTP is able to log all the contracts a user signs, it is in a very critical position. Few organizations would be trusted enough to be a TTP in such protocols. We reckon, that if the protocol prevented the TTP from linking the user with the merchant or service provider, more organizations would qualify to be a TTP. In the next section we propose three protocols that would allow user U to retain her privacy with respect to TTP.

4. Protocols to protect the user's privacy

Our goal is to develop a protocol for signature revocation, that allows U *to retain her privacy with respect to* TTP. While U may trust TTP for signature revocation, U does not want TTP to know, where, when and what messages she wanted to sign. Thus, apart from hiding the message from the eyes of TTP, U should hide her identity too, and the fact that she sent a message to M. Thus, she needs to protect the *confidentiality of message* m *and digital signature* $sig_U(x)$ she computes on any x.

It is clear, that she does not want to protect them against M, because she intends to send message m to service provider M. Moreover, she cannot protect m against T, because she types the message using the keyboard of the terminal in Step 1.

All three protocols we propose follow the concept of the protocol in Section 3.3. In Step 3, the smart card sends a cryptogram encrypted by the public key of TTP that contains condition c along with revocation token r. This cryptogram is forwarded to the merchant and later to TTP in Step 7. The user receives revocation token r from the card via a trusted terminal, and may repudiate her signature by submitting r to TTP in Step 6 via an *anonymous channel*. We assume, that such an anonymous channel exists. (Naturally, anybody may repudiate the signature using r, so it is advisable not to let the untrusted terminal compute it. Otherwise, T would be able to repudiate messages in the name of U, and could spoil the reputation of the user.)

TTP decrypts the cryptogram that was sent by the merchant in Step 7, and enforces condition c to become true (in Step 8) if the revocation token r inside the cryptogram was not submitted before. Revocation token r is a random number, statistically independent from the identity of U, the contents of message m or the value of conditional signature $sig_U(m, c)$. Based on r, TTP is unable to link U with M. (Note, that the identity of M is not hidden from TTP.)

While TTP needs to store revocation token r, it may not be necessary to store it forever. This problem could be solved e.g. by introducing a lapse time, so TTP could refuse to validate very ancient conditional signatures.

4.1 Bit commitment

Our first protocol follows the spirit of bit commitment protocols. [Schneier, 1996] User U commits herself to her signature to M. However, U does not reveal her signature to M immediately, only after deadline t contained in c. In this case, condition c is the following string: *"My signature on the above message is not valid before deadline t."*

In contrast to the classical bit commitment, the "reveal" phase is not performed by user U (because of reasons described in 3.4), but by trusted third party TTP in Step 8. Moreover, in this case not even TTP is allowed to read the bits U committed herself to. Thus, in this case not all known bit commitment methods can be used (e.g., the solution proposed in [Naor, 1991] cannot be used in this case), only those that do not require sending bits (the user committed herself to) in cleartext when they are revealed.

We propose the following protocol to protect the privacy of U with respect to TTP:

Step 1: $U \rightarrow T$: m

Step 2: $T \rightarrow C$: m

C generates random symmetric key k and revocation token r.

Step 3: $C \rightarrow T$: $c,\ E_k[sig_U(m)],\ E_{TTP}(r, k, c)$

Step 4: $T \rightarrow M$: $m,\ c, E_k[sig_U(m)],\ E_{TTP}(r, k, c)$

Unlike in the protocol described in Section 3.3, terminal T is unable to verify the signature in this step. However, C is a device trusted by T, so T may assume, that C follows the protocol, and is not sending garbage.

Later, at a trusted terminal:

Step 5: $C \rightarrow U$: M, m, c, r

If user U would like to repudiate the signature on message m then

Step 6: $U \rightarrow TTP$: r (via an anonymous channel)

After deadline t:

Step 7: $M \rightarrow TTP$: $E_{TTP}(r, k, c)$

If deadline t has passed, and r was not submitted to TTP, then:

Step 8: $TTP \rightarrow M$: k

Step 9: M decrypts $E_k[sig_U(m)]$ using k and obtains $sig_U(m)$.

It is an important merit of this protocol, that a third party needs to have m and $sig_U(m)$ only in order to verify the conditional signature of U. Since, this conditional signature is not different from a regular one, its verification requires the same procedure too. Note, that in this protocol TTP does not have to perform a digital signature operation.

In this protocol, U is able to retain a *provable degree of privacy* with respect to TTP. In case everyone behaves honestly, r is not sent in Step 6. In this case TTP only receives cryptogram $E_{TTP}(r, k, c)$. Since r and k are random numbers, and c is the same string in case of all users in the system, each of them is independent from U, m and $sig_U(m)$. Thus, if everyone behaves honestly, U has unconditional privacy.

On the other hand, if U chooses to revoke the signature, TTP receives random number r too. Naturally, r is independent from U, m and $sig_U(m)$, so in this case, the privacy of user U is equal to the one provided by the channel that is used in Step 6 for signature revocation.

4.2 Blind signatures

Our next protocol relies on the concept of blind signatures introduced in [Chaum, 1982]. In this scheme, the cryptogram the card outputs in Step 3 contains the conditional signature of the user. TTP has to countersign the conditional signature in order to validate it, without being able to read it. To prevent TTP from reading the conditional signature, the signing process of TTP is blinded by the card. Smart card C also releases a token that M will be able to use to unblind the signature.

In this case, condition c if the following: *"My signature on the above message is valid if and only if deadline t has passed and TTP countersigned it."*

We are going to use the following notation:

- The signature of the user on message x is denoted as $sig_U(x)$.
- TTP has two key pairs. One of them is an RSA key pair for digital signatures, the other keypair is for an arbitrary algorithm for encryption/decryption.
  - The RSA keypair of TTP for digital signature is the following: the public key is e, the public modulus is m, and the private exponent d. According to the RSA algorithm, the signature of TTP on message x is $x^d \bmod m$.

Note, that an attacker may obtain signature (or decryption) from this keypair on an arbitrary message, so TTP should not use this keypair for any other purpose.

– Encryption of message x with the public encryption key of TTP is denoted as $E_{TTP}(x)$. Decryption of message y with the private decryption key of TTP is denoted as $D_{TTP}(y)$.

- Symbol "$*$" stands for multiplication modulo m. Modulus m should be larger than the largest possible value of $sig_U(x)$.

The proposed protocol is as follows:

Step 1: $U \rightarrow T$: m

Step 2: $T \rightarrow C$: m

Step 3: $C \rightarrow T$: $c, b, E_{TTP}(c, r, sig_U(m, c) * b^e)$
C generates random numbers r and b, where r is a repudiation token and b is going be used to blind the signature of TTP.

Step 4: $T \rightarrow M$: $m, c, b, E_{TTP}(c, r, sig_U(m, c) * b^e)$

Later, at a trusted terminal:

Step 5: $C \rightarrow U$: M, m, c, r

If user U would like to repudiate the signature, then:

Step 6: $U \rightarrow TTP$: r (via an anonymous channel)

After the t deadline:

Step 7: $M \rightarrow TTP$: $E_{TTP}(c, r, sig_U(m, c) * b^e)$

If deadline t has passed and r was not submitted to TTP, then:

Step 8 : $TTP \rightarrow M$: $[sig_U(m, c)) * b^e]^d = [sig_U(m, c)]^d * b$
Step 9: M acquires $[sig_U(m, c)]^d$ using b.

A third party needs to have m, c and $(sig_U(m, c))^d$ in order to verify the conditional signature of U.

Again, this protocol provides a *provable degree of privacy for U with respect to TTP*, since TTP receives only r (which is just a random number) and $E_{TTP}(c, r, sig_U(m, c) * b^e)$. In this latter cryptogram only $sig_U(m, c) * b^e$ carries information that could be connected to user U. However, according to [Chaum, 1982], the blind signature is unconditionally secure, so no algorithm exists that can compute $sig_U(m, c)$ based on $sig_U(m, c) * b^e$ (with a probability better than $1/2^{|sig_U(m,c)|}$) without knowing parameter b. Thus, the degree of privacy user U has is the same as the one provided by the channel that is used

in Step 6 for signature revocation. Moreover, if everyone behaves honestly, U obtains unconditional privacy.

Note, that in the above protocol TTP signs an incoming message without being able to see what it is. Although TTP may authenticate M to prevent denial of service attacks, M may still obtain the signature of TTP on an arbitrary message. Since TTP does not use this private key for anything else, such a signature is useful only if a conditional signature of the user was signed. Although M can compute the cryptogram and repeat Step 7 at will, M can only gain countersignatures on signatures U did not revoke.

4.3 Halving the digital signature

In our final protocol we make use of the fact that even if TTP obtains one half of the bits of the conditional signature of U, TTP is still unable to compute the other half. Meanwhile, if TTP countersigns the half of the bits of the conditional signature, it is authenticated not much less securely than if TTP countersigned the whole signature.

In this case, condition c looks as follows: *"My signature on the above message is valid if and only if TTP countersigned its right half and deadline t has passed."*

According to our notation, $left(x)$ means the left half of the bits of bitstring x, and $right(x)$ means its right half. Naturally, $left(x)||right(x) = x$ where operation $||$ is concatenation. The proposed protocol is as follows:

Step 1: $U \rightarrow T$: m

Step 2: $T \rightarrow C$: m

Step 3: $C \rightarrow T$: $c,\ left[sig_U(c,m)],\ E_{TTP}(r, c, right[sig_U(c,m)])$
C generates random number r. Again, the terminal cannot verify the conditional signature, but since C is a trusted device, T may believe that the conditional signature on m was encrypted by the public key of TTP.

Step 4: $T \rightarrow M$: $m, c,\ left[sig_U(c,m)],\ E_{TTP}(r, c, right[sig_U(c,m)])$

Later, at a trusted terminal:

Step 5: $C \rightarrow U$: M, m, c, r

If user U would like to repudiate the signature, then

Step 6: $U \rightarrow TTP$: r (via an anonymous channel)

After the deadline t:

Step 7: $M \rightarrow TTP$: $E_{TTP}(r, c, right[sig_U(c,m)])$

If deadline t has passed and r was not submitted to TTP, then:

Step 8: $TTP \rightarrow M$: $right[sig_U(c,m)],\ sig_{TTP}(right[sig_U(c,m)])$

Step 9: M computes: $left[sig_U(c,m)] \;||\; right[sig_U(c,m)] = sig_U(c,m)$

A third party needs to have m, c, $sig_U(c,m)$, $sig_{TTP}(right[sig_U(c,m)])$ in order to verify the conditional signature of U.

Again, M may obtain multiple countersignatures on signatures not revoked by U. However, this is not a problem. See the note at the end of Section 4.2. In contrast to the other solutions we proposed, the smart card does not need to perform any complex computation (like blinding or encryption) in this one, apart from generating the digital signature.

5. Conclusion and future work

If a user at a malicious terminal would like to perform sensitive operations like digital signature, she is subject to man-in-the attacks from the terminal. The user may rely on a conditional signature scheme to gain time to review signatures from a trusted terminal and revoke signatures on messages she did not intend to sign. Unfortunately, those conditional signature schemes that allow this and are most suitable for commercial use, have to rely on the help from a trusted third party.

In this paper we have shown three protocols that allow the user to retain her privacy with respect to this third party when signing messages with conditions on untrusted terminals.

In our future work we intend to examine if a user is able to identify the untrusted terminals that mount attacks against her, and in what extent she is able to prove the attack to third party.

References

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

Anonymizer Inc. (1999). . `http://www.anonymizer.com`.

Asokan, N., Debar, Hervé, Steiner, Michael, and Waidner, Michael (1999). Authenticating Public Terminals. Computer Networks, 1999.

Berta, I. Zs. and Vajda, I. (2003). Documents from Malicious Terminals. SPIE Microtechnologies for the New Millenium 2003, Bioengineered and Bioinspired Systems, Maspalomas, Spain.

Berta, István Zsolt, Buttyán, Levente, and Vajda, István (2004). Mitigating the Untrusted Terminal Problem Using Conditional Signatures. Proceedings of International Conference on Information Technology ITCC 2004, IEEE, 2004, IEEE, Las Vegas, NV, USA, April.

Brands, S. A. (1994). Untraceable off-line cash in wallets with observers. In Crypto'93 Springer-Verlag, LNCS 773 pp. 302-318.

Chaum, David (1981). Untraceable electronic mail, return addresses and digital pseudonyms. Communications of the ACM, v24, n.2 pp.84-88.

Chaum, David (1982). Blind signatures for untraceable payments. Advances in Proceedings of Crypto 82, D. Chaum, R.L. Rivest, & A.T. Sherman (Eds.), Plenum, pp. 199-203.

Clarke, Dwaine, Gassend, Blaise, Kotwal, Thomas, Burnside, Matt, Dijk, Marten van, Devadas, Srinivas, and Rivest, Ronald (2002). The Untrusted Computer Problem and Camera-Based Authentication.

Franklin, M. and Yung, M. (1992). Towards provably secure effi cient electronic cash. Columbia Univ. Dept. of CS TR CSUCS-018-92.

Jakobsson, M. and Raïhi, D. (1998). Mix-based electronic payments. Fifth Annual Workshop on Selected Areas in Cryptography (SAC'98), Queen's University, Kingston, Ontario, Canada.

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

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

Naor, Moni (1991). Bit Commitment Using Pseudo-Randomness. Journal of Cryptology: the journal of the International Association for Cryptologic Research, volume 2, pp 151-158.

Naor, Moni and Pinkas, Benny (1997). Visual Authentication and Identifi cation. Lecture Notes in Computer Science, volume 1294.

Naor, Moni and Shamir, Adi (1995). Visual Cryptography. Lecture Notes in Computer Science, vol 950, pp 1–12, 1995, `http://citeseer.nj.nec.com/naor95visual.html`.

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

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

Schneier, B. and Shostack, A. (1999). 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.

Schneier, Bruce (1996). Applied Cryptography. John Wiley & Sons, ISBN: 0471117099.

Schneier, Bruce (1999). The Solitaire Encryption Algorithm. http://www.counterpane.com/solitaire.htm.

Stabell-Kulo, Tage, Arild, Ronny, and Myrvang, Per Harald (1999). 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.

Syverson, Paul, F., Goldschlag, David M., and Reed, Michael G. (1997). Anonymous Connections and Onion Routing. IEEE Symposium on Security and Privacy, Oakland, California.