Documents from malicious terminals

István Zsolt Berta and István Vajda

Laboratory of Cryptography and Systems Security,
Budapest University of Technology and Economics, Budapest, Hungary

ABSTRACT

The user wishes to communicate with a remote partner over an insecure network using a terminal. Cryptographic algorithms running on the terminal may provide authenticity for the user's messages. In this paper the problem of sending authentic messages from untrusted terminals is analyzed. If attackers are able to gain total control over the terminal, the user must consider the terminal a potential attacker. Smart cards are often considered the ultimate tool for this problem. However, their lack of user interface enables man-in-the middle attack from the terminal. The user is a human with limited memory and computational power, only exceptional users can authenticate messages without a trusted device. Several biometric media encapsulate the content of the message and the identity of the sender, such as speech, video and handwriting. The authors suggest, that such media are far more difficult to counterfeit than plaintext, so users should rely on their biometric resources. In the protocol proposed by the authors, the user sends messages in a biometric format, strengthened by simple algorithmic authenticators. The smart card functions as a secure time gate ensuring, that the attacker has extremely little time to counterfeit both the biometric and the algorithmic protection on the message. The authors claim, that with the proper calibration of the biometric method and the time gate of the smart card, their protocol is strong enough for practical use.

Keywords: authentication, smart card, malicious terminal, biometry

1. INTRODUCTION

Sending authentic messages to remote partners has gained great importance with the spreading of digital signatures and electronic commerce. Smart cards are tamper-resistant personal microcomputers with great cryptographic potential. These devices are often considered the perfect tool for generating, storing and using cryptographic keys. Unfortunately, smart cards lack user interface, and they are only able to communicate with the user via terminals. If the terminal is malicious, various new attack possibilities arise. In this paper the problem of untrusted terminals is examined, and a solution is proposed to block the terminal's man-in-the middle attacks.

2. THE PROBLEM

2.1. Authentication

The user U wishes to send the m message to the remote partner R. In order to gain authenticity, U transforms m to $f(m)$, and sends the $f(m)$ message on the channel. If R is able to invert the received message using the f^{-1}, then accepts the $m = f^{-1}(f(m))$ as a valid one. Unfortunately, the attacker has full control over the channel and is able to observe, intercept and replace packages, and is also able to send messages in the name of U. This is why the f transformation must have a secret parameter k, that is unknown to the attacker. An attack takes place, if the attacker sends X to R in the name of U. The attack is successful if R accepts $m' = f^{-1}(X)$ as a valid message originating from U, and $m' \neq m$.

R may consider the m message *authentic*, if R accepts that m was sent by U (*partner identification*) and m arrived unmodified (*integrity*). If each m message contains a fresh element (such as a timestamp), the attacker is not able to replay previously recorded messages without having to modify it. Thus is the protection against replay attacks a question of integrity. While in case of encryption, perfect secrecy[1] exists, in case of authenticity the problem has no theoretically perfect solution. It is impossible to guarantee, that messages from the attacker are never accepted by the receiver as valid messages from the user, so only probabilistic solutions exist.[2]

Further author information:

I. Zs. Berta: Email: istvan.berta@crysys.hit.bme.hu, Telephone: +36302483630, Address: Department of Telecommunications, Budapest University of Technology and Economics, Po. Box 91, Budapest, Hungary

I. Vajda: Email: vajda@hit.bme.hu, Telephone: +3614631603 Address: Department of Telecommunications, Budapest University of Technology and Economics, Po. Box 91, Budapest, Hungary

2.2. Untrusted terminals

In order to communicate with the remote partner R, the user U needs the terminal T to communicate on the network. Since sensitive information is transmitted, various cryptographic protocols (e.g.: MAC, digital signature[3]) are used to guarantee the authenticity of the messages. Since U has little computational power, these protocols are performed by T. However, in many cases T cannot be considered secure. Sometimes, the party operating T is not trusted by U. Certain software or hardware componens of T might have hidden features. Moreover, even if the intentions of the party operating T are clear, the integrity of the terminal often cannot be guaranteed.

To provide security, another party is introduced. C is a smart card possessed by U, responsible for managing the k secret key required for the used cryptographic protocol. Today smart cards have cryptographic hardware that enables the keys to spend their entire lifecycle on C. They are generated, stored, used and finally destroyed on the card. Since the key never leaves C, T is not able to intercept it, so an authentic message can only be sent if C is present.

2.3. Assumptions

1. Limitations of resources

   (a) U is a human being, who can perform at most 10 additions (between bytes) in a minute, and can memorize at most random 10 characters (10 bytes). Although this latter may seem little, it is an optimistic approach. The information memorized by U should not be regular words, (because they can be guessed by the attacker) cannot be written anywhere, and should be changed regularly.
   We also assume, that U does not possess any trusted computational device. (If U can use such a device, there is no need to use a smart card.)

   (b) R is (or has access to) a trusted computer. R can perform one billion (10^9) additions in a second, and has many gigabytes of memory.

   (c) C is a smart card. The computational power of cards with crypto-coprocessor is in a similar magnitude to R, while their memory is measured only in kilobytes (e.g. 64k).

2. U, R and C may only communicate through T, they are not connected to each-other directly.

3. The parties U, R and C are not malicious, and their integrity can be guaranteed.

4. The attacker can take full control of T. *

2.4. Models

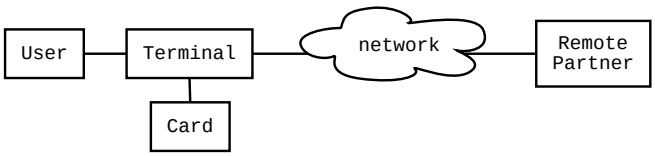

Figure 1. The commonly used model

The most common[4, 5] model of a smart card system has four parties (figure 1). The user U, the user's smart card C, the terminal T, and the remote partner R. T and R are connected to each-other by an insecure network. However, T is insecure, because the attacker has the chance to take full control of it. Moreover, T is a part of all communicational

*Note, that the situation is often not so dark. For example, if the terminal is the user's home machine, U might have have some control over T. Moreover, U might have additional more than realistic assumptions about T. For example U may assume, that no extra processors have been inserted into his machine, so the computational power of T can be approximated. U might also have the possibility to prevent T from cooperating with a supercomputer through the network (and thus summon arbitary computational power) if U can control when T is online. By buying a cheaper computer and a slower network connection, U can limit the possibilities of the attacker. However, in this paper U will suppose that T is under the total control of the attacker.

channels between the other parties. This is why the model introduced by Stabell-Kulo et al.[6] does not consider T an independent party, but considers it a part of the insecure communication system (figure 2).

This latter model describes systems with untrusted terminals better, because T is the only party in the system, that does not and should not generate information. This model also shows that the 'logical distance' between the three other parties is equal: the same medium, T, separates them. To bridge this 'logical distance' C and R can rely on various communicational protocols to ensure the integrity or even confidentiality of their communication.[7] Unfortunately, these protocols cannot be used in case of U, because of U's limited resources. Smart card technology has received severe criticism by the literature (from e.g. Schneier and Shostack[5]) that called smart cards 'handicapped' microcomputers because their lack of user interface spoils security.

It must be noted, that the authentic communication between U and C and between U and R is the same problem: authentic communication between parties with unbalanced cryptographic capabilities on an insecure channel. Thus, if a good solution for the former problem could be given, the same solution could be used in the latter case too, and there would be no need for using a smart card. This paper examines the theoretical possibilities for solving the above problem, and proposes a solution for practical use.

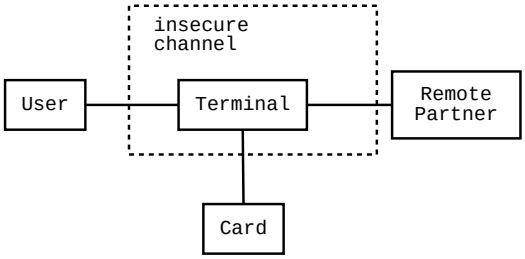

Figure 2. Model for untrusted terminals

3. TRADITIONAL SOLUTION

3.1. The protocol

The traditional solution to the problem described in 2 is the following:

1. U creates the m message.
2. $U \rightarrow C$: The user sends the m message to the card. (More precisely, it sends its hash code only.)
3. C: The card computes $sign_k\{m\}$, the signature of the message with U's private key, k.
4. $C \rightarrow U$: The card sends the signature ($sign_k\{m\}$) to the user.
5. $U \rightarrow R$: The user sends the signed message ($m, sign_k\{m\}$) to the remote partner.
6. R: The remote partner assumes the message to be authentic only if the signature is correct.

3.2. Attack

Since all communication between U and C is performed unprotected through T, in step 2 T can send its own message m' to C, and gain a valid $sign_k\{m'\}$ signature for it. T can send $sign_k\{m'\}$ to U (the user shall not be able to verify the signature without cryptographic capabilities) or may decide to obtain a valid $sign_k\{m\}$ too from C. In this latter case U shall not even have the chance to recognize the fraud even with more computational power. Later, in step 5 T will send its own message $m', S_k\{m'\}$ to R, who will find the signature correct, and would assume, that m' was sent by U. The attacker can be successful, because there is no protection against a man-in-the-middle attack in the system.

3.3. Attempts to block MIM attacks

- Today it is common that smart cards are protected by *PIN code*. While this may be useful if the card is stolen, it does not help against untrusted terminals. Since the PIN passes through T, it may intercept it. The idea of *one-time PIN code* already guarantees that U has to be present when C is used. However, U may still not know, if in step 2 T transmits the same message to C that it received from U.

- It may happen, that the *user has to confirm* whenever the terminal turns to the card. Even if this can be implemented (assumption 4), T can still perform a man-in-the-middle attack.

- A similar idea is, that the *user should remove the card if not using it*. Unfortunately, this does not prevent the man-in-the-middle attack either.
  However, in its own limitations, this solution is very powerful. By disconnecting the card from the terminal, the user can guarantee, that no-one tampers with it.[8] state, that smart cards thus transfer the sense of security from the electronic world to the physical one.

- The area on the disk of the *terminal* that turns to the card can be made *read-only* (e.g. used from a CD ROM). In this case the attacker may create a tampered copy of this area and modify the references to the original area. To prevent this, the whole binary area of the terminal needs to be made read-only. Although, this is a workable solution, it is most uncomfortable. It contradicts customization and prevents the terminal manufacturer from correcting (or patching) its products. We do not deal with this option, because it contradicts our assumption (#4), that the attacker may take full control of T.

- *Biometric solutions of user identification* exist and may provide additional security. Even the most closely integrated solutions (where the finger print reader is on the card itself) do not prevent man-in-the-middle attacks.

- The user may ask help from a *trusted third party* to verify that the correct message was signed.[6] However, if U can only communicate with this third party through T, T may intercept and modify these messages too. In this case the same problem arises: authentic communication should be provided on an insecure channel not between U and C, but between U and the third party.

- Using *more developed smart cards* could improve security. E.g. Abadi et al[9] prove that a smart card with a display device would enable secure two-way communication between U and C. The approach of Clarke et al[10] also consume singnificantly more resources. Unfortunately, such solutins would massively increase costs. This is why no such cards exist yet.

- Another good solution would be to *protect the integrity of messages between U and C* with e.g. a MAC.[6] Unfortunately, this is not possible due to the limited resources of U. This topic is further explained in the next section.

4. LIMITED POSSIBILITIES OF THE USER

As it is described in section 2.3, the resources of the user are limited. U has a small amount of memory and little computational power. Moreover, U also makes mistakes during calculations, so U can be considered a 'poor cryptographic device'.

4.1. Dead ends

4.1.1. Signature on a hash value

The user U cannot carry out complex computations on his own, which are needed for calculation of a hash value, a cryptographic checksum. If we consider a general transformation from binary N bit input to binary M bit output, such that in average at least half of the inputs has effect on each output, then in average at least $N * M/2$ binary operations has to be performed. For instance, if $N = 1000$, $M = 64$, it means at least 32000 operations, far out of the capacity of user U taking into account the time restrictions and the reliability problems of U. (The needed number of calculations remains too high even in case if we assume character level operations, say with decimal characters). Easy linear transformations might be handled by U (like CRC), but those are inappropriate in making cryptographic checksums. In this case T – knowing the m message and the weak transformation – would be able to prepare an m' message with the same hash value easily.

4.1.2. Encryption of the whole message

Performing a strong *block-cipher* on the whole message is also beyond the capabilities of U. If we consider M-bit-long blocks, N/M transformations has to be performed. In order to make a transformation strong, it needs to meet the criteria of completeness and that of avalanche effect. Thus, U needs to perform at least $M*M$ operations for each block, which means altogether $N * M$ operations. Another option could be the one proposed by Stabell-Kulo et al[6]: using a *one-time-pad* in cascade with an additional nonlinear operation, like a character level substitution table, which can also be easily performed by the user. However, U does not have enough storage capacity to perform a one-time-pad on the whole message.

4.2. Pencil-and-paper solutions

4.2.1. Book cipher

The *book cipher*[11,12] is one of the oldest strong ciphers. It transforms characters of the message m into pointers to characters in a given book B.

$BookCipher(m[i]) : m[i] \mapsto j$, where $B[j] = m[i]$. The key in a book cipher is the book B.

However, if the attacker has information on m, the book cipher alone does not provide authenticity, because the attacker is able to e.g. swap characters in the message. The book cipher should be strengthened by the structure of the document to provide authenticity.

4.2.2. One-time-pad

The *one-time-pad*[1] is the only known algorithm, that provides perfect secrecy, and it its also simple enough for U to perform. $OTP(m[i]) = m[i] \oplus k[i]$, where k is a one-time random key, and $length(k) = length(m)$. However, the modular addition ($\oplus$) is a linear operation, so messages encrypted with one-time-pad are subject to linear manipulations.[6] propose that the one-time-pad can be used for authenticity if it is strengthened by a secret substitution table t. U has to memorize t and k as secret keys. The main drawback of this method is, that k can be very long, so U can only use it if he can use significantly large secure storage.

4.2.3. Solitaire

Solitaire[13] is merely an imperfect one-time-pad, where the 'one-time key' is the output of a pseudo-random number generator. Solitaire uses a deck of cards to generate random. The key is the seed of the PRNG, which is an ordering of the deck. The key is $log_2 56! \approx 248$ bits long, that is already far beyond the memory capacity of the user U. Solitaire provides strong secrecy but weak authenticity, just like one-time-pad, so it should be strengthened e.g. by a substitution table, that will add $log_2 26! \approx 88$ bits to the key length in case of a 26-character-long alphabet. The PRNG of Solitaire is very slow for commercial use. Schneier proposed it for secret agents.

4.2.4. Visual methods

Visual cryptography was introduced by Shamir and Naor.[14] It enables the user to recieve encrypted messages from the remote partner. This method relies on a one-time-pad, where the $\oplus$ operation is accelerated by the user's eye via transparencies placed on the terminal's screen. Naturally, the key of the one-time-pad is a transparency, that can be used one-time only. Naor and Pinkas[15] extended visual cryptography to support user identification and recieving authentic messages. However, each visual cryptography method (except for partner identification) is unidirectional: they only work from the remote partner towards the user. Matsumoto[16,17] has also developed a user identification method that has visual accelerations, but does not protect the session itself.

4.3. Summary of algorithmic possibilities

In order to give a perfect solution for the problem of insecure terminals, an authentication method would be needed that gives similar functionality to public key cryptosystems (RSA or ECC). However, in order to be used by 'weak computers' like humans, this algorithm should have a shorter key and should require dramatically less computational power. If an algorithm like this would exist, then it would be used in computer-environment too.

Although U possesses abilities like good quality random number generation, memorization of passphrases and 'common sense', finding a purely algorithmic solution for the problem of insecure terminals seems hopeless. In order to solve the above problem, non-algorithmic abilities of U should be exploited.

5. PROPOSED SOLUTION

5.1. Extensions to the model

In section 2.3 we modeled U, R and C as three independent computational devices wishing to communicate with each-other, where U has limited resources that prevent the application of advanced cryptography. Our model can be extended here by not considering the user U merely a slow computer, but a human being with the following abilities:

- U can be identified by means of *biometry*. Moreover, U can identify other humans known to U using biometry.
- We may assume, that U has a timer independent from T (e.g. a watch), so U is able to measure time.
- By removing the card U is able to block the channel between C and T.

5.2. Biometric signatures

Most biometric features only enable the identification of humans, but some can also be used to transmit information. For example, fingerprint verification can only identify people, but the recorded voice combines the identity of the individual with the content of the speech. Out of the known biometric methods speaker verification, audiovisual verification and handwriting verification can fulfill such criteria.

In this paper the concept of time is used (with the assumption, that manipulating a biometric message requires a certain amount of time or even user interaction) to reinforce biometric authenticity. Since handwriting is less connected to time, in this paper the two others shall be used as 'biometric methods'.

5.2.1. Security

The user U transforms the message m to a biometric format to ensure its authenticity. The $bio(m)$ message is in a format that carries the biometric features of the sender too. We suppose, it is 'safer' than the plaintext message m. In this section the extent of this 'safety' is analyzed.

1. *Unconditional security* would mean, that $bio(m)$ messages cannot be counterfeited or manipulated.
2. *Practical safety*: This means that the attacker needs considerably more time to counterfeit $bio(m)$, than to counterfeit m itself.
3. *No significant security*: The other extreme case would be to consider $bio(m)$ messages as secure as m (plaintext messages), so attackers do not need considerably more time to manipulate them.

Unconditional security is not a realistic assumption. We assume, that if the attacker has enough time and resources, it can manipulate any $bio(m)$, so considering them perfectly secure, is out of the question. We also assume, that it is possible to calibrate a biometric method (bio) to give practical safety.

5.2.2. 'Key space'

A biometrically signed message – just like a cryptographically signed one – is a result of a transformation based on some secret k. While in case of cryptography, k is a set of bits, in case of biometric signatures $k_{U_{bio}}$ is more complex: it is a combination of the biometrical characteristics of U. However, these can be affected by the following factors:

- Biometric identity: time-invariant biometric characteristics of U
- Current condition: Ever-changing factors, that cannot be pre-planned in any way. E.g.: mood, illnesses, fatigue, aging, etc.
- Conscious alterations: Various attempts of U to influence the current sample.

Although the above represent an infinitely large key space, only factors known to the R can be used as secret key. Moreover, the above factors have to be measured, and the precision of measurement limits the size of the key too. Biometric systems are constructed on the assumption, that the measured *biometric identity of each individual is different*. This means, that the key space of biometric identities is considered extremely large, even if their value cannot be measured exactly.

5.2.3. Attacks

If the attacker has observed the previous $bio(m_1)$, $bio(m_2)$, ..., $bio(m_n)$, and wishes to prepare a counterfeited $bio(m')$ message, two kind of attacks are possible:

1. The attacker may observe the biometrical characteristics of U and try to extract k_{bio} from the recorded messages in order to synthesize a complete $bio(m')$. In this case the attacker tries to masquerade itself as U, and trick the *partner identification* method of R. Fortunately, this attack is far beyond the capabilities of most attackers.
2. Not being able to synthesize a $bio(m')$ message, the attacker may choose to create it using cut-and-paste from one or more previous $bio(m_i)$ messages sent by U. Thus, the attacker violates the *integrity* of the message, while every part of $bio(m')$ originates from U.

5.2.4. Integrity

The main weakness of $bio(m)$ messages is that the bio transformation does not have the *completeness* property. This means, that modification of one part of $bio(m)$ does not require modifications in any other parts. What is the smallest part an attacker may try to modify? Since $bio(m)$ does not consist of blocks, the attacker may try to manipulate segments of any size. However, since the aim of the attacker (section 2.1) is more than successfully modifying $bio(m)$, but to make R accept X where $bio^{-1}(X) = m' \neq m$. If m consists of characters, no attack on $bio(m)$ can be called successful, that does not modify at least one character in m. Although, the bio transformation does not satisfy the criteria of completeness, biometric messages have an internal structure, that chains segments of $bio(m)$ together. On an audio message this can be the changes in amplitude of the voice or the tone of the speaker. On a video message this can be the changes in the facial expressions or the position, behavior or even clothes of the person.

While checking these properties using algorithms could be very awkward, the human brain triggers for any sudden or unpredicted changes in the biometrical characteristics of the speaker. The task of the attacker can be made especially hard if the biometric method is calibrated using other artificial methods to chain blocks of the $bio(m)$ together. Good examples for the audio channel could be a different systematic background noise or music under the message. On the video channel a clock or television in the background can be used to harden cut-and-paste attacks. The method to preserve the structure of the message need not be secret: U can announce the method at the beginning of $bio(m)$, since the method only has to be one-time and consistent in the whole $bio(m)$. If the attacker wishes to paste together segments from $bio(m_i)$ and $bio(m_j)$, the manipulation of the biometric structure of the messages is needed. Although, this is not impossible, it requires a significant amount of resources and human interaction too.

We assume, that the attacker needs significantly more time, to prepare such an attack, than U needs to create $bio(m)$.

5.2.5. Partner identification

In case of plaintext messages m contains no algorithmic protection, an attacker is able to send a valid m' message in the name of any user U without any information on the person. Viruses, that spread by email in the name of the user use this simple principle. However, every part of $bio(m)$ itself carries the identity of U. Since the key space k_{bio} is large, no attacker has realistic chances of creating a valid $bio(m')$ message without having any a priori information on U. Thus, an attacker has to *plan the attack against each specific user U*. Messages of U has to be observed and recorded in order to synthesize a valid $bio(m')$ message.

Biometric messages are very hard to counterfeit. However, if we suppose, that the attacker is able to extract k_{bio} from $bio(m_1)$, $bio(m_2)$, ..., $bio(m_n)$, and is able to synthesize a $bio(m')$ message using k_{bio}, $bio(m)$ does not provide proper partner identification. On one hand, neither of the above is true for most attackers, on the other hand, since the integrity of messages can be guaranteed, even the simple algorithmic protection (section 5.7) is able to prevent attacks.

5.3. Our protocol

We consider it a critical error in the traditional approach, that the smart card signs the transmitted message on behalf of U, while the card cannot make sure that the message truly originates from the user. In our solution m is 'signed' by U biometrically. The biometric 'signature' of $bio(m)$ certifies that m comes from the person U. C contains the private key k_c, and signs the message with its own (C's) signature. C also puts a header containing timestamps on the message before signing it. (The ways C can use to obtain secure timestamps are described in section 5.6.) C's signature only states that C was present at the given time, and that the message has not been altered from the time of signing.

1. $U \rightarrow C$: U inserts the card, C obtains the t_{start} timestamp, U removes the card.
2. U creates the $M = (t_{beginning}, bio(f_k(m)), t_{end})$ message in one of the above biometric formats.
3. $U \rightarrow C$: U inserts the card and sends a hash of M to C, C obtains the t_{sign} timestamp.
4. C calculates the $sign_{k_C}\{I_U, V_U, t_{start}, t_{sign}, k, M\}$ signature.
5. $C \rightarrow U$: $I_U, V_U, t_{start}, t_{sign}, k, sign_{k_C}\{I_U, V_U, t_{start}, t_{sign}, k, M\}$
6. $U \rightarrow R$: $I_U, V_U, t_{start}, t_{sign}, k, M, sign\{I_U, V_U, t_{start}, t_{sign}, k, M\}$
7. R verifies the message using biometric verification and by checking the digital signature and the timestamps. The verification is detailed in the next section.

5.4. Details of our protocol

1. $U \rightarrow C$: Before the creation of the biometric message, U notifies C by inserting the card. U must remove C after it has acquired a secure timestamp (t_{start}). C stores the t_{start} value. In the moment, when U removes the card from the terminal, begins the recording of the biometric message.

   The notification, and the fixed t_{start} timestamp ensure, that T cannot prepare a complete m' message before starting the recording, because the $t_{beginning}$ biometric timestamp should match the t_{start} timestamp, which is not known before the recording.

   The removal of the card is necessary, because otherwise T may delay U's notification to C, so C would obtain a t_{start} timestamp chosen by T. This would give T more time to prepare a manipulated message. Thus, if the card does not respond to the user, it should be removed after a short time (e.g. 30 seconds), to prevent this kind of attack. We emphasize the significance of card removal in general too, since it is a robust method for the user to prevent the terminal from accessing the card. However, the user still cannot know how many T-C transactions were performed while the card was in the reader.

2. Since the integrity of a plaintext message cannot be guaranteed, U has to prepare m in a biometric format. Thus, U prepares the message M, where $M = (bio(t_{beginning}, f_k(m), t_{end})$. Biometry ensures integrity, and f is a simple transformation (section 5.7) that protects against attackers who can synthesize a counterfeit $bio(m')$. (section 5.2) The transformation f has a one-time secret parameter k, that is a secret key between U and C. (see 5.7)

   The timestamps $t_{beginning}$ and t_{end} should be produced by U's own timer (e.g. watch) at the beginning and at the end of the recording of $bio(f_k(m))$. Since the message is being created on T, the terminal may try to manipulate at this point.

3. $U \rightarrow C$: U inserts the card, and sends the M message to it. (More precisely, it sends the hash value only, where the hash is calculated by the terminal.) This is the last point, where T may tamper with the message. After this, the digital signature will be computed behind the hardware firewall of C, and T will not be able to manipulate the signed M. U inserts C into T only for a short time again. If the signature (step 5) does not arrive in e.g. 30 seconds, there is a high chance of an attack similar to the one described in step 1, so the card should be removed. T must forward the message to C now, since it cannot do it when U inserts C next time, because then the t_{end} biometric timestamp in M would be considerably different from the t_{sign} timestamp of the signing. After the hash code of M has arrived, C obtains another secure timestamp t_{sign}.

4. C adds a header including t_{start} and t_{sign} to the message and signs both the message and the header with it's own private key k_c. From this point further manipulation of the message without k_c impossible.
   The header C adds should contain the following:

   I_U — The name of the user U and the public key of C (with a certificate).

   V_U — Information required for the biometrical identification of U. I_U and V_U make this protocol 'public key', so that it can work without U and R having to agree on U's biometric features using a secure channel.

   t_{start} — The time, when the recording of the biometric message was started. This is acquired in step 1.

t_{sign} The time, when the recording of the biometric message was finished, and the message was signed. This timestamp is acquired in this step.

k The secret parameter of f in step 2. (section 5.7)

Thus the signature computed by C becomes: $sign_{k_c}\{I_U, V_U, t_{start}, t_{sign}, k, M\}$

5. $C \rightarrow U$: $I_U, V_U, t_{start}, t_{sign}, k$, $sign_{k_c}\{I_U, V_U, t_{start}, t_{sign}, k, M\}$
6. $U \rightarrow R$: U removes the card from the terminal and forwards the above message to R together with M.
7. R has to verify the following:
   (a) If $m = bio^{-1}(f_k^{-1}(m))$, the message has been tampered with.
   (b) If $m_{bio} = (t_{beginning}, m, t_{end})$, is it true, that $t_{end} - t_{beginning} = length(bio(m))$? This forces the attacker, to prepare a m' manipulated message where $length(bio(m)) = length(bio(m'))$.
   (c) Is it true, that $t_{start} < t_{beginning} < t_{end} < t_{sign}$? This checks for causality. Any naturally and correctly created message must have this feature. If not, than the message has obviously been tampered with.
   (d) Is $t_{beginning} - t_{start} < t_{safety1}$ and is $t_{sign} - t_{end} < t_{safety2}$, where $t_{safety1}$ and $t_{safety2}$ are system parameters that should be scaled properly. Both of them are relatively small, approximately a few seconds. Their impact is critical on the security of the system, their role is detailed in section 6.1.
   While t_{start} and t_{sign} are securely acquired by the card, $t_{beginning}$ and t_{end} are acquired by the user and transferred to C with no other protection than the biometric features of U. So, while the attacker cannot modify t_{start} and t_{sign}, there is a chance, that $t_{beginning}$ and t_{end} have been tampered with. In this point, R checks if $t_{beginning}$ and t_{end} are close enough to t_{start} and t_{sign}.
   (e) If the message came from U, was it correctly signed by U's card, C? (This is the normal PKI problem, that has to be solved in the traditional approach.)
   The digital signature certifies, that C was present, and the message has not been modified from the time it was signed. The I_U parameter also serves as a secure way to identify U. However, the digital signature of C does not certify integrity.
   (f) Was the message created by the person, whose characteristics are described by V_U?
   Since the V_U parameter originates from C, it can be assumed to be authentic. This enables, that U and R do not have to agree on U's biometric features before the transaction using a secure channel.
   (g) Was the m message signed by the f_k transformation correctly? (see 5.7)
   (h) Is the structure of the biometric message consistent or has it been tampered with? (E.g. Do the lips of the person on the video move, when the speech is heard? Is the background music consistent?)
   (i) Was the theft of C not reported by U?

If all the above have been ensured, R knows, that:

- The message was created by U.
- The message was signed by U's card.
- The message was created between t_{start} and t_{sign}.
- The attacker had extremely little (only $t_{safety2}$) time to alter the message.

5.5. The function of the card

In the above protocol the card does not certify the users identity, and does not prove the authenticity of the message either. It has only one function: the card is a *secure time gate*. In section 5.2 we assumed that attackers are able to manipulate biometric messages if they have enough time. The card limits the attacker's time for this manipulation. Moreover messages can only start when the time gate opens and can only finish when the gate closes.

5.6. How can C obtain a secure timestamp?

Today's smart cards do not have a timer. However, certain secure computational devices, such as iButtons ([18]), do have a built-in secure clock. These devices can safely use their own timer to obtain a secure timestamp. Other cards need to get a secure time from a 'secure time server' (STS) with the following protocol:

1. $C \rightarrow STS$: n, where n is a nonce value.
2. $STS \rightarrow C$: $t, sign_{k_{STS}}\{n, t\}$

The purpose of this protocol is to obtain ($t_{request}$), the time of step 1. However, communication with STS takes time, so STS will return t instead of $t_{request}$. It also takes time for the message in step 2 to return to C, so C will receive t only at t_{answer}. Note, that C can only communicate with the STS through T. While T is not able to modify the timestamp, it may delay any message. Thus, C can only be sure, that $t_{answer} \geq t \geq t_{request}$. Alas, C – not having a timer – cannot know the exact $t_{request}$. If T delays step 1, $t_{request} \ll t$ occurs, and the terminal has more time to forge a $bio(m')$ message. If T delays step 2, $t \ll t_{answer}$ would occur, but T would gain no advantage of this.

However, in the above protocol, not only steps 1 and 2 can be delayed. Since C is always a slave in the T-C communication,[8] step 1 is also initiated by the terminal. We shall call this initiation step 0 that takes place after the user inserted the card at t_U. A delay in step 0 causes $t_U \ll t_{request}$, which also gives T more time to forge a $bio(m')$ message. It is trivial, that no protocol is totally invulnerable to delays of step 0 and step 1. The fact, that C has to *request* the secure time, induces, that this request can be delayed. The practical solution for this is timeout. However, C cannot timeout this operation, because C has no timer, so U has to. U must remove the card from the terminal after a given time unconditionally (as it is described in our protocol in section 5.3), and reset the above protocol. Naturally, connecting to a STS takes time, and requires higher t_{safety} parameters (we suppose, 30 seconds would be enough for the entire process). In the protocol described in section 5.3, devices with their own timer are more secure.

5.7. Algorithmic protection (f_k)

In section 5.2 we have shown, that biometric messages can provide authenticity, if R is able to make sure, that the message originates from U. To ensure this, we can exploit U's computational capabilities even if they are far from being excellent. To provide algorithmic protection, a secret key k is needed. U must know k to use it for identification, and R must know it to check U's identity. We wish to avoid, that U and R has to agree on k, so in our protocol the key is known to U and C (k can be issued to U together with C just like C's PIN code). C is able to send k to R on a secure channel if needed.

In our solution the attacker has a very short time to counterfeit m. Moreover, the results of the f function has to be extracted from $bio(f_k(m))$, and performing the bio^{-1} transformation is difficult for T. Thus, we suggest, that $f_k(m)$ should mean, that a certain amount of random characters should be inserted into m at locations determined by k. Naturally, a human can extract the k key from $bio(f_k(m))$ if enough time is given, so k must be one-time. If m is 1000 characters long, and a known character is inserted into three different locations, there are $\begin{pmatrix} 1000 \\ 3 \end{pmatrix} = 10^8$ different possibilities, so if the attacker synthesizes an m', it has a $1 : 10^8$ chance of being accepted by R as a valid message. (In this number, the protection provided by biometry, and the secure time gate of the smart card are not included.) To gain this security, U has to memorize three numbers between 1 and 1000. If the message is short, solutions provided by Stabell-Kulo et al[6] can be used too.

6. ANALYSIS

6.1. The importance of $t_{safety1}$ and $t_{safety2}$

After the message reached the card the attacker does not have the possibility to manipulate it without having to crack the digital signature, so the message can only be modified before it is sent to the card. Since the card puts the t_{start} and t_{sign} timestamps onto the message right before signing it, these two cannot be modified by the attacker. The attacker does not have the chance to obtain additional – considerably different – timestamps, because the card is only inserted into the terminal at t_{start} and at t_{sign}. Moreover, the exact value of t_{start} is not known before U starts recording the message, and t_{sign} is not known before U finishes the message. If the attacker wishes to prepare $bio(m')$, it has to prepare

the $bio(t'_{beginning})$ and $bio(t'_{end})$ manipulated biometric timestamps that carry the biometric characteristics of U and also satisfy the $t'_{beginning} - t_{start} < t_{safety1}$ and $t_{sign} - t'_{end} < t_{safety2}$ criteria.

The attacker may have the following strategies:

1. The attacker may try to guess t_{start} and t_{sign}, and create $bio(m')$ messages with proper $bio(t'_{beginning})$ and $bio(t'_{end})$ satisfying the above criteria. If $t_{safety1}$ and $t_{safety2}$ are small, there are but minor chances that both timestamps are correct.

2. The attacker may do the manipulations online. While U is recording the $bio(m)$ message, the attacker may manipulate it on T. This requires the attacker to load all the tools for the precise manipulation of the biometric message onto T. Not all terminals are capable of this. Moreover, $bio(m')$ should have the same length as the $bio(m)$ message U is preparing, because the card's presence is needed for the signature, and the card is only present at t_{sign}. Since T has the time of the recording for manipulation, long messages are more vulnerable. However if the biometric message contains a checksum (even a very simple one) at the end, the attacker may only manipulate if the message is finished. This brings us to the next point.

3. The attacker may also try to manipulate the already finished message. That means, that after U finishes $bio(m)$, the attacker may intercept it, and modify certain parts before forwarding it to C. The attacker has only $t_{safety2}$ time for this, and if $t_{safety2}$ is small. According to section 5.2 this is almost impossible.

As it can be seen, the exact setting of the t_{safety} parameters is critical for the security of the whole system.

6.2. Chances of the attacker

In section 6.1 we have shown, that the manipulation of the finished message is almost impossible. So, the attacker can only have the chance to prepare $bio(m'_1)$, $bio(m'_2)$, ..., $bio(m'_n)$ with various $bio(t'_{begin})$ and $bio(t'_{end})$. When U sends m to C, the attacker checks if one of the $bio(m')$ matches the timestamps of U's $bio(m)$. If yes, the attacker sends the matching $bio(m'_i)$ message instead. In this section we shall analyze the chances of this. We suppose, that at least one message is sent in every hour with the maximum length of 10 minutes. We also suppose, that C does not have a secure timer, so network connection to STS is needed (section 5.6), so $t_{safety\,x} = 30\,sec.$ This means that the attacker should prepare 120 $bio(t'_{begin})$ timestamps. Since the maximum length is 10 minutes, for each beginning timestamp 20 ending $bio(t'_{end})$ timestamps should be prepared. Altogether this means 2400 $bio(m')$ messages. This can be increased further by using algorithmic protection like the one described in section 5.7.

6.3. Limits of our protocol

1. It enables U to send authentic messages to R. However, the U is still unable to recieve authentic messages. E.g. visual cryptographic methods[14, 15] could be used for recieving messages.

2. It does not provide encryption. According to Stabell-Kulo et al,[6] providing encryption is not possible in case of untrusted terminals.

3. It does not protect against DoS attacks. It is trivial, that in case of untrusted terminals, no protocol protects against DoS attacks.

6.4. Advantages of our protocol

1. Unlike the traditional protocol, it operates with untrusted terminals with great practical safety.

2. The proper manipulation of biometric messages is beyond the capabilities of most attackers.

3. The manipulation of biometric messages requires the attacker to prepare for each specific target.

4. The presence of the card is needed for the protocol.

5. Even if the card is stolen, the attacker needs to counterfeit the biometry of the user.

6. The protocol operates with cards, that exist today.
7. The protocol does not require the user to perform cryptographic operations.
8. The parties in the communication need not agree on parameters using an authentic channel.
9. The remote partner in the communication need not have large computational power. Even two humans may communicate this way.

7. CONCLUSION

Sending authentic messages using untrusted terminals is a practical problem, for which smart cards are often considered the ultimate tool. Unfortunately, their lack of user interface enables man-in-the-middle attacks of malicious terminals. In this paper various algorithmic and technological methods were considered to block the above attacks. Finally, the authors proposed a practically safe solution that relies on biometric features, and supports it with a smart card functioning as a secure time gate.

REFERENCES

1. C. E. Shannon, "Communication Theory of Secrecy Systems." Bell Systen Technicl Journal, vol 28, pp 656–715, 1949, 1949.
2. G. Simmons, "Authentication theory / coding theory." Advances in Cryptology – CRYPTO 84, Lecture Notes in Computer Science, No. 196, Berlin: Springer Verlag, 1985, pp 411-431, 1985.
3. B. Schneier, "Applied Cryptography." John Wiley & Sons, ISBN: 0471117099, 1996.
4. N. Asokan, H. Debar, M. Steiner, and M. Waidner, "Authenticating Public Terminals." Computer Networks, 1999, 1999.
5. 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.
6. 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.
7. T. Ylönen, T. Kivinen, M. Saarinen, T. Rinne, and S. Lehtinen, "SSH Protocol Architecture." Internet Engineering Task Force, Network Working Group, draft-ietf-secsh-architecture-07.txt, 1 2001. http://www.ietf.org/internet-drafts/draft-ietf-secsh-architecture-07.txt.
8. I. Berta and Z. Mann, "Smart Cards – Present and Future." Híradástechnika, Journal on C^5, 2000., vol 12, 2000.
9. 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.
10. 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.
11. D. Kahn, "The Codebreakers: The story of secret writing." Macmillan Publishing Company, New York, USA, 1967, 1967.
12. S. Singh, "The Code Book : The Science of Secrecy from Ancient Egypt to Quantum Cryptography." Anchor Books; ISBN: 0385495323, 2000.
13. B. Schneier, "The Solitaire Encryption Algorithm." http://www.counterpane.com/solitaire.htm, 1999.
14. 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.
15. M. Naor and B. Pinkas, "Visual Authentication and Identification." Lecture Notes in Computer Science, volume 1294, 1997.
16. T. Matsumoto, "Human-Computer cryptography: An attempt." In ACM Conference on Computer and Communications Security, pp 68-75, 1996.
17. T. Matsumoto, "Human identification through insecure channel." In Theory and Application of Cryptographic Techniques, pp 409-421, 1991.
18. Dallas Semiconductor, "iButton Home Page," 5 2001. http://www.ibutton.com/.