Tatra Mt. Math. Publ. **29** (2004), 1–16

**TATRA MOUNTAINS**
*Mathematical Publications*

# LIMITATIONS OF HUMANS WHEN USING MALICIOUS TERMINALS

István Zsolt Berta — István Vajda

ABSTRACT. A user wishes to communicate with a remote partner over an insecure network. Since the user is a human being, a terminal is needed to gain access to the network. In this paper the problem of sending authentic messages from insecure or untrusted terminals is analyzed. In this case attackers are able to gain total control over the terminal, so the user must consider the terminal a potential attacker.

The authors consider an important merit of the paper the construction of a formal model that is able to handle interesting problems in case of untrusted terminals. According to this model, the user is able to encrypt or authenticate messages with very small degree of security only, so these messages can be broken by the terminal with significant probability. Since the cryptographic abilities of the user are more than limited, and no solution is known for the problem, our model seems to be realistic.

We show that if the user lacks the ability to encrypt (and decrypt) messages in one step, i.e., without interaction with the remote partner, then the latter is unable to help the user in establishing a secret channel. We also show the same conclusion for authenticity: If the user is unable to calculate an authenticator that cannot be broken by the terminal, then the remote partner is unable to help the user in constructing an authenticated channel.

## 1. Introduction

We consider electronic commerce applications, where a user—a sole human being—wishes to make business with a remote partner. In such cases, the user transmits sensitive information via an insecure network. Such information needs protection, that is provided by cryptographic algorithms running on the user's terminal and the remote partner's computer. In such a situation it is essential that both of these computers are trusted.

Thus, in most cryptographic protocols it is assumed that a trusted terminal is present. However, in realistic scenarios, very few terminals can be called 'trusted'.

2000 Mathematics Subject Classification: 03D15, 94A60.
Keywords: untrusted terminal, human-computer cryptography, complexity theory.

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. It is also very hard to check if the hardware or software of the machine has been tampered with. Most users lack the skills to protect their own machine from trojans, viruses or intruders from the net, and they have little or no information on the software installed.

The user might have a smart card for assistance. Smart cards are often considered a magic bullet against untrusted terminals. However, since they do not have a direct user interface (keyboard or display) of their own, they need a terminal to communicate with the user. If this terminal is malicious, it can easily alter any message to and from the card.

The problem of untrusted terminals is a mass problem, that appears in almost every e-commerce application. The authors would like to contribute to the research of this important topic.

## 2. Related work

### 2.1 Terminal identification

Terminal identification is perhaps the most basic problem. It aims to authenticate terminals and to distinguish between terminals of various trust levels. In the most simple case, terminals are categorized into two main groups: 'good' terminals and 'evil' terminals. In the former group we wish to trust completely, while we wish to avoid the latter group. This approach is based on the assumption that while 'good' terminals are able to identify themselves correctly, 'evil' terminals are unable to do so. It is important to note that 'good' terminals have to be tamper resistant too, otherwise tampered legal terminals could serve the attacker, but would still be able to identify themselves.

Asokan *et al.* [3] and Rank and Effing [16] show a simple protocol that—using smart cards and one-time passwords—enables the identification of fake terminals. In their solution, a secret password is shared between the user and the smart card. The card presents the password to the terminal only if the terminal has identified itself correctly by a challenge and response method. The user accepts those terminals as 'good' ones that are able to present the password. This terminal identification protocol is simple, but requires the user to be very disciplined (to check and change the password at every single login) and to think very critically and suspiciously about the terminal.

Unfortunately, this protocol still does not provide protection terminal-in-the-middle attacks, so Asokan *et al.* propose that the terminal and the smart card should run distance bounding protocols [7] to limit the possibilities of terminal-in-the-middle attackers.

It is also very hard to identify tampered terminals. Thompson [21] demonstrates that it is impossible to verify if terminal has been tampered with, without a separate trusted machine that has direct access to the terminal's 'hard drive'. He argues that if the compiler program on the terminal is manipulated, than all software on the terminal (including the compiler itself) that is compiled with it can be malicious, even if the source code is verified to be correct. This leads to the fact that the terminal's integrity can neither be verified nor can be corrected using a smart card and the software on the terminal.

Anderson [2] studied several cases of electronic frauds and attacks on security systems, finding that most frauds were not performed by using crypto-analysis, but by exploiting blunders, implementation errors, management failures or insider knowledge. Many of the attacks discussed by Anderson were delivered by fake malicious terminals.

Another approach is to keep every terminal trusted by making it hard for attackers to tamper with them. Although this seems like fighting windmills, the industry supports this approach the most, since this could be the most comfortable for users. Several projects follow this direction, including TCPA, which would restrict a terminal to executing only digitally signed applications ([22], `http://www.trustedpc.org`, `http://www.finread.com`).

## 2.2 Using untrusted terminals

*2.2.1 Using smart cards on untrusted terminals.* In frequent cases we have no other option than using the terminal, while we know we cannot trust it. It is a wide-spread approach to use smart cards or other trusted personal devices to protect keys, and other sensitive data. However, due to their lack of user interface, smart cards can only communicate with the cardholder through the terminal only (Figure 1).

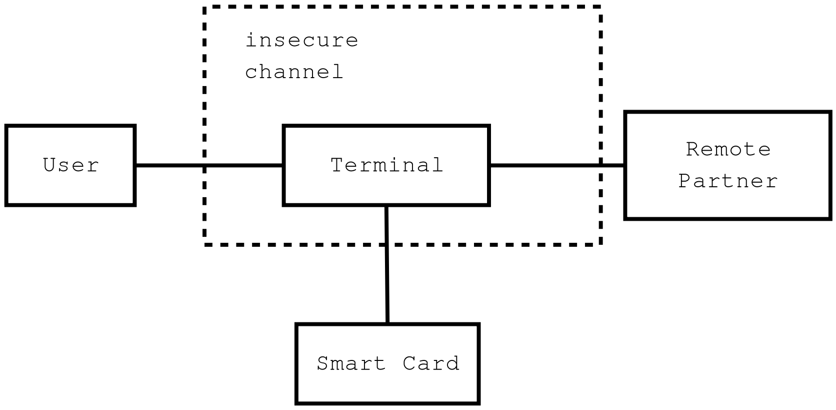


FIGURE 1. A widespread model for systems with untrusted terminals.

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 show that the problem can be solved with a smart card that has peripherals to communicate directly with the user, and they show secure protocols for such a device. Later on, they strip as much of these peripherals from the card, as possible. They prove that with the resulting card, which has no clock and no keyboard but only a display, the same degree of security can be implemented without placing too much load on the user. However, after more than 10 years of development, the smart card with a display is still not a feasible assumption.

Similarly, Gobioff *et al.* [9] analyze various hypothetic smart cards having secure input or output channels, and identify various classes of equivalence between them. For example, they show that a smart card with a private input channel (keyboard) is equivalent with one with a private output channel (display). Their contribution adds rather little to that of Abadi *et al.* Balfanz and Felten [4] also show that a trusted PDA with a trusted user interface could be more secure for generating digital signatures. Their work is an implementation that supports the principles of Abadi *et al.* by evidence, but does not extend them by any means. However, they also raise the question, if a PDA could be considered a trusted device. Moreover, a PDA is very expensive compared to a smart card, so organizations (like banks) are unlikely to equip their users with PDAs.

Stabell-Kulo *et al.* [20] proposed a protocol for sending authentic messages from untrusted terminals. In their solution, the user encrypts the message using a one-time-pad together with a monoalphabetic substitution table, and thus gains authenticity for structured messages. They use a smart card to sign the message and a trusted third party to sign it again and thus certify the card's signature. Unfortunately, in case of long messages the user is not able to memorize one-time keys, so the solution of Stabell-Kulo *et al.* works with short ones only. However, they state that obtaining secrecy in case of untrusted terminals is impossible, which contradicts their own assumption that the user can memorize a one-time key as long as the message.

Schneier and Schostack [18] analyze smart card systems as a whole, and call smart cards 'handicapped computers' because they do not have a user interface. They state that these devices enable several new attacks that would be impossible with traditional computers.

According to Rivest [17] there is a fundamental conflict between having a secure device and having a 'reasonable customizable user interface' that supports downloading of applications. Rivest argues that the more peripherals a device has, the more features it offers to the user, the more components can be customized or replaced by third party products, the less secure the device is.

While we prefer to work with such convenient and user-friendly devices, all of the above improvements grant new possibilities to the attacker, too. It is natural that customers tend to create their documents on devices where they can work conveniently. However, the user-friendly insecure device makes it impossible to guarantee the integrity of the documents. If such a document is signed by a smart card, the signature cannot prove that the document really originates from the user. Rivest suggests, that a 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.

Another approach takes advantage of the fact that the smart card is physically close to the user. Berta and Vajda [6] propose a solution, where the user can send authentic biometric (audio or video) messages from untrusted terminals. These biometric messages encapsulate the user's identity with the content of the message. In order to protect their integrity, they are strengthened by simple algorithmic authenticators, and are signed by the smart card along with a secure timestamp. The authenticity of the message can be determined by verifying the signature of the card, and comparing the timestamps supplied by the user and the smart card. This method works with today's smart cards, but relies on the fuzziness of biometry and is not a purely cryptographic solution.

The solution of Clarke *et al.* [8] 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. This camera-based device analyzes the contents of the screen, and compares it with the data received on an authentic channel. The device warns the user via a led in case of any difference. Although this device is currently technically infeasible, this solution would enable authentic communication without requiring the user to perform any calculations. However, such a complex device is hard to believe to be tamper resistant, on the other hand this futuristic device would still not solve the problem of customized user interfaces raised by Rivest [17].

*2.2.2 Human-computer cryptography.*

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 a method to protect the secrecy or authenticity of the message alone, 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.

Methods proposed by Naor and Pinkas [14] rely on visual cryptography, which uses transparencies placed on the computer's screen. [15] Their algorithm relies on a one-time-pad, where the xor operation is accelerated by the fast visual processing of the human being. The key is composed by the transparent and non-transparent sectors on the transparencies. The required key-size is very large (especially for long messages), thus visual cryptographic keys must be stored. However, methods of Naor and Pinkas enable a remote partner to send authentic messages to a user at an untrusted terminal, or to identify the user in a secure way, while the basic visual cryptography enables private communication towards the user.

Matsumoto [13] developed a human identification scheme that enables challenge and response identification of humans at untrusted terminals. His solution relies on an assumption that humans can easily understand and 'decode' certain images, while computers have trouble with it. The remote partner transmits a one-time key via such 'questions', and the user combines the answer with this one-time key. He suggests that such a scheme could be used for encryption, too. However, such a scheme would require the remote computer to select 'questions' from a significantly large space, which can be problematic. Moreover, the scheme can be undermined if the attacker can use human interaction, too.

The solitaire algorithm of Schneier [19] provides strong encryption, and uses a deck of card for keying. The key is the initial order of the deck. As the deck is shuffled, it is used as a pseudo-random number generator. Solitaire is a stream-cipher that modularly adds the output of this PRNG to the plaintext. 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.

## 3. Model

Let $U$ denote the user who wishes to communicate with the remoter partner $R$ using the untrusted terminal $T$. While $U$ is a human being, $R$ and $T$ are computers. User $U$ would like to send the message $m$ to $R$, and tries to 'protect' (encrypt or authenticate) it by combining it with the secret key $k$, which is a shared secret between $U$ and $R$. Both $m$ and $k$ are strings of characters from a binary alphabet $I = \{0,1\}$. Parties $U$, $R$ and $T$ are able to execute various efficient algorithms (randomized algorithms of polynomial complexity in their input parameter [11]) that perform an $I^* \to I^*$ mapping.

The key $k$ is $n$-bit-long, where $n$ is a security parameter. We assume the message length $\text{length}(k) < \text{length}(m) \leq p(n)$, where $p(n)$ is a polynom.

For input $x$ algorithm $h$ produces $h(x)$ as output. Notation $h(x)$ also implies that algorithm $h$ can process $x$ as its input. Henceforth, notation $x|y$ stands for the concatenation of strings $x$ and $y$. Furthermore, notation $[\alpha]$ stands for the value of the expression $\alpha$.

Using the above notations, we give a definition for the notion of computational easy and hard. Naturally, both easy and hard are relative to the amount of resources a certain party ($U$, $R$ or the attacker) has. According to literature (e.g., [10]), we express such resource constraints as functions of the security parameter $n$.

**Definition 1.** Let $a$ and $b$ be two efficient algorithms. We say that it is *$t(n)$-hard to compute $b(x)$ from $a(x)$*, if for all efficient algorithms $h$ with $t(n)$ resources and input $a(x)$, for all polynomials $p(n)$, and for large enough $n$:

$$\Pr_x\Big(b(x) = h_{t(n)}\big(a(x)\big)\Big) < \frac{1}{p(n)} ,$$

where $x$ is uniformly chosen from its space.

Based on the above definition, we introduce predicate $\mathrm{hard}\big(a(x), b(x), t(n)\big)$ that takes value true if computing $b(x)$ based on $a(x)$ is $t(n)$-hard. Otherwise, the predicate takes value false.

The above definition follows the usual formalism of the asymptotic approach of algorithm complexity theory. Note that the above problem can only be addressed probabilistically, because the attacker can always guess $b(x)$ with a non-zero probability. A problem is considered hard, if it is hard to solve it on the average, so the probability of the attacker's success is negligible. Negligible means being bounded above by all functions of the form $\frac{1}{p(n)}$. [10] Thus, the higher the security parameter is, the harder the problem is, so the lower chance of success is allowed for the attacker.

**Definition 2.** Let $a$ and $b$ be two efficient algorithms. We say, it is *$t(n)$-easy to compute $b(x)$ from $a(x)$*, if (for algorithms $a$, $b$), there exists an efficient algorithm $h$ with $t(n)$ resources and input $a(x)$, such that for any[1] $x$:

$$b(x) = h_{t(n)}\big(a(x)\big) .$$

Based on the above definition, we introduce predicate $\mathrm{easy}\big(a(x), b(x), t(n)\big)$ that takes value true if computing $b(x)$ based on $a(x)$ is $t(n)$-easy. Otherwise, the predicate takes value false.

Using $t(n)$-easy and $t(n)$-hard, we define ciphers and message authentication codes (MAC).

**Definition 3.** Algorithm $f$ is a *$t_{\mathrm{attacker}}(n)$-strong cipher*, if it is $t_{\mathrm{attacker}}(n)$-
-hard to obtain the whole input $M$ of algorithm $f$ from $f(k|M)$ without knowing key $k$. If the key is known, $t_{\mathrm{encrypt}}(n)$ resources are required to perform the

[1] This definition of easy can be generalized if we allow an $\varepsilon$ error rate for the user. However, such generalization does not yield any more possibilities for the user.

encryption and $t_{\mathrm{decrypt}}(n)$ resources are required for decryption. Formally:

$$\begin{aligned}&\mathrm{cipher}(f,\, t_{\mathrm{encrypt}}(n), t_{\mathrm{decrypt}}(n), t_{\mathrm{attacker}}(n)) \iff \\ &\qquad \mathrm{hard}\big(f(k|M), M, t_{\mathrm{attacker}}(n)\big) \\ &\qquad \wedge\, \mathrm{easy}\big(k|M, f(k|M), t_{\mathrm{encrypt}}(n)\big) \wedge \mathrm{easy}\big(f(k|M)|k, M, t_{\mathrm{decrypt}}(n)\big)\,.\end{aligned}$$

Definition 3 of an encryption resembles the well-known security definition of "plaintext recovery" for symmetric key encryption transformation. [5] The main difference is, that we do not allow access to an encryption oracle. The reason is that we assume one-time keying (as described below).

**Definition 4.** Algorithm $f$ can *compute $t_{\mathrm{attacker}}(n)$-strong message authentication code* if—based on one pair of observed input $M$ and output MAC—it is $t_{\mathrm{attacker}}(n)$-hard to present a different input $M'$ and corresponding MAC without knowing the key. If the key is known, it is $t_{\mathrm{calc}}(n)$-easy to compute the MAC, and $t_{\mathrm{check}}(n)$-easy to check it. Formally:

$$\begin{aligned}&\mathrm{mac}(f,\, t_{\mathrm{calc}}(n),\ t_{\mathrm{check}}(n),\ t_{\mathrm{attacker}}(n)) \iff \\ &\qquad \mathrm{hard}\big(M|f(k|M), M'|f(k|M'), t_{\mathrm{attacker}}(n)\big) \\ &\qquad \wedge\, \mathrm{easy}\big(k|M, f(k|M), t_{\mathrm{calc}}(n)\big) \wedge \mathrm{easy}\big(k|M''|x, \big[f(k|M'') = x\big], t_{\mathrm{check}}(n)\big)\,,\end{aligned}$$

where $M \neq M'$ (but both $M = M''$ and $M \neq M''$ are possible).

We also have to define the amount of resources $U$, $R$ and $T$ have. Since $R$ and $T$ are computers, they have $t_{\mathrm{computer}}(n)$ resources. Both of them are able to execute algorithms of polynomial complexity in $n$. Definition of the $t_{\mathrm{human}}(n)$ resources of the human being is more difficult. Heuristically, $t_{\mathrm{human}}(n) \ll t_{\mathrm{computer}}(n)$. Our implicit definition of $t_{\mathrm{human}}(n)$ is given by the following natural way:

$$\begin{aligned}\models \neg\, \exists\, f\big\{&\mathrm{cipher}(f,\ t_{\mathrm{human}}(n),\ t_{\mathrm{computer}}(n),\ t_{\mathrm{computer}}(n))\big\} \wedge \\ \neg\, \exists\, f\big\{&\mathrm{cipher}(f,\ t_{\mathrm{computer}}(n),\ t_{\mathrm{human}}(n),\ t_{\mathrm{computer}}(n))\big\} \wedge \\ \neg\, \exists\, f\big\{&\mathrm{mac}(f,\ t_{\mathrm{human}}(n),\ t_{\mathrm{computer}}(n),\ t_{\mathrm{computer}}(n))\big\} \wedge \\ \neg\, \exists\, f\big\{&\mathrm{mac}(f,\ t_{\mathrm{computer}}(n),\ t_{\mathrm{human}}(n),\ t_{\mathrm{computer}}(n))\big\}\,.\end{aligned} \tag{1}$$

This way, we defined $t_{\mathrm{human}}(n)$ by claiming that no $t_{\mathrm{computer}}(n)$-strong cipher and no $t_{\mathrm{computer}}(n)$-strong mac exist, that can be executed with $t_{\mathrm{human}}(n)$ resources at the coding or decoding side. Thus, $U$ is able to perform weak encryption or weak authentication only, that can be successfully attacked by the terminal with high probability.

Assume, we are in a world, where the above limitations hold. We also suppose that the Kerckhoff principle is valid, so attacker $T$, knows every algorithm $U$

and $R$ uses, but does not know the secret key $k$. In order to make our proposition more general, we make the environment as advantageous for the user as possible. Thus, we assume that:

- One-time keying is used, so key $k$ is replaced after each message $m$ is sent. (Note that length$(k)$ < length$(m)$, so a one-time-pad cannot be used.)
- In case of secrecy, the attacker is able to eavesdrop only, and cannot modify the messages on the channel.
- In case of message authenticity, the attacker is active.

After these preparations we are able to examine the following problem: Is it possible for $U$ and $R$ to solve the problems of encryption and message authentication by interaction, i.e., not in one step, but in several interactive protocol steps?

## 4. Secrecy

In this section we examine if $U$ with $t_{\text{human}}$ resources can transmit a secret message to $R$ using a finite two-party protocol.

**PROPOSITION 1.** *If user $U$ cannot encrypt message $m$ with $t_{\text{human}}(n)$ resources with a security that cannot be broken with $t_{\text{computer}}(n)$ resources with significant probability, then no $N$-step-long protocol $P$ exists between $U$ and $R$ that has the following properties:*

(S1) *$P$ enables $R$ to decrypt $m$ with $t_{\text{computer}}(n)$ resources,*

(S2) *$P$ prevents the attacker (who also has $t_{\text{computer}}(n)$ resources but does not know $k$) from decrypting $m$.*

P r o o f. We consider the following general protocol for interactive encryption:

Initially: $\sigma_0 = \emptyset$.

1. $U \to R$: $f_1(k|M_1)$, where $M_1 = m|\sigma_0$,
   $\sigma_1 = f_1(k|M_1)$,

2. $R \to U$: $g_2(k|M_2)$, where $M_2 = \sigma_1$,
   $\sigma_2 = \sigma_1|g_2(k|M_2)$,

... ...

$(2L-1)$. $U \to R$: $f_{2L-1}(k|M_{2L-1})$, where $M_{2L-1} = m|\sigma_{2L-2}$,
   $\sigma_{2L-1} = \sigma_{2L-2}|f_{2L-1}(k|M_{2L-1})$,

$(2L)$. $R \to U$: $g_{2L}(k|M_{2L})$, where $M_{2L} = \sigma_{2L-1}$,

$\sigma_{2L} = \sigma_{2L-1}|g_{2L}(k|M_{2L})$,

... ...

$N.$ $U \to R$: $f_N(k|M_N)$, where $M_N = m|\sigma_{N-1}$,
$\sigma_N = \sigma_{N-1}|f_N(k|M_N)$,

where in each step $j$, $\sigma_j$ denotes all the data that was interchanged by $U$ and $R$ via the public channel by $U$ and $R$, thus $\sigma_j$ denotes the database of the attacker, too. We consider $N$-step-long protocols, so $R$ is able to acquire $m$ after step $N$ only.

The proposition (S1 and S2) can be formalized as follows:

$$(1) \to \neg \exists \sigma_N \{\mathrm{hard}(\sigma_N, m, t_{\mathrm{computer}}(n)) \wedge \mathrm{easy}(k|\sigma_N, m, t_{\mathrm{computer}}(n))\}. \quad (2)$$

In contrary, assume that:

$$\exists \sigma_N \{\mathrm{hard}(\sigma_N, m, t_{\mathrm{computer}}(n)) \wedge \mathrm{easy}(k|\sigma_N, m, t_{\mathrm{computer}}(n))\}. \quad (3)$$

If algorithm $f_j$ can be executed by $U$, then user $U$ has enough resources to run it:

$$\models \forall f_j \{\mathrm{easy}(k|M_j,\ f_j(k|M_j),\ t_{\mathrm{human}}(n))\}. \quad (4)$$

According to the assumption about the abilities of the human (1), no algorithm that $U$ can run, can be a cipher. So, according to (4), none of the algorithms $f_j$ can be a cipher. This has the following implication:

$$\models \forall f_j \{\mathrm{easy}(k|M_j, f_j(k|M_j), t_{\mathrm{human}}(n)) \\ \to \neg\ \mathrm{hard}(f_j(k|M_j), M_j, t_{\mathrm{computer}}) \vee \neg\ \mathrm{easy}(f_j(k|M_j)|k, M_j, t_{\mathrm{computer}}(n))\}. \quad (5)$$

Note that $M_j = m|\sigma_{j-1}$ if $j$ is odd.

User $U$ cannot compute a cipher, but can choose between two lesser alternatives. One of them is to choose an algorithm, where $\neg\ \mathrm{easy}(f_j(k|M_j)|k, M_j, t_{\mathrm{computer}}(n))$. This means, remote partner $R$ is unable to obtain $m$. The other alternative is to choose an algorithm, where $\neg\ \mathrm{hard}(f_j(k|M_j), M_j, t_{\mathrm{computer}})$. Such an algorithm is a weak encryption, where the attacker might be able to obtain $m$. This latter would violate the S2 property, so user $U$ should choose an $f_j$ where:

$$\models \forall f_j \{\neg\ \mathrm{easy}(f_j(k|M_j)|k, M_j, t_{\mathrm{computer}}(n))\}. \quad (6)$$

According to (3), $R$ should be able to obtain $m$ after step $N$ using an algorithm $m = g_{N+1}(k|\sigma_N)$ from $k$ and $\sigma_N$, while the attacker can be successful with a negligible probability only.

$$\models \mathrm{hard}(\sigma_N, m, t_{\mathrm{computer}}(n)) \wedge \mathrm{easy}(k|\sigma_N, m, t_{\mathrm{computer}}(n)). \quad (7)$$

Let's substitute $\sigma_N$ with $\sigma_{N-1}|f_N(k|m|\sigma_{N-1})$ into (7).

$$\begin{aligned}\models\ &\mathrm{hard}\big(\sigma_{N-1}|f_N(k|m|\sigma_{N-1}), m, t_{\mathrm{computer}}(n)\big)\\ &\wedge \mathrm{easy}\big(k|\sigma_{N-1}|f_N(k|m|\sigma_{N-1}), m, t_{\mathrm{computer}}(n)\big)\,.\end{aligned}$$

The above formula can be simplified if we suppose that $f_N$ includes $\sigma_N$ in its output. This does not spoil the security of the system, since $\sigma_N$ is already public. Then we obtain:

$$\begin{aligned}\models\ &\mathrm{hard}\big(f_N(k|m|\sigma_{N-1}), m|\sigma_{N-1}, t_{\mathrm{computer}}(n)\big)\\ &\wedge \mathrm{easy}\big(k|f_N(k|m|\sigma_{N-1}), m|\sigma_{N-1}, t_{\mathrm{computer}}(n)\big)\,.\end{aligned}$$

Finally, we substitute $m|\sigma_{N-1}$ with $M_N$:

$$\models \mathrm{hard}\big(f_N(k|M_N), M_N, t_{\mathrm{computer}}(n)\big) \wedge \mathrm{easy}\big(k|f_N(k|M_N), M_N, t_{\mathrm{computer}}(n)\big)\,. \tag{8}$$

Note that (8) contradicts (6) for $f_N$. We have come to a contradiction, so the above protocol does not exist. □

## 5. Message authenticity

In this section the question of message authenticity shall be considered. Assuming that $U$ is unable to provide strong message authenticity in one step, we prove that $U$ and $R$ cannot solve the problem with several interactive protocol-steps either.

Since single steps cannot be authenticated in a 'secure way', so neither $U$, nor $R$ will be able to decide if the messages have been tampered with before the protocol is finished. We shall use the following notation:

$A \Rightarrow B\colon \alpha$ means, party $A$ sends message $\alpha$ towards party $B$ via an insecure channel, where the attacker can modify $\alpha$ on the channel to $\alpha'$, so $B$ receives $\alpha'$.

**Proposition 2.** *If $U$ is unable to perform strong authentication with $t_{\mathrm{human}}(n)$ resources, then no $N$-step-long protocol $P$ exists between $U$ and $R$ that has the following three properties:*

(A1) *$R$ learns message $m$ when protocol $P$ terminates (after step $N$).*

(A2) *$R$ is able to verify that $m$ is authentic.*

(A3) *Without the key $k$, the attacker is unable to produce a valid pair of datablocks $\sigma'_N$ and $m'$ ($m \neq m'$) with significant probability.*

P r o o f . We consider the following general protocol for interactive authentication:

Initially: $\sigma_0 = \emptyset$, $\omega_0 = \emptyset$.

1. $U \to R$: $f_1(k|m)$,
$\sigma_1 = \sigma_0 | f_1(k|m)$, $\omega_1 = \omega_0$,

2. $R \to U$: $g_2(k|m_1)$,
$\sigma_2 = \sigma_1$, $\omega_2 = \omega_1 | g_2(k|\sigma_1')$,

3. $U \Rightarrow R$: $f_3(k|m|\omega_2')$,
$\sigma_3 = \sigma_2 | f_3(k|m|\omega_2')$, $\omega_3 = \omega_2$,

... ...

$(2K)$. $R \Rightarrow U$: $g_{2K}(k|\sigma_{2K-1}')$,
$\sigma_{2K} = \sigma_{2K-1}$, $\omega_{2K} = \omega_{2K-1} | g_{2K}(k|\sigma_{2K-1}')$,

$(2K+1)$. $U \Rightarrow R$: $f_{2K+1}(k|m|\omega_{2K}')$,
$\sigma_{2K+1} = \sigma_{2K} | f_{2K+1}, (k|m|\sigma_{2K}')$, $\omega_{2K+1} = \omega_{2K}$,

... ...

$N$. $U \Rightarrow R$: $f_N(k|m|\omega_{N-1}')$,
$\sigma_N = \sigma_{N-1} | f_N(k|m|\omega_{N-1}')$, $\omega_N = \omega_{N-1}$.

$R$ becomes more and more confident in the authenticity of $m'$ with every received datablock $\sigma_j$, because the probability of a successful attack decreases continuously. The protocol terminates at the $N$th step, when this probability is considered small enough, so that $R$ can verify the authenticity of $m'$. $R$ can check the authenticity of $m'$ by recalculating the $\sigma_j$ values. Note that both $U$ and $R$ are capable of executing probabilistic algorithms too, so $R$ has to determine if $\sigma_N'$ is a possible outcome of a valid protocol run. $R$ accepts $m'$ as authentic if:

$$F(k|m'|\omega_N) \equiv f_1(k|m'|\omega_0) \,|\, f_3(k|m'|\omega_2) \,|\, \dots \,|\, f_N(k|m'|\omega_{N-1}) = \sigma_N' . \tag{9}$$

The attacker is successful in the above protocol, if there is a non-negligible probability of $R$ accepting $m'$ as authentic, where $m' \neq m$.

The proposition (A1 and A2 and A3) can be formalized as follows:

$$\begin{aligned}(1) \to \neg \exists\, \sigma_N \big\{ &\mathrm{hard}\big(m|\omega_N|\sigma_N, m'|\omega_N|\sigma_N', t_{\mathrm{computer}}(n)\big) \\ &\wedge \mathrm{easy}\big(k|m'|\sigma_N'|\omega_N, \big[F(k|m'|\omega_N) = \sigma_N'\big], t_{\mathrm{computer}}(n)\big)\end{aligned} \tag{10}$$

where $m \neq m'$.

In contrary, assume that the above protocol provides secure authentication, i.e:

$$\exists\, \sigma_N \big\{ \operatorname{hard}\big(m|\omega_N|\sigma_N, m'|\omega_N|\sigma'_N, t_{\text{computer}}(n)\big) \wedge m \neq m' \\ \wedge \operatorname{easy}\big(k|m'|\sigma'_N|\omega_N, \big[F(k|m'|\omega_N) = \sigma'_N\big], t_{\text{computer}}(n)\big) \big\} \tag{11}$$

where $m \neq m'$.

Algorithm $F$ has the following properties:

- $U$ is able to run $F$. According to (9), the execution of $F$ requires as much resources as executing all of the algorithms $f_i$ sequentially. Formally:

$$\models \operatorname{easy}\big(k|m'|\omega_N, \sigma'_N, t_{\text{human}}(n)\big)\,. \tag{12}$$

- If $U$ is able to run an algorithm, $R$ is able to do it too, because $t_{\text{human}} < t_{\text{computer}}$. Thus, $R$ is able to check $F$ by simply recalculating it. Formally:

$$\models\ \operatorname{easy}\big(k|m'|\omega_N, \sigma'_N, t_{\text{computer}}(n)\big) \\ \rightarrow \operatorname{easy}\big(k|m'|\omega_N|\sigma'_N, \big[F(k|m'|\omega_N) = \sigma'_N\big], t_{\text{computer}}(n)\big)\,. \tag{13}$$

If we substitute $m|\omega_N$ with $M$ and $m'|\omega_N$ with $M'$, and we also substitute $\sigma_N$ with $F(k|M)$, then from (12) and (13) we obtain:

$$\models\ \operatorname{easy}\big(k|M, F(k|M), t_{\text{human}}(n)\big) \\ \wedge \operatorname{easy}\big(k|M'|\sigma'_N, \big[F(k|M') = \sigma'_N\big], t_{\text{computer}}(n)\big)\,. \tag{14}$$

Because of our assumptions about the user's abilities (1), (14) implies:

$$\neg\ \operatorname{hard}\big(M|F(k|M), M'|F(k|M'), t_{\text{computer}}(n)\big)\,,$$

where $M \neq M'$.

If we revert the above substitutions, we obtain:

$$\neg\ \operatorname{hard}\big(m|\omega_N|\sigma_N, m'|\omega_N|F(k|m'|\omega_N), t_{\text{computer}}(n)\big)\,,$$

where $m|\omega_N \neq m'|\omega_N$, which is equivalent with $m \neq m'$.

The attacker can compute $F(k|m'|\omega_N)$, and is able to use it as $\sigma'_N$ while attacking. Thus, we can substitute $F(k|m'|\omega_N)$ with $\sigma'_N$ and obtain:

$$\neg\ \operatorname{hard}\big(m|\omega_N|\sigma_N, m'|\omega_N|\sigma'_N, t_{\text{computer}}(n)\big)\,, \tag{15}$$

where $m \neq m'$.

Note that (15) contradicts the indirect assumption (11). We have come to a contradiction, so the above protocol does not provide secure authentication. $\square$

## 6. Conclusions

We have shown a formal model that we could use to handle interesting basic problems for insecure terminals. According to our propositions, if the user is unable to perform strong cryptographic operations (like encryption or message authentication) in a single step, then a remote partner cannot help the user in establishing a secure (encrypted or authenticated) channel. Thus, it has completely no sense to develop protocols for the user to communicate with a remote partner via an insecure terminal. The situation remains the same if we involve a smart card, because the user has to communicate with the card through the same insecure channel.

This does not mean that there cannot be any (e.g., physical or procedural) circumstances when the above problems can be solved. Although in this model the user does not have any chance to send encrypted or authenticated messages from a malicious terminal, other 'softer' models might yield more possibilities. For example, we can suppose that the user has significant amount of secure memory. This can be easily achieved if we allow passwords, secret keys and other sensitive information to be written into a copybook. (In certain applications, against certain attackers a copybook may provide adequate security.) Thus, the encryption of messages shorter than the key can be performed using a one-time-pad.

We may also suppose that the user can find two untrusted terminals, that are probably not cooperating with each other. (For example, a borrowed mobile phone and an insecure PC in an internet café.) It might be possible to develop a protocol for this scenario, similarly to the one developed by Kucner and Kutylowski [12], that uses two possibly malicious smart cards against each other to create a secure one.

The way we modeled the user can be considered equivalent with modeling him or her as a slow and weak computer, a handicapped machine. However, humans have other special abilities too, and can perform certain tasks much faster than computers can. They can recognize images quickly (visual cryptography described in Section 2.2.2. is one successful branch for this approach), or they can follow association chains that computers would never figure out. However, it is probably very hard to formalize and quantify such abilities.

Furthermore, in our model, the user was an average person. However, certain humans might have an exceptional amount of resources, enough to perform strong cryptographic operations. Some master chessplayers are able to beat today's best computers in this well defined algorithmic game. Another example: a secret agent might sacrifice a significant amount of time to send one single authentic message.

Untrusted terminals pose a severe problem for average people in commercial applications. Probably, the future solution for this problem will be based on a special hardware device. For instance, a super smart card will evolve in the future, that has a direct interface towards the user, and makes it possible to view the message before it is signed.

## REFERENCES


[1] ABADI, M.—BURROWS, M.—KAUFMAN, C.—LAMPSON, B.: *Authentication and delegation with smart-cards.*, in: Theoretical Aspects of Computer Software: Proc. of the International Conference TACS '91, Springer-Verlag, Berlin, 1991, pp. 326–345.

[2] ANDERSON, N.: *Why cryptosystems fail. William Stallings*, Practical Cryptography for Data Internetworks, IEEE Computer Security Press, Vol. 37, 1994, pp. 32–40.

[3] ASOKAN, N.—DEBAR, H.—STEINER, M.—WAIDNER, M.: *Authenticating Public Terminals*, Computer Networks, Vol. 3, 1999, pp. 861–870.

[4] BALFANZ, D.—FELTEN, E.: *Hand–held computers can be better smart cards.*, in: Proceedings of the 8th USENIX Security Syposium, Washington, DC., 1999.

[5] BELLARE, M.—ROGOWAY, P.: *Introduction to modern cryptography*, `http://www.cse.ucsd.edu/users/mihir/ cse207/classnotes.html, 2002`.

[6] BERTA, I. ZS.—VAJDA, I.: *Documents from malicious terminals*, in: SPIE Microtechnologies for the New Millenium 2003 (A. Rodriguez-Vazques, D. Abbott, R. Carmona, eds.), Bioengineered and Bioinspired Systems, Vol. 5119, Maspalomas, Spain, 2003, pp. 325–336.

[7] BRANDS, S.—CHAUM, D.: *Distance-bounding protocols.*, in: EUROCRYPT '93, Lecture Notes in Comput. Sci., Vol 765, Springer-Verlag, Berlin, 1993, pp. 344–359.

[8] CLARKE, D.—GASSEND, B.—KOTWAL, TH.—BURNSIDE, M.—VAN DIJK, M.—DEVADAS, S,—RIVEST, R.: *The untrusted computer problem and camera–based Authentication*, in: Proceedings of the First International Conference on Pervasive Computin table of contents, Lecture Notes in Comput. Sci., Springer-Verlag, London, 2002, pp. 114–124.

[9] GOBIOFF, H.—SMITH, S.—TYGAR, J. D.: *Smart cards in hostile environments*, in: Proceedings of the 2nd USENIX Workshop on Electronic Commerce, 1996, 23–28.

[10] GOLDREICH, O.: *The foundations of modern cryptography*, in: Proceedings of CRYPTO '97, Notes in Comput. Sci., Vol. 1294, Springer-Verlag, 1997, `http://theory.lcs.mit.edu/~oded/frag.html`.

[11] GOLDREICH, O.: *Introduction to complexity theory*, `http://www.wisdom.weizmann.ac.il/~oded/cc02.html`, 1997.

[12] KUCNER, D.—KUTYLOWSKI, M.: *How to use untrusty cryptographic devices*, in: TATRACRYP '03, The 3rd Central European Conference on Cryptology, Bratislava, Slovak Republic, 2003.

[13] MATSUMOTO, T.: *Human–computer cryptography: An attempt*, in: In ACM Conference on Comput. Commun. Security, 1996, pp. 68–75.

[14] NAOR, M.—PINKAS, B.: *Visual authentication and identification*, in: CRYPTO '97, Lecture Notes in Comput. Sci., (B. Kaliski, ed.), Vol. 1294, Springer-Verlag, Berlin, 1997, pp. 322–336.

[15] NAOR, M.—SHAMIR. A.: *Visual cryptography*, in: Lecture Notes in Comput. Sci., Vol. 950, 1995, pp. 1–12, `http://citeseer.nj.nec.com/naor95visual.html`.

[16] RANKL, W.—EFFING, W.: *Smart Card Handbook*, John Wiley & Sons, 2nd edition, 1997.

[17] RIVEST, R.: *Issues in cryptography*, in: Computers, Freedom, Privacy 2001 Conference, `http://theory.lcs.mit.edu/~rivest/Rivest-IssuesInCryptography.pdf`, 2001.

[18] SCHNEIER, B.—SHOSTACK, A.: *Breaking up is hard to do: Modelling security threats for smart cards*, in: USENIX Workshop on Smart Card Technology, Chicago, Illinois, USA, `http://www.counterpane.com/smart-card-threats.html`, 1999.

[19] SCHNEIER, B.: *The solitaire encryption algorithm*, `http://www.counterpane.com/ solitaire.htm`, 1999.

[20] STABELL-KULO, S.—ARILD, R.—MYRVANG, P. H.: *Providing authentication to messages signed with a smart card in hostile environments*, in: Usenix Workshop on Smart Card Technology, Chicago, Illinois, USA, 1999, 10–11.

[21] THOMPSON. K.: *Reflections on trusting trust*, in: Communication of the ACM, Vol. 29, 1984, pp. 761–763.

[22] YEE, B.—TYGAR, J. D.: *Secure coprocessors in electronic commerce applications*, in: Proceedings of the 1st USENIX Workshop on Electronic Commerce, 1995, pp. 155–170.

Received October 15, 2003

*Laboratory of Cryptography and Systems Security*
*Department of Telecommunications*
*Budapest University of Technology and Economics*
*H–1521, BME–IT P. O. Box 91*
*HUNGARY*

*E-mail*: istvan.berta@crysys.hit.bme.hu
istvan.vajda@crysys.hit.bme.hu