hPIN/hTAN: A Lightweight and Low-Cost e-Banking Solution against Untrusted Computers*

Shujun Li[1], Ahmad-Reza Sadeghi[2,3], Sören Heisrath[3], Roland Schmitz[4] and Junaid Jameel Ahmad[1]

[1] University of Konstanz, Germany
[2] Darmstadt University of Technology and Fraunhofer SIT, Darmstadt, Germany
[3] Ruhr-University of Bochum, Germany
[4] Stuttgart Media University, Germany

Abstract. In this paper, we propose hPIN/hTAN, a low-cost hardware token based PIN/TAN system for protecting e-banking systems against the strong threat model where the adversary has *full* control over the user's computer. This threat model covers various kinds of attacks related to untrusted terminal computers, such as keyloggers, screen scrapers, session hijackers, Trojan horses and transaction generators.
The core of hPIN/hTAN is a secure and easy user-computer-token interface. The security is guaranteed by the user-computer-token interface and two underlying security protocols for user/server/transaction authentication. The hPIN/hTAN system is designed as an open framework so that the underlying authentication protocols can be easily reconfigured. To minimize the costs and maximize usability, we chose two security protocols dependent on simple cryptography (a cryptographic hash function). In contrast to other hardware-based solutions, hPIN/hTAN depends on neither a second trusted channel nor a secure keypad nor external trusted center. Our prototype implementation does not involve cryptography beyond a cryptographic hash function. The minimalistic design can also help increase security because more complicated systems tend to have more security holes. As an important feature, hPIN/hTAN exploits human users' active involvement in the whole process to compensate security weaknesses caused by careless human behavior.

1 Introduction

Nowadays e-banking becomes more and more popular all over the world. A 2010 survey of the American Bankers Association showed that e-banking is now

* An extended abstract of this paper was published in *Financial Cryptography and Data Security: 15th International Conference, FC 2011, Gros Islet, St. Lucia, February 28 - March 4, 2011, Revised Selected Papers*, *Lecture Notes in Computer Science*, vol. 7035, pp. 235-249, Springer, 2012. Companion web page of the paper: `http://www.hooklee.com/default.asp?t=hPIN/hTAN`.

the most preferred banking method of bank customers [2]. There is no doubt that most (more than 90% according to a recent survey made in Germany [36]) users consider security as the most important issue about e-banking. The earliest and simplest defense protecting e-banking systems is user authentication based on static PINs. The end-to-end secure communications between the client (i.e., the web browser installed on the user's computer) and the e-banking server is typically achieved via the SSL/TLS protocol [29, 35].

While SSL/TLS is considered secure [67], static PINs are prone to identity theft based on social engineering attacks, in which the users are spoofed to expose their PINs. One of the most prevailing social engineering attacks is phishing attack [40]. In its simplest form the attacker (called phisher) sends phishing emails to lure gullible users to disclose their PINs on a bogus e-banking web site. Once the attacker gets the PIN of a victim, he will be able to steal the victim's money by logging into the e-banking system. To provide higher security, two-factor authentication has been widely deployed by financial institutions for strengthening their e-banking systems. The most prominent two-factor authentication scheme used for e-banking is PIN/TAN, which uses static PINs for login and one-time TANs (Transaction Authentication Numbers) for online transactions [69].

In early implementations of PIN/TAN, each user gets a paper list of n TANs from the financial institute, and the TANs are used for online transactions in a sequential order. After one TAN is consumed, the user should strike it from the TAN list. This approach has an obvious security problem: while asking for too many unused TANs will definitely raise the user's suspicion, it is often not very difficult for a phisher to lure one or a small number of TANs which will allow him to make one or more online transactions (see, e.g., [40, Section 8.2]). To overcome this security problem, indexed TANs (iTANs) were introduced by some financial institutes [7, 58]. The idea is simple: the n TANs are indexed by n distinct sequence numbers $1, \ldots, n$, and for each online transaction, the user has to input the TAN with a specific index for confirmation. In case a phisher lured $k < n$ iTANs from a user, he will be able to input the correct TAN for the next online transaction with probability $k/n \ll 1$. If at most m consecutive errors of TAN input are allowed, the phisher will manage to make an online transaction with probability $1 - (1 - k/n)^m$. When $n = 100$, $k = 1$ and $m = 3$, the probability is about 0.03.

While iTANs can reduce the risk of simple phishing attack in a probabilistic sense, it does not offer any security against man-in-the-middle (MitM) attacks, in which the adversary controls the communication channel between the user and the e-banking server. In a typical MitM attack, the adversary establishes an SSL/TLS connection with the e-banking server and another one with the user, and then forwards the PIN and TANs from the user to the e-banking server as usual, but tampers with the transaction data in real time so that neither the user nor the e-banking server notices the ongoing attack. A phishing site can be easily configured to be a MitM proxy. Although this security problem has been known [51] and demonstrated [3] as early as in 2005, so far PIN/iTAN still

remains the most prevailing two-factor authentication scheme in some countries such as in Germany [36].

MitM attacks can be made stronger if the attacker partially/fully compromises the user's computer. This is possible due to the wide spread of malware over the Internet. Some malware can inject malicious code into the web browser, so that the attacker can do more than in MitM attacks: monitoring the user's input in the web browser, redirecting the user to a fake web site, modifying the contents of web pages shown in the web browser, and so forth. This kind of attacks are sometimes called man-in-the-browser (MitB) attacks [32]. Other malware such as Trojans or rootkits can even allow the attacker to take *full* control over the user's computer. In the worst case, all the software, hardware drivers, the operating system and even reprogrammable hardware are under the *full* control of the attacker, thus rendering the user's computer *totally* untrusted.

In this paper, we consider e-banking solutions against attacks related to *fully* untrusted computers, and call them "man-in-the-computer" (MitC) attacks. Depending on the contexts, MitC attacks have different names in the literature, e.g. malware-based attack or malicious software attack [68], Trojan attacks [56], content-manipulation attacks [47], transaction generator attack [38], active observer attack [43], and so on.

Since the main goal of MitC attacks is transactions manipulation rather than identity theft, it is clear that the corresponding solutions for secure e-banking aim at providing transaction authentication. Roughly speaking, there are two basic approaches to achieve transaction authentication: the first approach requires message authentication of the transaction data sent from the user to the server, and the second one requires secure transmission of the transaction data and a transaction-dependent TAN back to the user for re-confirmation of the requested transaction. Normally, the first approach involves a trusted input method of the transaction data and a trusted channel for secure data transmission from the user to the server, and the second one involves a trusted out-of-band (OOB) or encrypted channel for data transmission from the server back to the user. The re-confirmation in the second approach is achieved by simply sending the transaction-dependent TAN back to the server without protection.

A typical solution in use is mTAN (also known as mobileTAN or smsTAN) deployed by many financial institutions around the world [5,50]. The mTAN system follows the second approach, and use the cellular network as the additional trusted OOB channel to send the transaction data and the transaction-dependent TANs back to the user via SMS. The user verifies the transaction data and then sends the TAN to the server to re-confirm the transaction. While mTAN is able to offer an acceptable level of security against MitC attacks, the OOB channel based on cellular network and a smart phone is not always available at the user side. Furthermore, the cellular network is not free from potential attacks [63]. In addition, the user's smart phone may also be infected by malware [62] and is still prone to some more advanced attacks such as SIM card swop frauds [55] and insider attacks from the telecommunication service providers [37]. The high complexity of today's smart phones may also lead to potential security holes induced

by software bugs. For instance, according to a recent news report [49], criminals are offering € 25000 for a discontinued Nokia 1100 smart phone with an alleged software bug, which allows the criminals to receive mTANs that should be sent to someone else's phone. Finally but not the least, the inevitable costs incurred by sending SMS messages is also a major drawback of mTAN.

In addition to mTAN, there are many other e-banking solutions against MitC attacks. A lot of these solutions are based on hardware devices such as general-purpose personal computing devices (smart phones or PDAs), smart card readers or USB-tokens. Although many of them do work in practice, they all have non-trivial drawbacks, which include relatively high implementation/maintenance costs, dependence on an external network/server, low usability, doubtful security, and so forth. As far as we know, all hardware-based solutions depend on at least one of the following three components: second trusted channel, secure keypad, and encryption. Some solutions also require optical devices such as digital cameras or optical sensors. In Section 5, we will give a more detailed survey on existing solutions and their drawbacks.

Our contributions: In this paper, we propose hPIN/hTAN, the first (to the best of our knowledge) hardware-based solution against MitC attacks that depends on none of the following components: second trusted channel, secure keyboard, trusted third-party, encryption. The main design goal of the hPIN/hTAN system is to achieve a better tradeoff between security and usability with a low-cost and easy-to-use USB-token. The security requirements are guaranteed through a secure user-computer-token interface and two security protocols. The interface can also reduce or compensate careless errors made by humans who may become the weakest link in the whole system.

Paper organization: In the next section, we introduce our hPIN/hTAN system in detail. The security analysis of hPIN/hTAN is given in Sec. 3. The usability, deployment issues and some possible variants of the basic hPIN/hTAN system are discussed in Sec. 4. In Sec. 5, we overview related work, their drawbacks and compare hPIN/hTAN with existing solutions. The last section concludes the paper and gives some planned work in the future.

2 The Proposed hPIN/hTAN System

Our hPIN/hTAN system is composed of two parts – hPIN and hTAN, which protect the login process and online transactions, respectively. The hPIN part also protects the hTAN part from potential abuse by enabling it only after the user successfully passes the hTAN part. In the following, we discuss the model, notations, requirements and the two protocols involved, respectively.

System Model: As shown in Fig. 1, the involved parties in hPIN/hTAN are a human user U, a trusted USB-token T issued by the financial institute to the user, an untrusted terminal computer C (i.e., a MitC attacker), and the remote e-banking server S. In a typical scenario, the human user U plugs the USB-token T into a USB-port of the untrusted computer C, tries to access the remote e-banking server S and then makes some online transactions. We assume that the

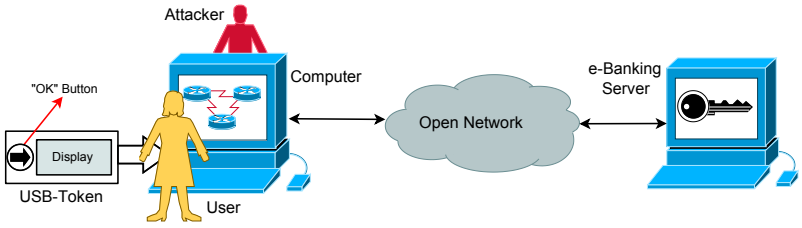

Fig. 1. The threat model of the hPIN/hTAN system.

e-banking server S is trusted, which is a reasonable assumption in practice.[5] The main threat we consider is the MitC attacker who is able to both observe and manipulate communications between U and C, T and C, S and C. Moreover, we assume the USB-token T is a trusted device to the user so that the MitC attacker has no access to any data stored inside T.

The notations used in this paper are summarized in Table 1.

Table 1. Symbols and notations used in this paper.

IDU	User ID.
K_T	Secret key shared between T and S.
PIN	n-character PIN shared between U and T.
PIN(i)	The i-th character of PIN.
s	Salt used to be hashed together with PIN.
STD	Sensitive transaction data that are authenticated.
NSTD	Non-sensitive transaction data that are not authenticated.
$h(\cdot)$	m-bit cryptographic hash function.
HMAC($K_T, \cdot$)	HMAC constructed based on $h(\cdot)$.
$a \parallel b$	Concatenation of a and b.
K_T^*	$= K_T \oplus h(\text{PIN} \parallel s)$ (stored in T).
PIN*	$= h(\text{PIN} \parallel K_T \parallel s)$ (stored in T).
$\mathcal{F}_i : \mathbb{X} \to \mathbb{Y}$	Random code mapping PIN(i) $\in \mathbb{X}$ to a printable character in $\mathbb{Y}$.
C_T, C_S	Two counters stored in T and S.
V_T, V_S	Maximal numbers of consecutive failures allowed by T and S.

Security Requirements: Under the above system model, hPIN/hTAN is designed to achieve the following security requirements:

1. *PIN confidentiality*: the attacker cannot get the user's PIN in clear, except for a negligible probability;
2. *User authenticity*: the attacker cannot access the e-banking server without the presence of the legitimate user, except for a negligible probability;
3. *Server authenticity*: the attacker cannot cheat the user into connecting to a fake e-banking server, except for a negligible probability;

[5] Potential threats related to insiders will be briefly discussed later in Sec. 3.7.

4. *Transaction integrity/authenticity*: the attacker cannot modify/forge a transaction without being detected, except for a negligible probability.

Note that the second and the third requirements are equal to mutual authentication between the user U and the server S.

System Requirements: The USB-token used in the hPIN/hTAN system is designed following a minimalistic principle. In addition to the basic components for building a USB device, it also includes a small display and an "OK" button. Two security protocols are embedded in the USB-token to implement user/server/transaction authentication. For our prototype system, we chose two security protocols based on an m-bit keyed hash function (HMAC). We avoid using any more cryptography (e.g., asymmetric algorithms) to minimize the system complexity.

When a USB-token is manufactured, an m-bit secret key K_T and an initial PIN are assigned to it, where the PIN is composed of n characters in a finite set $\mathbb{X}$. The secret key K_T is crucial for the security of the hPIN/hTAN system, and is never shown in clear to the user and cannot be changed by the user. In contrast, the PIN is mainly used to protect the USB-token from theft and loss, and can be changed later by the user. As a whole, in the USB-token, the following data are stored in its non-volatile memory:

$$\text{IDU},\ s,\ K_T^* = K_T \oplus h(\text{PIN} \parallel s),\ \text{PIN}^* = \text{HMAC}(K_T, \text{PIN} \parallel s),\ C_{\mathsf{T}},$$

where C_{T} is used to signal locking the USB-token if more than V_{T} wrong PINs have been entered consecutively. The salt s is used to frustrate rainbow table attacks. Note that K_T is encrypted, and cannot be recovered without access to the correct PIN. The e-banking server stores the following data for the user:

$$\text{IDU},\ h(K_T),\ C_{\mathsf{S}},$$

where C_{S} is used to signal locking the user's account if more than V_{S} consecutive failures of user authentication have happened. Both C_{T} and C_{S} are initialized to be zeros when the USB-token is issued to the user.

Based on the above system requirements, the following two subsections describe how the hPIN and hTAN parts work. Note that running both parts needs installation of a plugin to the web browser of the terminal computer, which is in charge of communications between the USB-token T and the computer C.

2.1 The hPIN Part

The hPIN part protects the login process via the following two components: authentication of the user to the USB-token, and mutual authentication between the USB-token and the e-banking server. The second component can be implemented by any mutual authentication protocol. In this paper, we choose the SKID3 protocol [21], a generalized edition of the unilateral authentication protocol defined in ISO/IEC 9798-4. More complicated mutual authentication protocols can certainly be used here, but the simple SKID3 protocol is sufficient

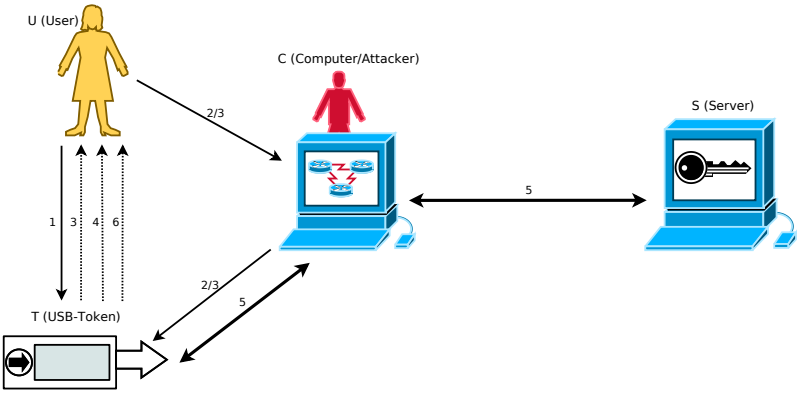

Fig. 2. The hPIN part, where solid lines denote real interactions/communications and dashed lines denote information display (the same hereinafter). The thick solid lines highlight the reconfigurable mutual authentication protocol.

to achieve the security requirements of hPIN/hTAN. Thanks to the simplicity of SKID3, the computational complexity of the hPIN/hTAN is very low. Figure 2 and the following description explain how the whole hPIN part works.

Step 1: U connects T to C, and presses the "OK" button on T.
Step 2: U enters IDU on the untrusted keyboard and sends it to T via C.
Step 3: For $i = 1, \ldots, n$, T and U perform the following interactive protocol:

a) T randomly generates a one-time code $\mathcal{F}_i : \mathbb{X} \to \mathbb{Y}$, shows all codewords $\{\mathcal{F}_i(x)|x \in \mathbb{X}\}$ to U via its display (see Fig. 4 for a typical way of showing the random code on T's display);
b) U enters $\mathcal{F}_i(\text{PIN}(i))$ (which is always a printable character)with the untrusted keyboard of C;
c) if U presses the "Backspace" key, T performs $i = i - 1$ and goes to Step 3a; otherwise T decodes $\mathcal{F}_i(\text{PIN}(i))$ and performs $i = i + 1$.

An example: when $\mathbb{X} = \{0, \ldots, 9\}$ and $\mathbb{Y} = \{a, \ldots, z\}$, the following randomly generated code is generated (in Step 3a): $\mathcal{F}_i(0) = z$, $\mathcal{F}_i(1) = u$, $\mathcal{F}_i(2) = i$, $\mathcal{F}_i(3) = e$, $\mathcal{F}_i(4) = n$, $\mathcal{F}_i(5) = g$, $\mathcal{F}_i(6) = b$, $\mathcal{F}_i(7) = h$, $\mathcal{F}_i(8) = a$, $\mathcal{F}_i(9) = k$. Assuming the PIN is "123456" and the current PIN character is 4, the user U presses 'n'-key (in Step 3b). After that T goes to the next round if $i < n$ or finishes Step 3 if $i = n$.

Step 4: T verifies if $\text{PIN}^* = h(\text{PIN} \parallel (K_T^* \oplus h(\text{PIN} \parallel s)) \parallel s)$. If so, then T recovers the secret key as $K_T = K_T^* \oplus h(\text{PIN} \parallel s)$, stores $h(K_T)$ in its volatile memory for future use, shows a "PIN correct" message to the user U via its display, and goes to Step 5; otherwise T performs $C_\mathsf{T} = C_\mathsf{T} + 1$, shows an alert to U and stops. If $C_\mathsf{T} > V_\mathsf{T}$, T locks itself.[6]

[6] The PIN protection mechanism mainly protects the USB-token from theft and loss. If unlocking is necessary for a legitimate user, she has to contact the financial institute to reset the PIN.

Step 5: T and S authenticate each other by following a mutual authentication protocol. When the SKID3 protocol is used, the mutual authentication process works as follows:
$\mathsf{T} \rightarrow \mathsf{S}$: (UID,$r_{\mathsf{T}}$),
$\mathsf{S} \rightarrow \mathsf{T}$: ($r_{\mathsf{S}}$, $H_1 = \text{HMAC}(h(K_T), r_{\mathsf{S}} \parallel r_{\mathsf{T}} \parallel \mathsf{S})$),
$\mathsf{T} \rightarrow \mathsf{S}$: $H_2 = \text{HMAC}(h(K_T), r_{\mathsf{T}} \parallel r_{\mathsf{S}} \parallel \mathsf{T})$,
where r_{S} and r_{T} are nonces generated by S and T respectively.

Step 6: T shows a message on its display to inform U about the result of the mutual authentication protocol in Step 5.

After U successfully logs into the e-banking system with T, she can change the PIN if she wants. To do so, U asks C to signal T about the input of a new PIN. The new PIN can be entered in the same way as in Step 3 of the above hPIN process. After completing the PIN input, U presses the "OK" button on T twice and then T updates the values of K_T^* and PIN*.

2.2 The hTAN Part

The hTAN part protects online transactions from MitC attacks after the user has successfully passes the hPIN process. As shown in the previous subsection, the hTAN part is enabled upon the completion of the hPIN process. After that, a countdown clock will be triggered, so that the hTAN part will be disabled if T is idle for a specific time (e.g., several minutes). Once the hTAN part is disabled, $h(K_T)$ is removed from T's volatile memory.

The core of the hTAN part is a human-computer-token interactive protocol that allows *simultaneous* transaction input on the untrusted keyboard of C and transaction verification via the trusted display of T. This interactive protocol ensures that T receives the real transaction data U wants to make. After that, T runs a transaction authentication protocol to send the real transaction data to S. In our prototype of hPIN/hTAN, we use the same HMAC scheme involved in the hPIN part for construct the transaction authentication protocol, so that the whole system is based on a single hash function and a single HMAC scheme.

Step 1: U clicks a button on the e-banking web page to inform T about the start of a new online transaction attempt. Then, she inputs each STD item one after another on the untrusted keyboard of C by repeating Steps 1–4.
To embed STD verification into the input process, each character in the STD is shown like passwords (e.g., as a number of asterisks) on the untrusted monitor of C, but in clear on the trusted display of T. This can naturally force U to verify the STD *simultaneously* while she is entering the STD.[7]
If U presses "Backspace" key, T shows an eye-catching warning message to

[7] The attacker may choose not to follow this rule, but simply show the STD in clear in the online form. But the user U will immediately know she is interacting with a bogus e-banking server S or malware in her computer C, since the genuine server should never behave this way. As a result, the attacker is forced to adhere to the hTAN protocol.

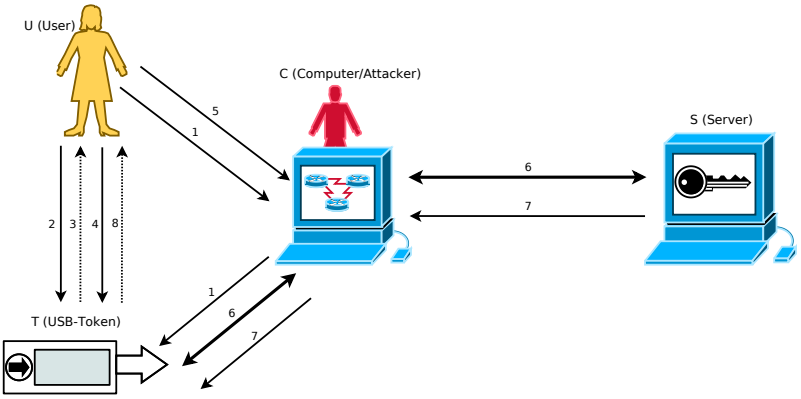

Fig. 3. The hTAN protocol. The thick solid lines highlight the reconfigurable transaction/message authentication protocol.

inform the user for a few seconds and then the previously entered character is deleted. The goal of such a special design is explained later in Sec. 3.3.

Step 2: Upon completion of one STD item, U presses the "OK" button on T.

Step 3: T highlights the current STD item for a few seconds, and prompts U to press the "OK" button again if the current STD item was not changed by C after Step 2.

Step 4: U presses the "OK" button again to approve the current STD item.

Step 5: U inputs NSTD to T by filling a blank on the web page in clear.

Step 6: T sends STD and NSTD to S by running a transaction/message authentication protocol. Here, we use HMAC to build the following protocol:
$\mathsf{T} \to \mathsf{S}$: (IDU, STD, NSTD, r_{T}^*),
$\mathsf{S} \to \mathsf{T}$: ($r_{\mathsf{S}}^*$, $H_3 = \mathrm{HMAC}(h(K_T), r_{\mathsf{S}}^* \parallel r_{\mathsf{T}}^* \parallel \mathrm{STD})$),
$\mathsf{T} \to \mathsf{S}$: (IDU, $H_4 = \mathrm{HMAC}(h(K_T), r_{\mathsf{T}}^* \parallel r_{\mathsf{S}}^* \parallel \mathrm{STD})$),
where r_{T}^* and r_{S}^* are two new nonces generated by T and S, respectively.

Step 7: S checks if $H_4 = \mathrm{HMAC}(h(K_T), r_{\mathsf{T}}^* \parallel r_{\mathsf{S}}^* \parallel \mathrm{STD})$. If so, S executes the requested transaction and sets $M =$"success", otherwise sets $M =$"error". Then, S sends $H_5 = \mathrm{HMAC}(h(K_T), r_{\mathsf{S}}^* \parallel r_{\mathsf{T}}^* \parallel M \parallel \mathrm{STD})$ to T.

Step 8: T checks if $H_5 = \mathrm{HMAC}(h(K_T), r_{\mathsf{S}}^* \parallel r_{\mathsf{T}}^* \parallel$ "success" $\parallel \mathrm{STD})$. If so, it shows "transaction executed", otherwise "transaction failed", on its display.

3 Security of hPIN/hTAN

In this section, we analyze the security of the hPIN/hTAN system, based on the assumption that $h(\cdot)$ and the HMAC scheme are both cryptographic secure against attackers whose computational power is limited by $2^{m/2}$.

3.1 PIN Confidentiality

The PIN protects the USB-token from theft and loss. Leaking the PIN to an attacker actually does not compromise the security of hPIN/hTAN (as long as

the USB-token is not available to the attacker), but it may compromise the user's privacy, since the PIN often relates to the user's personal information such as birthday. In addition, many users share the same PIN/password (or part of it) over multiple e-banking systems, so leaking the PIN of one e-banking system protected by hPIN/hTAN may lead to compromise of other e-banking systems.

The PIN confidentiality is achieved by the use of the n random codes $\mathcal{F}_1$, ..., $\mathcal{F}_n$ in the hPIN process. In Step 2, the USB-token T does not send the n codewords to the untrusted computer C, but shows them on its own display. Since the USB-token is a trusted device, the attacker has no access to any of the n codes and thus is unable to decode the PIN from the user's input $\mathcal{F}_1(\mathrm{PIN}(1)), \ldots, \mathcal{F}_n(\mathrm{PIN}(n))$. Each PIN character is mapped to a printable character by a *different* code, the attacker cannot figure out repeatedly used characters in the PIN, either. Instead, the attacker can only exhaustively search all the possible PINs. This corresponds to a success probability $|\mathbb{X}|^{-n} \ll 1$, when $|\mathbb{X}|^n \gg 1$. Note that an offline attack on the PIN is not possible because no information about the PIN is transmitted to C. An online attack is also impossible because Step 1 requires physical access to T. The above facts imply that the attacker has no any clue to judge if a random guess is correct or not, thus making the brute-force attack useless.

A potential risk exists for the randomness of the codes $\mathcal{F}_i$. If there are some weaknesses in the random number generator of the USB-token T, the attacker may be able to reveal some useful information about the PIN if the number of observed hPIN sessions is sufficiently large. We see this risk is very low, because the local time clock of T does not synchronize with the attacker's. T can also make use of some mechanism to slightly disturb the local time clock in a random manner, which can also depend on K_T^* or PIN^*.

When the PIN does not have a fixed length, i.e., the value of n is not fixed, we can further enhance Step 1b of the hPIN process to hide the PIN length. This can be achieved by constructing a code $\mathcal{F}_i^* : \mathbb{X} \to \mathbb{Y}^* = \mathbb{Y} \cup \mathbb{Y}^2 \cup \cdots \cup \mathbb{Y}^l$, where $l > 1$. The code $\mathcal{F}_i^*$ satisfies the following property: at least two codewords have different sizes. The maximal size of a codeword, i.e., the value of l, is determined by the font size and the display size of the USB-token T. With such an enhanced PIN-entry mechanism, the MitC attacker has to guess the PIN length, too. Assuming the length of each codeword $\mathcal{F}_i^*(j)$ distributes uniformly in $\{1, \ldots, l\}$, the average length of the input string $\mathcal{F}_1^*(\mathrm{PIN}(1)) \cdots \mathcal{F}_n^*(\mathrm{PIN}(n))$ will be $\frac{n(l+1)}{2}$.

3.2 User/Server Authenticity

The mutual authentication between U (actually T) and S is guaranteed by the underlying security protocol in the hPIN part. For the SKID3 protocol, mutual authentication is guaranteed because the attacker can only passively forward communications between U (i.e., T) and S. That is, without the presence of U and T, the attacker cannot authenticate itself to S; without the presence of S, the attacker cannot authenticate itself to T. Note that we do not attempt to prevent the attacker from reading communications between U (i.e., T) and S, since they have been exposed to the attacker by the untrusted computer C.

3.3 Transaction Authenticity/Integrity

Transaction authenticity/integrity is achieved by the hTAN part. There are two stages of transaction authentication: 1) the human-token-computer interactive protocol in Steps 1–4 guarantees the integrity of STD from U to T; 2) the transaction/message authentication protocol in Step 6 guarantees the integrity of STD from T to S. Note that Step 8 is for the integrity of the "success" message from S, so it is independent of the integrity of STD and will not be discussed further.

The human-token-computer interactive protocol (Steps 1–4) ensures that T gets the STD without being manipulated. Since the user has to look at T's display to verify her input and then press the "OK" button twice to show confirmation, T will always receive the real STD that the user intends to input. Thanks to the use of the trusted display of T, the user can fully focus on the correctness of STD in the data input process. As long as the whole STD is correct, there is no risk that it is manipulated by the attacker despite the fact that the user is using an untrusted computer to input STD. This is how *simultaneous* STD input and verification is achieved. The main goal of the special design on "Backspace" key is to prolong the time of deleting previously entered characters so that malicious deletion of STD characters by C can be easily noticed by U.

Although Steps 2–4 look like "verify after input", the real purpose is to resist a competition attack: after U finishes typing the STD, the attacker sends one or more new digits to T and append them to U's STD. If this happens just before U's finger touches the "OK" button, U may wrongly confirm the attacker's STD. By asking U to press the "OK" button in Step 2 and then press it again after a short delay, the above competition attack will become nearly impossible. To detect an ongoing competition attack, U does not need to re-verify the whole STD explicitly, but just pay attention to possible abrupt change of the STD. This is a very easy task since U keeps watching T's display during Steps 1–4.

One may wonder why "simultaneous input and verify" is better than the traditional "verify after input" process. There are three main reasons: 1) recent research [1] has shown that human users are not dependable in distinguishing manipulated e-banking transactions (especially when only a few characters are manipulated) under the "verify after input" setting of mTAN; 2) we believe that asking the user to input and verify STD simultaneously can reduce the total time of STD input and verification (see Sec. 4 for more detail); 3) we believe that the user's active involvement at the very beginning of the hTAN process can help to enhance the user's feeling and awareness of e-banking security.

After T gets STD from the user, it sends STD to S for execution. The transaction authentication and re-confirmation is ensured by the two STD-dependent HMAC values H_3 and H_4. Under the assumption that the HMAC scheme is cryptographically secure, neither H_3 nor H_4 can be manipulated by the attacker with a non-negligible probability. Note that we also make the HMAC values depend on two new nonces to render replay attacks negligible.

3.4 Security against Side Channel Attacks

Since the computer is untrusted, the attacker may be able to launch side channel attacks on the USB-token to gain some useful information that can reduce the system's security or even break the system. Since hPIN/hTAN is an open framework, we assume that the HMAC involved has been hardened to be secure against energy analysis and timing attacks. In this case, we only consider if the user interfaces of the hPIN and hTAN protocols bring new side channel attacks. For hTAN part, all STD elements are actually not secret, so there is no any interesting target to launch a side channel attack. On the other hand, for hPIN part, the PIN could be the target of a timing attack: the malware can measure the response time of each input character $\mathcal{F}_i(\mathrm{PIN}(i))$ to get some information about the relative locations of the PIN digits on the T's display. This is possible because most users look at texts shown on a display in a fixed direction (left to right for most users and right to left for others). This means that the fastest response among all n input characters most likely corresponds to the PIN digit at the leftmost (or rightmost) location. Apparently, such information can be used to reduce the searching space of the PIN. This potential timing attack can be countered by two improvements to the way how the PIN digits and the random code are shown on the T's display: 1) randomly shuffling the n elements of each one-time random code $\mathcal{F}_i$; 2) showing the n elements of each one-time random code in a random order. Both approaches can effectively cancel the correlation information between the response time and the location of the PIN digit, thus rendering the timing attack useless. If there exist other forms of side channel attacks remains a topic of future research.

3.5 Security against System Attacks

Both the hPIN and the hTAN parts of the USB-token are designed to be based on a single finite-state machine. For each state, there is only one direction the user and the USB-token can forward. Once an invalid request of state transition is detected, the whole process will be canceled and the USB-token will go back to its initial state "Ready for hPIN". With such a design, if the untrusted computer tries to launch a system-level attack by disturbing the synchronization between the user and the USB-token, it will simply force the USB-token to go to its original state and gain nothing.

3.6 Security against Physical Access Attacks

In case an adversary gets physical access to the USB-token T, there is a risk that T can be used to access the legitimate user's account. Since the USB-token T is protected by the PIN, the adversary has to break it first. Assuming he randomly guesses the PIN, the success probability is $p_3 \approx 1 - (1 - |\mathbb{X}|^{-n})^{v_1} \approx v_1/|\mathbb{X}|^n$. When $v_1 \ll |\mathbb{X}|^n$, we have $p_3 \ll 1$. While the adversary has only a very tiny chance to be successful by randomly guessing the PIN, he may try to break into the chip or use side-channel attacks to extract the data stored in T. Then, if

$|\mathbb{X}|^n$ is not sufficiently large, an offline brute-force attack to the PIN becomes possible. That is, the attacker can exhaustively search all possible PINs and verify them in the same way as in Step 4 of the hPIN protocol. For a typical PIN, $\mathbb{X} = \{0, \ldots, 9, a, \ldots, z\}$ and $n = 8$, which corresponds to a complexity of $O(2^{31})$ – far from enough to resist such an offline attack. To overcome this problem, we recommend strengthening the hash function used in the hPIN protocol. One possible approach is to replace $h(\text{PIN} \parallel s)$ by its t-fold composition $h^t(\text{PIN} \parallel s)$. This will increase the complexity of verifying all the PINs from $O(|\mathbb{X}|^n)$ to $O(t|\mathbb{X}|^n)$. As a consequence, we can choose the values of t, n and $|\mathbb{X}|$ so that the brute-force attack becomes computationally impractical for most attackers. Note that increasing t too much will slow down the hPIN protocol thus decrease usability, since T only has a low-power CPU. To have a good balance, we recommend $t = 2^{20}$, when $\mathbb{X} = \{a, \ldots, z, 0, \ldots, 9\}$ and $n = 8$. This ensures that the brute-force attack has a complexity $O(2^{61})$. While this recommended setting still requires that T is able to perform $O(2^{20})$ hashing calculations per second, which may not be realistic for the low-power CPU of the USB-token, computational resources can actually be borrowed from the much more powerful computer C. For instance, it can calculate $t_1 \ll t$ rounds of hashing calculations first and then ask C to perform the left ones. Note that C will not be able to launch an offline brute force attack to break PIN due to the unknown salt s. If the user U chooses a weak PIN such that the dictionary attack is possible, the password space $|\mathbb{X}|^n$ may be reduced dramatically and then the complexity of the attack may be feasible. This problem is out of the scope of this paper, and has to be solved with other countermeasures.

3.7 Security against Other Potential Attacks

Social engineering attacks: In the hPIN/hTAN system, the K_T plays the most important role. If an attacker is able to get K_T from the user, the security of the whole system is immediately compromised. We design the hPIN/hTAN system so that both K_T and $h(K_T)$ are never shown in clear to the user. As long as the user herself does not know the secret, a social engineer (such as a phisher) will not be able to get the secret from the user. Note that most hardware-based PIN/TAN systems are also secure against social engineering attacks, as long as there is a secret stored in the hardware device and the user has no direct access to it. In contrast, the iTAN system is not secure, since the user has access to all the TANs printed on her iTAN list.

Malicious code injection attacks: The USB-token is a special-purpose device, and the user is not allowed to update the firmware or install any additional software on her own. This helps prevent malicious code injection by a third party.

Insider attacks: If there are untrusted insiders who have access to the e-banking server, the security of the hPIN/hTAN system may be compromised. For an insider who has only read access to the e-banking server S, we may add one more hash value $\text{HV}^* = h(h(\text{IDU} \parallel K_T) \parallel h(K_T))$ to the data stored in the server S. In Step 7 of the hPIN protocol, the USB-token T sends one more

hash value to the server S for authentication: HV = $h(\text{IDU} \parallel K_T)$. As long as the insider cannot eavesdrop the communications between T and S, he will not be able to get HV and unable to pass the hPIN protocol. A random guess of the value of HV will lead to a success probability of $2^{-(m^*-1)}$. However, if the insider can collude with a MitM attacker, or himself is a MitM attacker, he will be able to get HV from the USB-token T, thus make the hPIN/hTAN system insecure. How to design a system secure against such a stronger threat model is still an open problem.

4 More Discussions

4.1 Usability and Deployment Issues

We have developed a prototype implementation of hPIN/hTAN to test its usability. Three prototype USB-tokens have been produced and the hPIN/hTAN system has been implemented as firmware inside the USB-token and hostware running on a PC with Linux OS installed. Thanks to the simplistic design, the system is extremely lightweight: the size of the firmware is only around 10KB and the data memory requirement is less than 2KB. The actual costs of all components are about around 3–5 € per token. A virtual e-banking web site `http://www.hPIN-hTAN.net` was setup to simulate a genuine e-banking server. Figure 4 shows one prototype USB-token running Step 3 of the hPIN stage.

Fig. 4. One prototype USB-token running the hPIN user authentication step.

A small-scale user study with 20 students and staff members of our universities shown that with a 4-digit PIN the median login time is 27.5 seconds and a transaction with 55 characters can be completed in around 70 seconds (1.27 seconds per character). The overall success rate of logins is $60/66 \approx 91\%$. We expect the login time may reduce significantly after the user becomes more familiar with both the system and the PIN. A survey of the participants showed that none of them had major difficulties understanding how the system works. Most users rated the overall usability of the hPIN/hTAN system as "very usable" and the mean opinion score is 3.65 on a 5-point scale. In the following, we give a qualitative analysis of the usability of hPIN/hTAN.

For the hPIN part, it is clear that the user interacts with T and C only in the first three steps and the following steps are done by T automatically. In Steps 1 and 2, the user only needs to press the "OK" button once and then input her ID, which does not add any additional usability problem compared with the traditional PIN scheme. The user interface in Step 3 is a bit more complicated due to the use of the random codes. To enhance usability, we propose to show the codewords of each random code $\mathcal{F}_i$ on T's display as follows (see also Fig. 4):

$$\begin{array}{cccccccc} 0 & 1 & \cdots & 8 & 9 & \cdots \\ \mathcal{F}_i(0) & \mathcal{F}_i(1) & \cdots & \mathcal{F}_i(8) & \mathcal{F}_i(9) & \cdots \end{array}$$

The first row lists all possible PIN characters and the second shows the corresponding code of each PIN character. This allows the user to simply look for her PIN character $\mathrm{PIN}(i)$ and then input the character below it. With a list of codewords as shown above, an average user can input each $\mathcal{F}_i(\mathrm{PIN}(i))$ within a few seconds. This means the whole PIN can be input within $O(n)$ seconds.

For the hTAN part, user interaction occurs only in Steps 1–5. Step 5 is about NSTD input, which is the same as the traditional TAN scheme, so we do not consider this step. The STD input in Step 1 is very similar to the normal text input in a web page. The only difference is that the user now should look at T's display rather than C's monitor to get visual feedback about what she is entering. By using a USB extension cable, the user can place T just above (or below) her keyboard so that she can easily see T's display. In this setup, the distance between the input device (C's keyboard) and T's display is much smaller than the distance between the input device and C's monitor, so the user is actually in a better condition of STD input. Steps 2–4 are very easy because the user either just waits and observes or simply presses a button on T. As a whole, we expect the additional time spent by an average user will be at the same order of traditional TAN schemes. Note that for TAN/PIN systems, the user has to look for the correct TAN on a paper list or wait for an SMS from the e-banking server, which can consume much more time than the hTAN process.

The simultaneous STD input and verification is especially important for collective transaction. Traditionally, the user has to first input each transaction one after another to C and then carefully double check all transactions again before finally sending them collectively to S. For our proposed hPIN/hTAN system, the confirmation stage is drastically simplified by asking the user to confirm that the STD was not changed in the past a few seconds. This will help to roughly halve the total time the user has to spend on transaction input and verification.

Finally, we discuss some issues about possible deployment of our hPIN/hTAN system in real e-banking systems. The first one is about the manufacturing and distribution of the USB-token to customers. We do not see this as a major issue, since many financial institutes are issuing hardware tokens to their customers and many companies are offering technical support for manufacturing USB-tokens. The second issue is about the upgrade of the e-banking system. The hPIN/hTAN system corresponds to a stand-alone component of the e-banking system – the two-factor authentication module. We feel it should be not difficult to upgrade

this module without changing any other components of the whole e-banking system. This can be shown by the fact that in the past few years many financial institutes upgraded their e-banking systems quite often: from TAN to iTAN and then to mTAN. The third issue is the installation of a plugin in the web browser of the user's computer. This is not a problem because nowadays many financial institutions have already deployed hardware solutions that require installation of drivers and plugins for the hardware devices involved. Note that it is possible to incorporate the driver and the web browser plugin into the USB-token itself so that the installation process can be completely or partly automated.

4.2 Some Variants

A simplified variant: In the hPIN/hTAN system, the user has to input her PIN in a secure way to avoid exposing it to MitC attackers. The first security goal about PIN confidentiality may be relaxed, if the PIN is generated by the bank and cannot be changed by the user. This is a common policy for PINs of banking cards issued by German banks. Other security goals will not be compromised by potential leakage of the PIN to a MitC attacker, since they do not depend on the secure PIN-entry method at all. On the other hand, without physical access to the USB-token, the exposed PIN is useless to the MitC attacker.

Non-repudiation: The hPIN/hTAN system cannot provide the non-repudiation property because $h(K_T)$ is just a secret shared between the user/USB-token and the e-banking server. In case the non-repudiation property is important, we can add a digital signature function into the hPIN/hTAN system and generate H_4 in the hTAN protocol by a signature process instead of a hash process. The negative side effect is that this will definitely increase the implementation/deployment costs of the hPIN/hTAN system.

Transaction confidentiality: The hPIN/hTAN system does not protect confidentiality of transaction data. Usually, confidentiality "on the wire" is provided by standard security protocols such as SSL/TLS [29, 35]. If confidentiality towards a MitC is necessary, this function can be added to the hTAN protocol by asking the user to input STD in the same way as the PIN in the hPIN protocol. Then, the USB-token encrypts STD as follows: $E(\text{STD}) = \text{STD} \oplus h(r_e \parallel h(K_T) \parallel r_e)$, where r_e is an m-bit nonce. If STD has more than m bits, more nonces can be generated. All the nonces and the length of STD should be sent together with $E(\text{STD})$ to the server for decryption. This hash-based encryption is only used to protect STD, thus not leading to an increase of implementation costs.

Multi-bank and ATM/POS support: The USB-token only needs to store three values as user-specific data, so it is not difficult to store data for multiple e-banking systems. Then, the user can use a single USB-token to access all her e-banking systems, which can drastically improve user-friendliness of the whole e-banking service. By equipping ATMs/POSs with USB-ports, the USB-token can also be used as a replacement of banking cards. This eans that the whole e-banking/e-commerce chain can be implemented based on such USB-tokens.

5 Related Work

As we mentioned in Sec. 1, transaction authentication is the key measure against MitC attacks. There are mainly two approaches to achieve transaction authentication: 1) secure input and transmission of transaction data from the user U to the e-banking server S; 2) secure feedback from the server S to the user U for re-confirmation. To facilitate our discussion, we call the two approaches Sec-U2S and Sec-S2U, respectively. In the following, we first give a brief survey of existing solutions, which are classified into different categories according to the key secure mechanism used for input/transmission of transaction data. Then, we discuss some common problems with existing solutions and show how hPIN/hTAN outperforms other solutions.

5.1 Existing Solutions

Solutions based on trusted out-of-band (OOB) channel: The mTAN system [50] and some other solutions [16,48,57] depend on a trusted OOB channel to securely transmit the transaction data (and also the transaction-dependent TAN for the Sec-S2U approach). Apparently, the only OOB channel available to most users is the cellular network. Needless to say, solutions of this kind require the use of smart phones as trusted hardware devices.

Solutions based on CAPTCHAs: In some solutions, CAPTCHA images are used as trusted channels for transmitting transaction data from U to S [61, Solution 2] or from S to U [27, 60, 64]. The security is based on the assumption that a CAPTCHA image cannot be manipulated by the untrusted computer C in a real-time manner. A variant of iTAN, namely iTANplus, is based on CATPCHAs and has been deployed by some German banks [64].

Solutions based on encrypted Sec-S2U channel: Instead of using a trusted OOB channel, the secure data transmission in the Sec-S2U approach can also be realized though an encrypted channel. The key is shared between U and S in advance or established in real-time via a key exchange protocol. The security is guaranteed by the fact that the untrusted computer C has no access to the secret key. Since the human user is unable to perform complicated decryption operations, an additional trusted hardware device is needed to receive and decrypt the encrypted data from the server. Different kinds of hardware devices have been used in different solutions, such as smart phones [14, 24, 25, 44], PDAs [23], and special-purpose devices like personal AXS-tokens (also called Internetpass) [4] and "Sichere Fenster" ("Security Window" in English) smart card reader [17]. Some solutions [4, 14, 24, 25] use a special "visual channel" for secure data transmission from the untrusted computer to the hardware devices: encrypted data from the server are shown on the untrusted computer screen as an optical pattern such as a 2-D bar code, and the trusted hardware device directly reads the optical pattern from the computer screen and decodes/decrypts the data. It deserves mention that visual cryptography and grille cipher can also be used to encrypt the data sent from S to U, and then the user U decrypts the data by covering a transparency/grille on the computer screen [18, 20, 46], where the

transparency/grille plays the role of a low-tech trusted hardware device at the user's disposal.

Solutions based on hardware-generated transaction-dependent TANs: For the Sec-U2S approach, one way to realize secure transmission of the transaction data from U to S is to generate MACs or digital signatures as transaction-dependent TANs, which are sent together with the transaction data to the server. Normally, the calculation of MACs or the signing process is done by a trusted smart card reader equipped with a secure keypad and a secure display [8, 22, 52, 65, 66, 72].[8] The user inserts a smart card into the card reader to provide the key. Some solutions, such as Sm@rtTAN plus [66] and chipTAN [8], use offline smart card readers. The user needs to repeatedly input the transaction data on the secure keypad, and then presses a button to get the TAN. Such offline smart card readers can be enhanced with optical sensors to automatically read the transaction data from the computer screen, so the repeated input of transaction data is avoided [52,65]. Some smart card readers can be connected to the untrusted computer via wired link such as USB-connection, then the transaction data can be entered on the untrusted computer's keyboard and shown on a secure display of the smart card reader for user confirmation. After the user verifies the transaction data, she inputs a PIN on the secure keypad of the smart card reader to signal the signing process. Such online smart card readers include FINREAD [22] and HBCI-3 [72] smart card readers. In addition to smart card readers, in the QR-TAN system proposed in [59], the user's smart phone is used to calculate the MAC, and the key is stored in the smart phone. QR-TAN also uses QR 2-D bar code to avoid repeated input of transaction data on the smart phone. Another two solutions are "what-you-see-is-what-you-sign" [53] and "what-you-see-is-what-you-confirm" [19], which use camera phones to optically read the transaction data from the computer screen and then signs them or generates TANs from them.

Solutions based on manually-calculated transaction-dependent TANs: While hardware devices can be used to calculate transaction-dependent TANs, there are also some solutions that allow the human user to manually calculate the transaction-dependent TANs. A typical solution is Solution 1 proposed in [61]: a paper code book is shared between S and U, and for each online transaction the user U manually calculates the TAN from the transaction data by applying a code selected by the server S. Some similar solutions based on paper lists of secrets are also proposed in [9, 10].

Solutions based on encrypted Sec-U2S channel: Instead of sending a transaction-dependent TAN with the transaction ata to the server S, the transaction data can also be simply encrypted and sent to the server S without a transaction-dependent TAN. IBM ZTIC (Zurich/Zone Trusted Information

[8] Class-1 smart card readers (without both a secure keypad and a display) are also being used by some financial institutes, but they cannot resist MitM (and thus also MitC) attacks as shown in [31]. Class-2 smart card readers (without a secure keypad) are not secure against MitC attacks, since the PIN entry is done with the untrusted keyboard and thus can be easily stolen and replayed by the MitC attacker.

Channel) [34, 68] is a solution of this kind. ZTIC is based on a USB-token equipped with a TLS engine and an HTTP parser. The USB-token stores both client and server certificates, enabling TLS with client authentication. All communications between the client and the server are thus encrypted by the TLS engine, and the user is asked to press an "OK" button on the USB-token to approve each online transaction (or press another "Cancel" button to abort). Another low-tech solution called pPIN/pTAN [13, 15] uses a secret permutation mapping $\mathcal{G} : \{0, \ldots, 9\} \to \{0, \ldots, 9\}$ to conceal (encrypt) the PIN and transaction data from MitC attackers. The secret permutation mapping is chosen by the server from a paper list of n such mappings, which is issued to each user by the financial institute in advance. There are also some more complicated variants of pTAN scheme, such as 3pTAN [12] and ppTAN [11], but their usability is worse than pTAN.

Solutions based on trusted proxies: Some solutions base the security on trusted proxies [23, 28, 41, 54]. Normally the user and the trusted proxy share some secret information so that the user can input transaction data in a way secure against the MitC attacks. For instance, they may share a secret transform for secure input of the transaction data.

Solutions based on virtual machines: The SpyBlock solution proposed in [39] employs an authentication agent in a trusted host OS, and runs all user-level applications in an untrusted virtual machine. The trusted authentication agent takes care of secure input and transmission of transaction data to the server.

5.2 Problems with Existing Solutions

Although a lot of existing solutions have been proposed and some have been deployed by financial institutes, all of them have some nontrivial problems.

Among all the existing solutions, the simplest one is the CAPTCHA-based solution proposed in [61], which has no any special requirement on the user side. However, the security of CAPTCHA-based solutions is doubtful and prone to future attacks, since many existing CAPTCHA schemes have been broken [30, 33, 45, 70, 71]. A recent paper [42] reports that almost all e-banking CAPTCHA schemes are not secure against automated MitM attacks. In addition, human-assisted attacks may always be used to circumvent e-banking CAPTCHAs [6, 42].

SpyBlock [39] does not have any special requirement on hardware on the user side, but it requires installation of a trusted host OS and a virtual machine on the user's computer. Such a change of operating systems may be too heavy and not practical for some applications. In addition, the trusted host OS itself may also be compromised, thus rendering the authentication agent untrusted.

Solutions based on paper lists, transparencies or Cardano grilles [9–13, 15, 18, 20, 46, 61] have a major advantage: the implementation costs are relatively low compared with other hardware-based solutions. However, these solutions often require the user to perform some mental computations or align a transparency/grille with the computer screen, thus reducing the usability. In addition, paper lists, transparencies and Cardano grilles are often less portable than small hardware devices such as smart phones and USB-tokens. This problem becomes

even worse when the user has to bring many of such paper lists, transparencies and Cardano grilles [18,20,61]. Furthermore, when a user wants to make a large number of online transactions in a short period of time, all the available paper lists, transparencies or Cardano grilles may be quickly consumed, leading to a denial of service. Therefore, using hardware devices becomes a preferable choice for many users.

To save implementation costs, many hardware-based solutions [14,16,19,23–25,41,44,48,50,53,57,59] use general-purpose personal computing devices (i.e., smart phones and PDAs) as trusted hardware devices. The most severe problem with such general-purpose hardware devices is the potential risks of being infected by mobile malware [62], which can render the trusted devices untrusted. What's worse, in order to circumvent the language or functionality limits set by the manufacturers or the service providers, many users are accustomed to sending their smart phones and PDAs out to some private companies or alleged professionals to update the firmware/OS and/or install some additional programs, which makes it very easy for some attackers to inject any malicious code into a large number of devices. In addition, as we point out for mTAN in Sec. 1, the high complexity of general-purpose hardware devices leads to a higher risk of having software bugs and security holes. If dependency on the cellular network is involved, then other weaknesses of mTAN – unavailability and insecurity of cellular network, SIM card swop frauds and insider attackers from the telecommunication service providers – will also be major problems.

In addition to smart phones and PDAs, other trusted hardware devices used in solutions against MitC attacks include smart card readers [8,17,22,52,65,66,72], USB-tokens [34,68] and special-purpose devices like personal AXS-token [4]. All the smart card readers have a secure keypad as an essential component against MitC attacks. In addition, a smart card reader cannot work without a smart card. These features of smart card readers not only increase the costs, but also reduce the portability of the device (i.e., reduce the usability of the whole system). In addition, some smart card readers are also improperly optimized to cause security flaws, and the separation of the smart card and the reader leaves space for more potential risks [26]. The personal AXS-token does not have a secure keypad, but it is equipped with an optical interface for reading data from the computer screen and a biometric identification component, leading to a very expensive solution. Due to the need of maintaining an encrypted channel, all hardware devices without a secure keypad, including personal AXS-tokens, ZTIC USB-tokens and "Sichere Fenster" smart card reader [17], have a strong encryption module, which often increases the implementation costs of the whole solution.

Finally, solutions based on trusted proxies [23, 28, 41, 54] and the solutions proposed in [48, 57] are designed for making secure transactions on untrusted public terminal computers. In other words, these solutions work only when the user is aware of the untrustworthiness of the computer and intentionally choose to use these solutions. However, in our threat model, we assume that the user's private computer is infected by malware and become untrusted. In this case,

users are often unaware of the untrustworthiness of their computers, so they will not have the motivation to use these more complicated solutions for accessing e-banking servers. In addition, we believe that building a trusted proxy and maintaining its security are not trivial tasks for most common users.

5.3 Performance Comparison: hPIN/hTAN vs. Existing Solutions

The proposed hPIN/hTAN system is designed in a minimalistic way to reduce implementation costs without compromising security and usability. It uses a USB-token, not a smart phone or a PDA as the trusted hardware device, thus not suffering from the problems with general-purpose personal computing devices. Instead of using a keypad for secure input, human-involved interactive protocols are proposed to create a trusted path from the untrusted keyboard of the untrusted computer to the trusted device. We also intentionally make the hPIN and hTAN protocols independent of an additional trusted channel and strong encryption. Such a minimalistic design not only implies lower costs and better usability, but also leads to less software bugs and security holes.

Table 2 shows a brief comparison of hPIN/hTAN and some other selected hardware-based solutions. One can clearly see that the hPIN/hTAN system has the minimal hardware requirements, thus leading to minimal costs.

One may argue that hTAN is just like a simplified edition of IBM ZTIC. However, there are several key differences between hPIN/hTAN and IBM ZTIC. First, IBM ZTIC does not have any mechanism of PIN protection, which makes it vulnerable to loss and stealth. Second, hPIN/hTAN is an open framework and can work with any mutual authentication protocol (in hPIN part) and transaction authentication protocol (in hTAN part). As a matter of fact, we can also use TLS to implement the underlying protocols in hPIN and hTAN parts (i.e., hPIN/hTAN can be used to further enhance IBM ZTIC), but this just unnecessarily complicates the whole system. Third, one of the key features of hPIN/hTAN is the introduction of a trusted path from the user to the trusted token based on untrusted keyboard. This is achieved without major compromise of usability.

6 Conclusion

In this paper, we propose hPIN/hTAN, an enhanced PIN/TAN system based on a low-cost and easy-to-use USB-token, to protect e-banking systems from attacks related to untrusted computers, namely, man-in-the-computer (MitC) attacks. Our proposed system offers a better tradeoff between security and usability than existing solutions. The main feature of the system is the low complexity of the USB-token, which only needs to support a cryptographic hash function and some other simple functionalities. In addition, we carefully designed the protocols involved in the system to effectively exploit the human users' attention so that they will not be the weakest link in the system any more. Security analysis shows

Table 2. Comparison of hPIN/hTAN with selected hardware-based solutions.

	Smart phone/PDA	Secure keypad	Encryption	Data channel	External party	Smart card
hPIN/hTAN (this work)	No	No*	No	No	No	No
mTAN/mobileTAN [5, 50]	Yes	No	No	OOB	Yes	Yes
Sm@rtTAN plus [66]	No	Yes	No	No	No	Yes
Sm@rtTAN optic [65]	No	Yes	No	Optic	No	Yes
FINREAD [22], HBCI-3 [72]	No	Yes	Yes	USB	No	Yes
photoTAN [24], Fotohandy-TAN [14]	Yes	Yes	Yes	Optic	No	Yes
Sesam-Öffne-Dich ("Open Sesame") [16]	Yes	Yes	Yes	Optic	Yes	Yes
what-you-see-is-what-you-sign/confirm [19, 53]	Yes	Yes	Yes	Optic	No	No
QR-TAN [59]	Yes	Yes	Yes	Optic	No	No
ZTIC [68]	No	No*	Yes	USB	No	Yes/No**
"Sichere Fenster" [17]	No	No	Yes	USB	No	Yes
AXSionics [4]	No	No	Yes	Optic	Yes	No
MP-Auth [44]	Yes	Yes	Yes	Wireless	No	No

*: The hPIN/hTAN and the ZTIC systems do not require a secure keypad, but need one and two trusted buttons on the hardware device involved, respectively.
**: The ZTIC can be optionally equipped with a smart card reader to support smart card.

that hPIN/hTAN can achieve three security requirements: PIN confidentiality, user/server authenticity, and transaction authenticity/integrity.

We have developed a prototype system and performed a small-scale user study for demonstrating the usability of the hPIN/hTAN system. In the future we will investigate more variants of the basic design, and try to figure out if some variants have even better overall performance than the basic hPIN/hTAN system reported in this paper. For instance, we will study if the USB channel can be replaced by an optic or wireless one to enhance usability but with acceptable additional costs. We also plan to run further user studies to show the real performance of hPIN/hTAN for average bank customers.

Acknowledgments

Shujun Li and Junaid Jameel Ahmad were supported by the Zukunftskolleg of the University of Konstanz, Germany, as part of the "Excellence Initiative" Program of the German Research Foundation (DFG). Shujun Li and Roland Schmitz thank Prof. Walter Kriha of the Stuttgart Media University for valuable discussions and comments on an early draft of the paper. The authors also thank all participants of our user study running at the University of Konstanz and the Stuttgart Media University.

References

1. AlZomai, M., AlFayyadh, B., Jøsang, A., McCullagh, A.: An exprimental investigation of the usability of transaction authorization in online bank security systems. In: Proceedings of 6th Australasian Information Security Conference (AISC'2008). pp. 65–73 (2008)
2. American Bankers Association: ABA survey shows more consumers prefer online banking. http://www.aba.com/Press+Room/093010PreferredBankingMethod.htm (2010)
3. Arbeitsgruppe Identitätsschutz im Internet: A-I3 Pressemeldung: iTAN nur in Verbindung mit SSL sicher (Update). Web page (2005), https://www.a-i3.org/content/view/411/28
4. AXSionics AG: Personal AXS-token. Web page (2009), http://www.axsionics.ch/tce/frame/main/414.htm
5. Bank Austria: mobileTAN information. Web page (2007), http://www.bankaustria.at/de/19741.html
6. BBC News: PC stripper helps spam to spread. Web page (October 2007), http://news.bbc.co.uk/2/hi/technology/7067962.stm
7. Berliner Bank: Online-Banking mit indizierter TAN-Liste. Web page (2009), http://www.berliner-bank.de/bb/content/p_banking-sicherheit_internet_itan_details.html
8. Berliner Sparkasse: chipTAN: Listen werden überflüssig. Web page (2005), http://www.berliner-sparkasse.de/anzeigen.php?tpl=privatkunden/konten_karten/online_banking/tan_generator.html
9. Borchert, B.: Bild-TAN Verfahren. Web page (2009), http://www2-fs.informatik.uni-tuebingen.de/~borchert/Troja/BildTAN
10. Borchert, B.: Das indirekte iTAN Verfahren. Web page (2009), http://www2-fs.informatik.uni-tuebingen.de/~borchert/Troja/iiTAN
11. Borchert, B.: Das Positions-Permutations-TAN (ppTAN) Verfahren: Sicherung einer Überweisung gegen einen Man-in-the-Middle Angriff. Web page (2009), http://www2-fs.informatik.uni-tuebingen.de/~borchert/Troja/ppTAN
12. Borchert, B.: Die 3-Permutationen-TAN (3pTAN). Web page (2009), http://www2-fs.informatik.uni-tuebingen.de/~borchert/Troja/3pTAN
13. Borchert, B.: Die Knick-und-Klick-PIN, oder Permutations-PIN (pPIN). Web page (2009), http://www2-fs.informatik.uni-tuebingen.de/~borchert/Troja/pPIN
14. Borchert, B.: Fotohandy-TAN: Secure online banking via camera phone. Online document (January 2009), http://www2-fs.informatik.uni-tuebingen.de/studdipl/Fotohandy-PIN/Info-Blaetter/Info_E_Foto-TAN.pdf

15. Borchert, B.: Knick-und-Klick-TAN, oder Permutations-TAN (pTAN). Web page (2009), http://www2-fs.informatik.uni-tuebingen.de/~borchert/Troja/pTAN
16. Borchert, B.: Open sesame! – immediate access to online accounts via mobile camera phone. Web page (2009), http://www2-fs.informatik.uni-tuebingen.de/~borchert/Troja/Open-Sesame/indexEN.php
17. Borchert, B.: Sichere Fenster. Web page (2009), http://www2-fs.informatik.uni-tuebingen.de/~borchert/Troja/SichereFenster
18. Borchert, B.: Visuelle TAN. Web page (2009), http://www2-fs.informatik.uni-tuebingen.de/~borchert/Troja/VisuelleKryptografie
19. Borchert, B.: What-you-see-is-what-you-confirm fuer die Fotohandy-TAN: Ziffern-Erkennung mit dem Fotohandy. Web page (May 2009), http://www2-fs.informatik.uni-tuebingen.de/~borchert/Troja/what-you-see-is-what-you-confirm
20. Borchert, B., Beschke, S.: Cardano-TAN. Web page (2009), http://www2-fs.informatik.uni-tuebingen.de/studdipl/beschke
21. Bosselaers, A., Preneel, B.: SKID. In: Integrity Primitives for Secure Information Systems: Final Report of RACE Integrity Primitives Evaluation RIPE-RACE 1040, Lecture Notes in Computer Science, vol. 1007, chap. 6, pp. 169–178. Springer (1995)
22. CEN (Comité Européen de Normalisation, European Committee for Standardization): Financial transactional IC card reader (FINREAD). CEN Workshop Agreements (CWA) 14174 (January 2004), http://www.cen.eu/cenorm/sectors/sectors/isss/cen+workshop+agreements/finread.asp
23. Clarke, D., Gassend, B., Kotwal, T., Burnside, M., van Dijk, M., Devadas, S., Rivest, R.: The untrusted computer problem and camera-based authentication. In: Pervasive Computing (Proc. Pervasive'2002). Lecture Notes in Computer Science, vol. 2414, pp. 88–103. Springer-Verlag, Berlin Heidelberg (2002)
24. Cronto Limited: Commerzbank and Cronto launch secure online banking with photoTAN – World's first deployment of visual transaction signing mobile solution. Online document (November 2008), http://www.cronto.com/download/Cronto_Commerzbank_photoTAN.pdf
25. Cronto Limited: Cronto's visual cryptogram: A simple portable solution for enhanced security. Web page (2008), http://www.cronto.com/visual_cryptogram.htm
26. Drimer, S., Murdoch, S.J., Anderson, R.: Optimised to fail: Card readers for online banking. In: Financial Cryptography and Data Security (Proc. FC'2009). Lecture Notes in Computer Science, vol. 5628, pp. 184–200. Springer, Berlin / Heidelberg (2009)
27. Fischer, I., Herfet, T.: Visual CAPTCHAs for document authentication. In: Proceedings of IEEE 8th Workshop on Multimedia Signal Processing (MMSP'2006). pp. 471–474. IEEE (2006)
28. Florêncio, D., Herley, C.: KLASSP: Entering passwords on a spyware infected machine using a sharedsecret proxy. In: Proc. 22nd Annual Computer Security Applications Conference (ACSAC'2006). pp. 67–76. IEEE Computer Society (2006)
29. Freier, A.O., Karlton, P.L., Kocher, P.C.: The SSL protocol: Version 3.0. INTERNET-DRAFT (November 1996), http://www.mozilla.org/projects/security/pki/nss/ssl/draft302.txt
30. Golle, P.: Machine learning attacks against the Asirra CAPTCHA. In: Proc. 15th ACM Conference on Computer and Communications Security (CCS'2008). pp. 535–542. ACM (2008)

31. Gröbert, F., Wegener, C.: A proof-of-concept attack on SmartCard-secured online-banking. Accepted by the In-Depth Security Conference (DeepSec) Europe (2009), https://deepsec.net/docs/speaker.html#PSLOT05
32. Gühring, P.: Concepts against man-in-the-browser attacks. Online document (January 2007), http://www2.futureware.at/svn/sourcerer/CAcert/SecureClient.pdf
33. Hocevar, S.: PWNtcha: Pretend we're not a Turing computer but a human antagonist. Web page (2009), http://caca.zoy.org/wiki/PWNtcha
34. IBM: IBM Zone Trusted Information Channel (ZTIC): A banking server's display on your key chain. Web page (2008), http://www.zurich.ibm.com/ztic
35. IETF: The Transport Layer Security (TLS) protocol: Version 1.2. RFC 5246 (August 2008), http://tools.ietf.org/html/rfc5246
36. Initiative D21: Online-Banking – Mit Sicherheit!: Vertrauen und Sicherheitsbewusstsein bei Bankgeschäften im Internet. Web page (June 2008), http://www.fiducia.de/service/publikationen/nonliner_atlas_2008.html, (in German)
37. IT-Online: World-first SMS banking scam exposes weaknesses. Web page (July 2009), http://www.it-online.co.za/content/view/1092105/142/
38. Jackson, C., Boneh, D., Mitchell, J.: Transaction generators: Root kits for web. In: Proc. 2nd USENIX Workshop on Hot Topics in Security (HotSec'2007). pp. 1–4. USENIX Association (2007)
39. Jackson, C., Boneh, D., Mitchell, J.C.: Spyware resistant web authentication using virtual machines. Technical Report (2006), http://crypto.stanford.edu/SpyBlock/spyblock-2.pdf
40. Jakobsson, M., Myers, S. (eds.): Phishing and Countermeasures: Understanding the Increasing Problem of Electronic Identity Theft. John Wiley & Sons, Inc. (January 2007)
41. Jammalamadaka, R.C., van der Horst, T.W.: Delegate: A proxy based architecture for secure website access from an untrusted machine. In: Proc. 22nd Annual Computer Security Applications Conference (ACSAC'2006). pp. 57–66. IEEE Computer Society (2006)
42. Li, S., Shah, S.A.H., Khan, M.A.U., Khayam, S.A., Sadeghi, A.R., Schmitz, R.: Breaking e-banking CAPTCHAs. In: Proceedings of 26th Annual Computer Security Applications Conference (ACSAC'2010). pp. 171–180. ACM (2010), http://www.hooklee.com/default.asp?t=eBankingCAPTCHAs
43. Li, S., Shum, H.Y.: Secure human-computer identification against peeping attacks (SecHCI): A survey. Technical report (2003), http://www.hooklee.com/Papers/SecHCI-Survey.pdf
44. Mannan, M., van Oorschot, P.: Using a personal device to strengthen password authentication from an untrusted computer. In: Financial Cryptography and Data Security (Proc. FC'2007). Lecture Notes in Computer Science, vol. 4886, pp. 88–103. Springer-Verlag, Berlin Heidelberg (2007)
45. Mori, G., Malik, J.: Recognizing objects in adversarial clutter: Breaking a visual CAPTCHA. In: Proc. 2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'2003). vol. 1, pp. 134–141. IEEE Computer Society (2003)
46. Naor, M., Pinkas, B.: Visual authentication and identification. In: Advances in Cryptology – CRYPTO'97. Lecture Notes in Computer Sciences, vol. 1294, pp. 322–336. Springer-Verlag, Berlin Heidelberg (1997)
47. Oppliger, R., Rytz, R., Holderegger, T.: Internet banking: Client-side attacks and protection mechanisms. Computer 42(6), 27–33 (June 2009)

48. Oprea, A., Balfanz, D., Durfee, G., Smetters, D.: Securing a remote terminal application with a mobile trusted device. In: Proc. 20th Annual Computer Security Applications Conference (ACSAC'2004). pp. 438–447. IEEE Compter Society (2004)
49. PC World: Nokia: We don't know why criminals want our old phones. Web page (April 2009), http://www.pcworld.com/businesscenter/article/163515/nokia_we_dont_know_why_criminals_want_our_old_phones.html
50. Postbank: mTAN now free for all customers. Web page (May 2008), http://www.postbank.com/pbcom_ag_home/pbcom_pr_press/pbcom_pr_press_archives/pbcom_pr_press_archives_2008/pbcom_pr_pm1063_19_05_08.html
51. RedTeam Pentesting GmbH: New banking security system iTAN not as secure as claimed. Advisory rt-sa-2005-014 (August 2005), http://www.redteam-pentesting.de/advisories/rt-sa-2005-014
52. Reiner SCT: Demo chipTAN comfort. Web page (2009), http://support.reiner-sct.de/downloads/flashdemo/tanJack_optic/1024/tanJack_optic.html
53. Röhrich, J.M.Q.S.: What you see is what you sign. Presented at the Heidelberger Innovationsforum (November 2007), http://www.heidelberger-innovationsforum.de/fileadmin/_heidelberger/downloads/Praesentationen_Nov07/45_Roehrich.pdf
54. Ross, S.J., Hill, J.L., Chen, M.Y., Joseph, A.D., Culler, D.E., Brewer, E.A.: A composable framework for secure multi-modal access to internet services from post-PC devices. In: Proc. 3rd IEEE Workshop on Mobile Computing Systems and Applications (WMCSA'2000). pp. 171–182. IEEE Computer Society (2000)
55. Saturday Star: Victim's SIM swop fraud nightmare. Web page (January 12 2008), http://www.iol.co.za/index.php?art_id=vn20080112083836189C511499
56. Schneier, B.: Two-factor authentication: Too little, too late. Communications of the ACM 48(4), 136 (April 2005)
57. Sharp, R., Scott, J., Beresford, A.R.: Secure mobile computing via public terminals. In: Pervasive Computing (Proc. PERVASIVE'2006). Lecture Notes in Computer Science, vol. 3968, pp. 238–253. Springer-Verlag, Berlin Heidelberg (2006)
58. Sparkasse Heidelberg: Online-banking: Increased security with iTAN. Web page (2006), http://www.sparkasse-heidelberg.com/spk-hd_en/pk/banking/itan.html
59. Starnberger, G., Froihofer, L., Goeschka, K.M.: QR-TAN: Secure mobile transaction authentication. In: Proc. 2009 International Conference on Availability, Reliability and Security (ARES'2009). pp. 578–583. IEEE Computer Society (2009)
60. Steeves, D.J., Snyder, M.W.: Secure online transactions using a CAPTCHA image as a watermark. Patent US 2007/0005500 A1 (January 2007)
61. Szydlowski, M., Kruegel, C., Kirda, E.: Secure input for web applications. In: Proc. 23rd Annual Computer Security Applications Conference (ACSAC'2007). pp. 375–384. IEEE Computer Society (2007)
62. The Financial Express: Russian phone virus that 'steals money' may spread global. Web page (February 2009), http://www.financialexpress.com/news/russian-phone-virus-that-steals-money-may-spread-global/420770
63. Toorani, M., Shirazi, A.A.B.: Solutions to the GSM security weaknesses. In: Proceedings of 2nd International Conference on Next Generation Mobile Applications, Services, and Technologies (NGMAST'2008). pp. 576–581. IEEE Computer Society (2008)

64. Volksbank Freiburg eG: iTANplus – mehr sicherheit mit der indizierten TAN. Web page (2009), http://www.volksbank-freiburg.de/itan.cfm?CFID=10869033&CFTOKEN=34249989&rand=1246061956151
65. Volksbank Rhein-Ruhr eG: Bankgeschäfte online abwickeln: Mit Sm@rtTAN optic bequem und sicher im Netz. Web page (2009), http://www.voba-rhein-ruhr.de/privatkunden/ebank/SMTop.html
66. Volksbank Solling eG: Sm@rt-TAN-plus. Web page (2009), http://www.volksbank-solling.de/flycms/de/html/913/-/Smart+TAN+plus.html
67. Wagner, D., Schneier, B.: Analysis of the SSL 3.0 protocol. In: Proceedings of the Second USENIX Workshop on Electronic Commerce. pp. 29–40. USENIX Association (1996)
68. Weigold, T., Kramp, T., Hermann, R., Höring, F., Buhler, P., Baentsch, M.: The Zurich Trusted Information Channel – An efficient defence against man-in-the-middle and malicious software attacks. In: Trusted Computing – Challenges and Applications (Proc. TRUST'2008). Lecture Notes in Computer Science, vol. 4968, pp. 75–91. Springer-Verlag, Berlin Heidelberg (2008)
69. Wikipedia: Transaction authentication number. Web page (August 2009), http://en.wikipedia.org/wiki/Transaction_authentication_number
70. Yan, J., Ahmad, A.S.E.: Breaking visual CAPTCHAs with naïve pattern recognition algorithms. In: Proc. 23rd Annual Computer Security Applications Conference (ACSAC'2007). pp. 279–291. IEEE Computer Society (2007)
71. Yan, J., Ahmad, A.S.E.: A low-cost attack on a Microsoft CAPTCHA. In: Proc. 15th ACM Conference on Computer and Communications Security (CCS'2008). pp. 543–554. ACM (2008)
72. ZKA (Zentralen Kreditausschuss): FinTS – financial transaction services. Web page (July 2004), http://www.hbci-zka.de/spec/4_0.htm