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

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

[1] University of Konstanz, Germany
[2] Darmstadt University of Technology and Fraunhofer SIT, Darmstadt, Germany
[3] 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/screenloggers, 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 and a random number generator).

In contrast to other existing 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 not only enhances usability but also increases security since more complicated systems tend to have more security holes and software bugs. As an important feature, hPIN/hTAN exploits the human user's active involvement in the whole process to compensate security weaknesses caused by careless human behavior.

1 Introduction

Due to the rapid growth of the Internet, e-banking becomes more and more popular all over the world. A 2009 survey of the American Bankers Association showed that e-banking has been the preferred banking method of bank customers [2]. It is not surprising that most (more than 90% according to a recent survey made in Germany [17]) users consider security as the most important issue about e-banking. The earliest and simplest defense protecting e-banking systems from attacks is user authentication based on static PINs (Personal Identification Numbers). 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 [14, 16].

While SSL/TLS is considered secure [36], the static PINs are prone to identity theft based on social engineering attacks, in which the users are spoofed to hand

* Part of this work (without the hardware prototype) was presented at FC 2010 as a poster. A full edition of this paper will be available as an IACR eprint soon.

over their PINs. One of the most prevailing social engineering attacks to e-banking systems is phishing attack [20]. In its simplest form the attacker (called phisher) sends phishing emails to lure gullible users to expose their PINs on a bogus e-banking web site. Once the phisher gets the fixed 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 Access Numbers) for online transactions [38].

While PIN/TAN (and its variants like PIN/iTAN [6]) can reduce the risk of simple social-engineering attacks like email based phishing, 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 connection 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 a real-time manner 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.

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. There are different kinds of malware. Some malware can inject malicious code into the web browser of the user's computer, 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 [15]. 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, including malware-based attack or malicious software attack [37], Trojan attacks [28], content-manipulation attacks [24], transaction generator attack [19], and so forth.

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), which has been deployed by many financial institutions around the world [4, 26]. 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 mobile phone is not always available at the user side. Furthermore, the cellular network is not free from potential attacks [32]. In addition, the user's mobile phone may also be infected by malware [31] and is still prone to some more advanced attacks such as SIM card swop frauds [27] and insider attacks from the telecommunication service providers [18]. The high complexity of today's smart phones may also lead to potential security holes induced by software bugs [25].

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 nontrivial 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 and deployment issues 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.

2.1 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

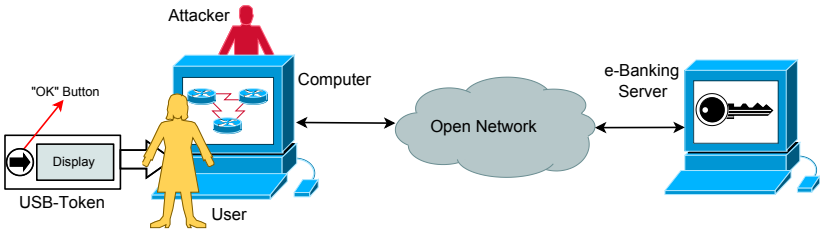

Fig. 1. The threat model of the hPIN/hTAN system.

computer C, tries to access the remote e-banking server S and then makes some online transactions. We assume that the e-banking server S is trusted, which is a reasonable assumption in practice. 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.

2.2 Notations

The notations used in this paper are summarized in the following table.

IDU	The ID of the user U.
K_T	A secret key shared between T and S.
PIN	An n-character PIN shared between U and T.
PIN(i)	The i-th character of PIN.
s	A salt used to be hashed together with PIN.
STD	Sensitive transaction data that are authenticated.
NSTD	Non-sensitive transaction data that are not authenticated.
$h(\cdot)$	An 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).
$\mathcal{F}_i : \mathbb{X} \to \mathbb{Y}$	A 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	The maximal numbers of consecutive failures allowed by T and S.

2.3 Requirements Analysis

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;

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 ensure a low complexity of the system.

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:

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

where C_{T} is used to signal locking the USB-token if more than V_{T} wrong PINs are input 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.

2.4 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 [10], 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 we will show that the simple SKID3 protocol is already sufficient 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.

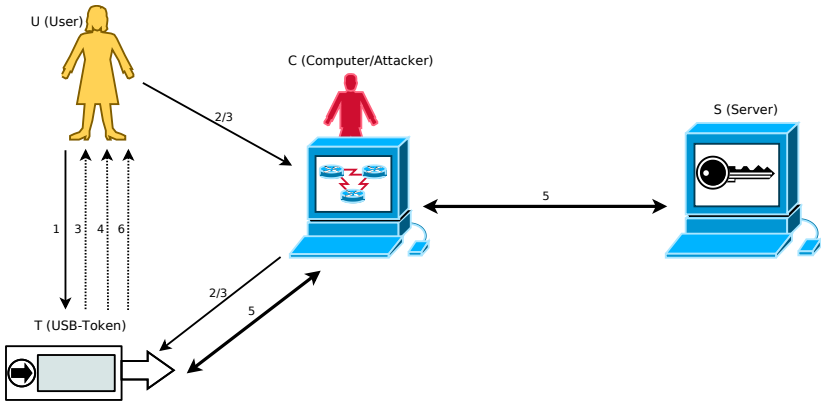

Fig. 2. The hPIN part, where solid lines denote real interactions/communications and dashed lines denote information display (the same hereinafter). The thick solid lines highlight the reconfigurable mutual authentication protocol.

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 $\mathcal{F}_i(\mathrm{PIN}(i))$ 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 $\mathcal{F}_i(\mathrm{PIN}(i))$ 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 $\mathrm{PIN}^* = h(\mathrm{PIN} \parallel (K_T^* \oplus h(\mathrm{PIN} \parallel s)) \parallel s)$. If so, then T recovers the secret key as $K_T = K_T^* \oplus h(\mathrm{PIN} \parallel s)$, stores $h(K_T)$ in its volatile memory for future use in the hTAN part, 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.

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:

$\mathsf{T} \to \mathsf{S}$: UID,

$\mathsf{S} \to \mathsf{T}$: r_T,

$\mathsf{T} \to \mathsf{S}$: (UID, r_S, $H_1 = \mathrm{HMAC}(K_T, r_\mathsf{S} \parallel r_\mathsf{T} \parallel \mathsf{T})$),

$\mathsf{S} \to \mathsf{T}$: $H_2 = \mathrm{HMAC}(K_T, r_\mathsf{T} \parallel r_\mathsf{S} \parallel \mathsf{S})$,

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

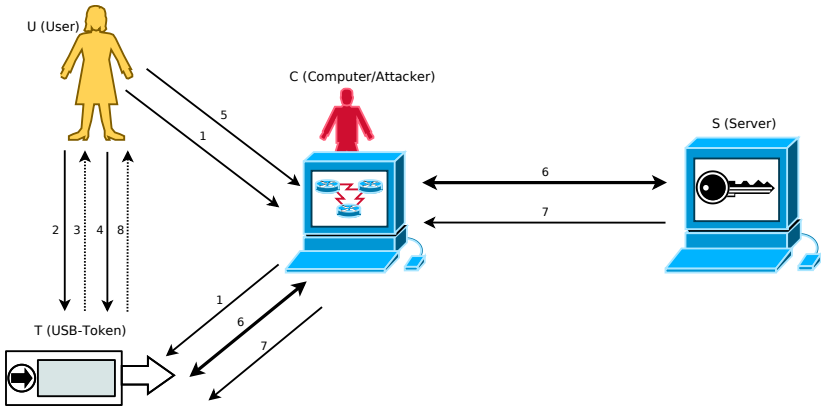

Fig. 3. The hTAN protocol. The thick solid lines highlight the reconfigurable transaction/message authentication protocol.

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*.

2.5 The hTAN Part

The hTAN part protects online transactions from MitC attacks after the user has successfully logged into the e-banking system by passing 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 which 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 HMAC scheme involved in the hPIN part for construct the transaction authentication protocol, so that the whole hPIN/hTAN system is based on the same hash function $h(\cdot)$ and the same 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. 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 as an asterisk "*" 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.[4] If U presses "Backspace" key, T shows an eye-catching

[4] 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.

warning message to inform the user for a few seconds and then the previously entered character is deleted. The goal of such a special design on "Backspace" key is explained later in Sec. 3.3.

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

Step 3: T highlights 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 Tby 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:
$\mathsf{T} \rightarrow \mathsf{S}$: (IDU, STD, NSTD, r_{T}^*),
$\mathsf{S} \rightarrow \mathsf{T}$: ($r_{\mathsf{S}}^*$, $H_3 = \text{HMAC}(K_T, r_{\mathsf{T}}^* \parallel r_{\mathsf{S}}^* \parallel \text{STD})$,
$\mathsf{T} \rightarrow \mathsf{S}$: (IDU, $H_4 = \text{HMAC}(K_T, r_{\mathsf{S}}^* \parallel r_{\mathsf{T}}^* \parallel \text{STD})$,
where r_{T}^* and r_{S}^* are two new nonces generated by T and S, respectively.

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

Step 8: T checks if $H_5 = \text{HMAC}(K_T, r_{\mathsf{T}}^* \parallel r_{\mathsf{S}}^* \parallel$ "success" $\parallel$ 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 at least part of it such as the birthday) 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 $\mathcal{F}_1, \ldots, \mathcal{F}_n$ 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 $\mathcal{F}_1(\text{PIN}(1)), \ldots, \mathcal{F}_n(\text{PIN}(n))$. Each PIN character is mapped to a printable character by a *di erent* 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 for the attacker, since 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*.

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, the two related security requirements are guaranteed because the attacker can only passively forward the 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 the communications between U (i.e., T) and S, since the user's input on the untrusted computer C has already been leaked to the attacker.

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 the USB-token T and the server S 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 in the following.

The human-token-computer interactive protocol (Steps 1–4) ensures that T gets the STD without being manipulated by the attacker. Since the user has to look at T's display to verify her input and then press the "OK" button twice to show confirmation, we can ensure that T always receives 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 the STD. As long as the whole STD is correct, there is no risk that it is manipulated by the attacker. This is how *simultaneous* STD input and verification is achieved. The main goal of the special design on "Backspace" key is to significantly prolong the time of deleting previously entered characters and to alert U so that malicious deletion of STD characters by C can be detected 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 (if not completely) 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 the correct STD from the user, it will try to send STD to S for execution. The transaction authentication and re-confirmation is ensured by the two HMAC values H_3 and H_4, which are both STD-dependent. 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 the risk of replay attack 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.

4 Usability of hPIN/hTAN

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. A virtual e-banking web site is implemented to simulate the role of a genuine e-banking server. Figure 4 shows a prototype USB token running Step 3 of the hPIN stage.

A quick user study with 9 computer science students has shown that with a 4-digit PIN the average login time is around 40 seconds and a typical hTAN process can be

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

done in around one minute. The login time counts the whole hPIN process, not only the PIN entry part. Despite the relatively long average login time, which is partly due to the fact that the students used the system for the first time, the students did rate the login process quite positively (the average rating for the hPIN process was 3.56 on a five point scale). We expect the login time will reduce significantly after they become more familiar with both the system and their PINs. A survey of the participants showed that all students have no trouble understanding how the system works. A longer-term and large-scale user study will be organized at our universities and the results will be reported in the final edition of the paper. 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 list all the possible PIN characters and the second shows the corresponding code of each PIN character. This will allow the user to simply look for her current PIN character $\mathrm{PIN}(i)$ and then input the character below $\mathrm{PIN}(i)$. With a list of codewords as shown above, we expect that 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 will not consider this step in the following. 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. Therefore, we expect an average user will be able to input the STD. Steps 2–4 are very easy because the user either just waits and observes or simply presses a button on T. We expect that 2–4 seconds should be enough for an average user to finish all

the steps. 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 may consume more time than the hTAN process.

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.

5 Related Work

As we mentioned in Sec. 1, transaction authentication is the key measure against MitC attacks, which can be achieved through two main approaches: 1) secure input and transmission of transaction data from U to S; 2) secure feedback from S to U for re-confirmation. In this section, we briefly overview previous solutions.

The first approach can be realized by transmitting the transaction data from U to S though an encrypted channel. For instance, in IBM ZTIC [37], a USB-token is used to establish a TLS channel for encrypting all communications between U and S. The USB-token has a trusted display and two buttons so that the user can explicitly confirm or cancel the transaction data. A low-tech solution called pTAN [7] is based on a paper list of secret permutation mappings, one of which is used for each transaction to conceal (encrypt) the input of transaction data from MitC attackers. Some other solutions are based on transaction-dependent TANs sent together with the transaction data to ensure transaction integrity. The TAN can be a MAC or a digital signature of the transaction data. A hardware device equipped with a secret key, such as a smart card reader [11, 34, 35, 39] or a mobile phone [29], is normally needed to calculate the TAN. To ensure that the TAN is calculated from correct transaction data, either a trusted keypad is necessary or the trusted hardware device reads the transaction data from the computer screen optically. The user can also manually calculate the TAN, as proposed in [30], under the help of a large paper codebook.

The second approach requires a trusted channel for U to receive the feedback from S. Some solutions use an out-of-band (OOB) channel like the cellular network [8, 26]. Other solutions use an encrypted channel for this purpose. Different kinds of hardware devices are used for decrypting data sent from S, including mobile phones [12, 22] and special-purpose devices like personal AXS-tokens [3]. Some solutions [3, 12] also support direct readout of encrypted data from the computer screen. Visual cryptography and grille cipher are also used for this purpose [9, 23].

Among all the existing solutions, the simplest ones are based on CAPTCHAs [30,33], which use CAPTCHA images as a virtual channel that cannot be handled by automated MitC attackers. However, [21] shows 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 [5, 21].

Solutions based on low-tech "hardware" such as paper lists [7,9,23,30] have a major advantage: the implementation costs are relatively low compared with solutions based on electronic devices. However, these solutions often require the user to perform mental computations and/or align a paper sheet/transparency/grille with the computer screen, thus reducing the usability. In addition, low-tech "devices" are often less portable than small electronic devices. This problem becomes even worse when the user has to bring more than one such low-tech "device" [9,30]. Furthermore, when a user wants to make a large number of online transactions in a short period of time, low-tech "devices" can be quickly used up, leading to a denial of service. Therefore, using electronic devices becomes a preferable choice for many users.

To save implementation costs, many solutions based on electronic devices use mobile phones and PDAs as trusted hardware devices [8, 12, 22, 26, 29]. The most severe problem with such general-purpose electronic devices is the potential risks of being infected by mobile malware [31]. Even worse, in order to circumvent the language or functionality limits set by the manufacturers or the service providers, many users are willing to send their mobile phones/PDAs to some private companies or alleged professionals to update the firmware/OS, 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 mobile phones and PDAs 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 will also be major problems.

In addition to mobile phones and PDAs, other trusted hardware devices used against MitC attacks include smart card readers [11, 34, 35, 39], USB-tokens [37] and special-purpose devices like personal AXS-tokens [3]. All the smart card readers have a secure keypad as n essential component against MitC attacks. This not only increases the costs, but also reduces the portability of the device. 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 [13]. The personal AXS-tokens do not have secure keypads, but are equipped with optical interfaces for reading data from the computer screen and biometric identification components, leading to a very expensive solution. Due to the need of maintaining an encrypted channel, all electronic devices without a secure keypad have a strong encryption module, which also increases the implementation costs of the whole solution.

In comparison with other existing solutions, hPIN/hTAN is designed to reduce implementation costs without compromising security and usability. It uses a USB-token as the trusted hardware device, so it does not suffer from the problems with mobile phones and PDAs. Instead of using a keypad for secure input, we propose human-involved interactive protocols to create a trusted path from the untrusted keyboard of the untrusted computer to the trusted device. We also intentionally make hPIN/hTAN independent of an additional trusted channel and strong encryption. Such a minimalistic design not only leads to lower costs and better usability, but also to less software bugs and security holes. Table 1 shows a comparison of hPIN/hTAN and some selected hardware-based solutions. One can clearly see that hPIN/hTAN has the minimal hardware requirements, thus leading to minimal costs.

Table 1. Comparison of hPIN/hTAN with selected hardware-based solutions.

	Mobile phone or PDA	Secure keypad	Encryption	Optical component	External dependency	Smart card
hPIN/hTAN	No	No*	No	No	No	No
mTAN [4, 26]	Yes	No	No	No	Yes	Yes
Sm@rtTAN plus [35]	No	Yes	No	No	No	Yes
Sm@rtTAN optic [34]	No	Yes	No	Yes	No	Yes
FINREAD/HBCI-3 [11, 39]	No	Yes	Yes	No	No	Yes
photoTAN [12]	Yes	Yes	Yes	Yes	No	No
"Open Sesame" [8]	Yes	Yes	Yes	Yes	Yes	Yes
QR-TAN [29]	Yes	Yes	Yes	Yes	No	No
IBM ZTIC [37]	No	No*	Yes	No	No	Optional
AXSionics [3]	No	No	Yes	Yes	Yes	No
MP-Auth [22]	Yes	Yes	Yes	No	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.

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

We have developed a prototype system for demonstrating the usability of the hPIN/hTAN system. A first small-scale user study has been done to show the feasibility of hPIN/hTAN solution. A long-term and larger-scale user study is under way and the results will be reported in the final edition of this paper. 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.

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: Proc. AISC'2008. pp. 65–73
2. American Bankers Association: ABA survey: Consumers prefer online banking. `http://www.aba.com/Press+Room/092109ConsumerSurveyPBM.htm` (2009)
3. AXSionics AG: Personal AXS-token. Web page (2009), `http://www.axsionics.ch/tce/frame/main/414.htm`
4. Bank Austria: mobileTAN information. Web page, `http://www.bankaustria.at/de/19741.html`
5. BBC News: PC stripper helps spam to spread. Web page (2007), `http://news.bbc.co.uk/2/hi/technology/7067962.stm`
6. Berliner Bank: Online-Banking mit indizierter TAN-Liste. Web page, `http://www.berliner-bank.de/bb/content/p_banking-sicherheit_internet_itan_details.html`
7. Borchert, B.: Knick-und-Klick-TAN, oder Permutations-TAN (pTAN). Web page (2009), `http://www2-fs.informatik.uni-tuebingen.de/~borchert/Troja/pTAN`
8. 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`
9. Borchert, B., Beschke, S.: Cardano-TAN. Web page (2009), `http://www2-fs.informatik.uni-tuebingen.de/studdipl/beschke`
10. Bosselaers, A., Preneel, B.: SKID. In: Integrity Primitives for Secure Information Systems: Final Report of RACE Integrity Primitives Evaluation RIPE-RACE 1040, LNCS, vol. 1007, chap. 6, pp. 169–178. Springer (1995)
11. CEN (European Committee for Standardization): Financial transactional IC card reader (FINREAD). CEN Workshop Agreements (CWA) 14174 (2004)
12. Cronto Limited: Commerzbank and Cronto launch secure online banking with photoTAN – World's first deployment of visual transaction signing mobile solution. Online document (2008), `http://www.cronto.com/download/Cronto_Commerzbank_photoTAN.pdf`
13. Drimer, S., Murdoch, S.J., Anderson, R.: Optimised to fail: Card readers for online banking. In: Financial Cryptography and Data Security (Proc. FC'2009). LNCS, vol. 5628, pp. 184–200. Springer (2009)
14. Freier, A.O., Karlton, P.L., Kocher, P.C.: The SSL protocol: Version 3.0. INTERNET-DRAFT (1996), `http://www.mozilla.org/projects/security/pki/nss/ssl/draft302.txt`
15. Gühring, P.: Concepts against man-in-the-browser attacks. Online document (2007), `http://www2.futureware.at/svn/sourcerer/CAcert/SecureClient.pdf`
16. IETF: The Transport Layer Security (TLS) protocol: Version 1.2. RFC 5246 (2008)
17. Initiative D21: Online-Banking – Mit Sicherheit! Web page (2008), `http://www.fiducia.de/service/publikationen/nonliner_atlas_2008.html`, (in German)
18. IT-Online: World-first SMS banking scam exposes weaknesses. Web page (2009), `http://www.it-online.co.za/content/view/1092105/142/`

19. Jackson, C., Boneh, D., Mitchell, J.: Transaction generators: Root kits for web. In: Proc. HotSec'2007. pp. 1–4. USENIX
20. Jakobsson, M., Myers, S. (eds.): Phishing and Countermeasures: Understanding the Increasing Problem of Electronic Identity Theft. John Wiley & Sons, Inc. (2007)
21. Li, S., Shah, S.A.H., Khan, M.A.U., Khayam, S.A., Sadeghi, A.R., Schmitz, R.: Breaking e-banking CAPTCHAs. In: ACSAC'2010. pp. 171–180
22. 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). LNCS, vol. 4886, pp. 88–103. Springer
23. Naor, M., Pinkas, B.: Visual authentication and identification. In: Advances in Cryptology – CRYPTO'97. LNCS, vol. 1294, pp. 322–336. Springer (1997)
24. Oppliger, R., Rytz, R., Holderegger, T.: Internet banking: Client-side attacks and protection mechanisms. Computer 42(6), 27–33 (2009)
25. PC World: Nokia: We don't know why criminals want our old phones. Web page (2009), http://www.pcworld.com/businesscenter/article/163515/nokia_we_dont_know_why_criminals_want_our_old_phones.html
26. Postbank: mTAN now free for all customers. Web page (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
27. Saturday Star: Victim's SIM swop fraud nightmare. Web page (2008), http://www.iol.co.za/index.php?art_id=vn20080112083836189C511499
28. Schneier, B.: Two-factor authentication: Too little, too late. Comm. ACM 48(4), 136 (2005)
29. Starnberger, G., Froihofer, L., Goeschka, K.M.: QR-TAN: Secure mobile transaction authentication. In: Proc. ARES'2009. pp. 578–583. IEEE Computer Society
30. Szydlowski, M., Kruegel, C., Kirda, E.: Secure input for web applications. In: Proc. ACSAC'2007. pp. 375–384. IEEE Computer Society
31. The Financial Express: Russian phone virus that 'steals money' may spread global. Web page (2009), http://www.financialexpress.com/news/russian-phone-virus-that-steals-money-may-spread-global/420770
32. Toorani, M., Shirazi, A.A.B.: Solutions to the GSM security weaknesses. In: Proc. NGMAST'2008. pp. 576–581. IEEE Computer Society
33. Volksbank Freiburg eG: iTANplus – mehr Sicherheit mit der indizierten TAN. Web page, http://www.volksbank-freiburg.de/itan.cfm?CFID=10869033&CFTOKEN=34249989&rand=1246061956151
34. Volksbank Rhein-Ruhr eG: Bankgeschäfte online abwickeln: Mit Sm@rtTAN optic bequem und sicher im Netz. Web page, http://www.voba-rhein-ruhr.de/privatkunden/ebank/SMTop.html
35. Volksbank Solling eG: Sm@rt-TAN-plus. Web page, http://www.volksbank-solling.de/flycms/de/html/913/-/Smart+TAN+plus.html
36. Wagner, D., Schneier, B.: Analysis of the SSL 3.0 protocol. In: Proc. USENIX Workshop on Electronic Commerce. pp. 29–40 (1996)
37. 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). LNCS, vol. 4968, pp. 75–91. Springer (2008)
38. Wikipedia: Transaction authentication number. Web page, http://en.wikipedia.org/wiki/Transaction_authentication_number
39. ZKA (Zentralen Kreditausschuss): FinTS – financial transaction services. Web page (2004), http://www.hbci-zka.de/spec/4_0.htm