hPIN/hTAN: Low-Cost e-Banking Secure against Untrusted Computers

Shujun Li[1], Ahmad-Reza Sadeghi[2] and Roland Schmitz[3]

1 Universität Konstanz

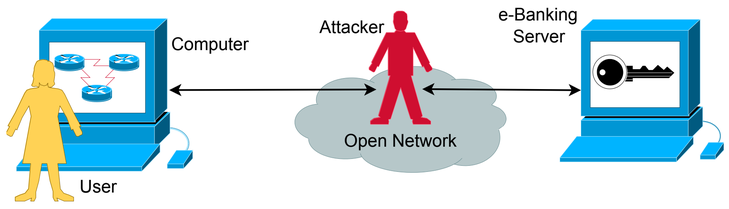
2

3 HOCHSCHULE DER MEDIEN

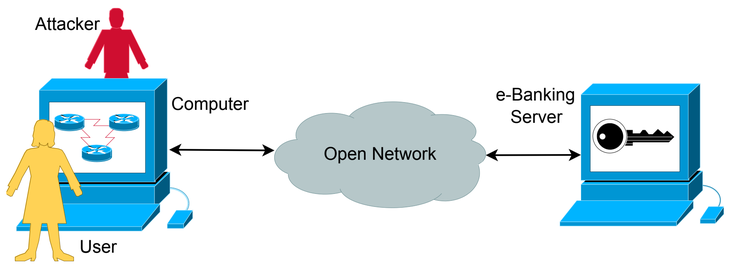
Abstract

We propose hPIN/hTAN, a low-cost hardware-based e-banking protection scheme when the adversary has full control over the user's computer. Compared with existing hardware-based solutions, hPIN/hTAN depends on neither trusted out-of-band channel, nor secure keypad, nor data encryption. Its security is based on a cryptographic hash function and human attention.

The Problem: Untrusted Computer



Man-in-the-Middle (MitM) Attacks: Untrusted Network.

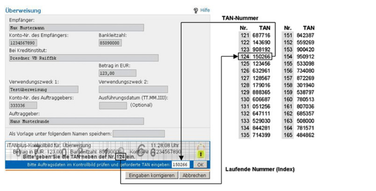


Man-in-the-Brower/Computer (MitB/MitC) Attacks: Untrusted Computer.

We consider the worst case of MitC attacks: the attacker has full control over the user's computer, including software, OS, and even hardware programmable via software.

Existing Solutions

The attacker's new goal: manipulating transactions.
The solution: transaction verification.

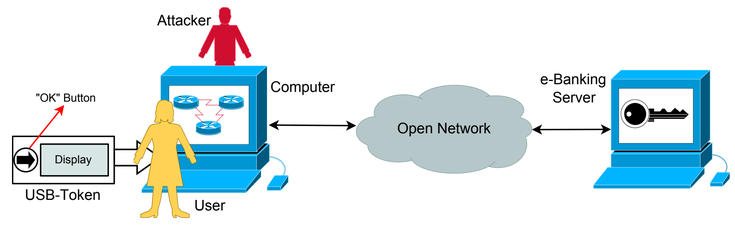
- Trusted out-of-band (OOB) channel: mTAN 
- CAPTCHAs: iTANplus 
- Encrypted Server2User or User2Server channel: Trusted devices such as mobile phones, PDAs, USB-tokens, etc. 
- Transaction-dependent TANs or digital signature: Trusted devices with secure keypad/display 
- Others: Trusted proxies, trusted computing platform, virtual machines, ...

Problems with Existing Solutions

- Trusted OOB channel may be untrusted: insecure cellular (especially GSM) network
- Trusted devices may be untrusted: mobile malware
- CAPTCHAs: prone to AI-based and human-assisted attacks
- Secure keypad: Low usability/portability, high costs
- Encryption (often asymmetric): high costs
- External dependency: cellular network, smart card, paper lists, optical sensors/cameras, trusted proxies, ...

The Research Question: Where is the best balance between security and usability?

Our Solution: hPIN/hTAN



- USB-token based solution: trusted display, "OK" button.
- Two parts: hPIN for login and hTAN for transactions.
- A better balance between security and usability: human attention is effectively explored, the costs are well controlled.
- Simplistic design: Less chance of bugs and security holes.
- Three "No"s: No encryption + No secure keypad + No trusted out-of-band channel.
- Three "h"s: hardware + hash + human.

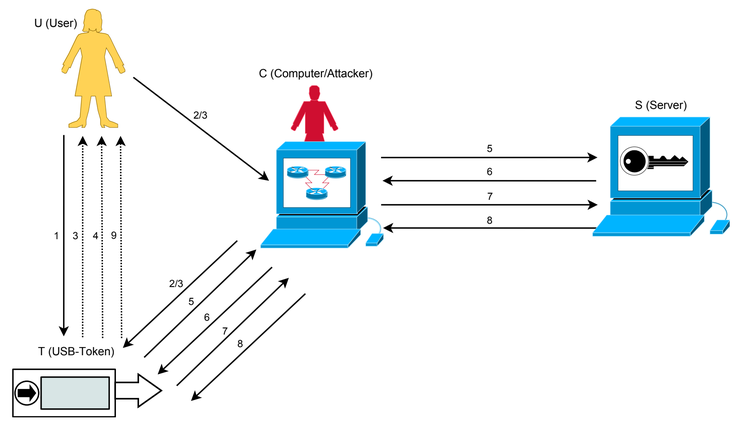
Notations

- Four parties involved: U (User), T (USB-token), C (Untrusted computer/Attacker), S (Server).
- User ID: IDU. User PIN: PIN. Shared secret: K_T. A cryptographic hash function: $h(\cdot)$.
- T: IDU, s, $K_T^* = K_T \oplus h(\text{PIN} \parallel s)$, $\text{PIN}^* = h(\text{PIN} \parallel K_T \parallel s)$, C_1.
- S: IDU, $h(K_T)$, C_2.

Security Requirements

1. *PIN confidentiality*
2. *User authenticity*
3. *Server authenticity*
4. *Transaction integrity/authenticity*

hPIN Protocol (Login)



Step 1: U connects T to C (if not yet connected), and presses the "OK" button of the USB-token.

Step 2: U enters IDU on the untrusted keyboard and sends it to T via C.

Step 3: For $i = 1, \ldots, n$, the following interaction is performed between T and U:

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;

b) U enters $\mathcal{F}_i(\text{PIN}(i))$ (which is always a printable character) with the untrusted keyboard or presses the "Backspace" key if she notices the last input is wrong;

c) if U presses the <Backspace> key, T performs $i = i-1$ and goes to Step 3a; otherwise T decodes $\mathcal{F}_i(\text{PIN}(i))$, shows $\text{PIN}(1) \cdots \text{PIN}(i)$ on its display for a few seconds and performs $i = i + 1$.

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 in the hTAN protocol, shows a "PIN correct" message to U via its display, and goes to Step 5; otherwise T performs $C_1 = C_1 + 1$, shows an alert to U and stops. If $C_1 > v_1$, T locks itself.

Step 5: T generates a nonce r_1, and sends (IDU, r_1) to S via C.

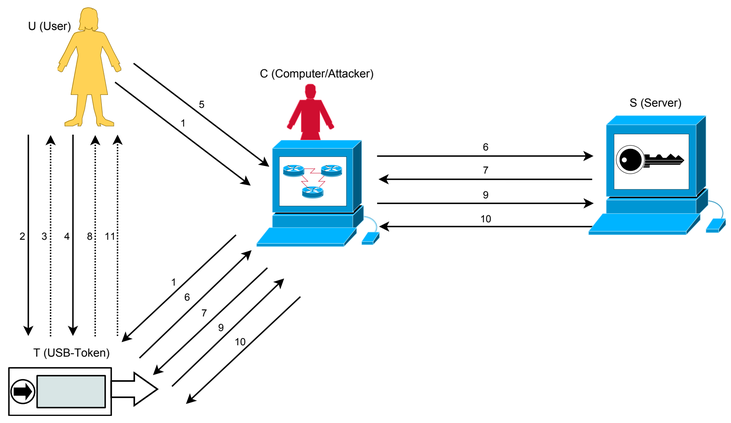
Step 6: S generates a new nonce r_2 and sends $(r_2, H_1 = h(r_1 \parallel h(K_T) \parallel r_2))$ to T via C.

Step 7: T checks if $H_1 = h(r_1 \parallel h(K_T) \parallel r_2)$. If so, T sends $(\text{IDU}, H_2 = h(r_2 \parallel h(K_T) \parallel r_1))$ to S via C; otherwise issues an alert to U via its display and stops.

Step 8: S checks if $H_2 = h(r_2 \parallel h(K_T) \parallel r_1)$. If so, S sets M ="success", otherwise performs $C_2 = C_2 + 1$ and sets M ="error". If $C_2 > v_2$, S locks the user's account. Then, S sends $\widetilde{H}_1 = h(r_1 \parallel M \parallel h(K_T) \parallel M \parallel r_2)$ to T (via C).

Step 9: T checks if $\widetilde{H}_1 = h(r_1 \parallel \text{"success"} \parallel h(K_T) \parallel \text{"success"} \parallel r_2)$. If so, it displays a "success" message, otherwise an "error" message, to U.

hTAN Protocol (Transaction Verification)



Step 1: U inputs sensitive transaction data (STD) on the untrusted keyboard of C.

To force U to verify the STD *simultaneously* on the trusted display of T, the STD are shown in clear *only* on T's display. In the online form, they appear as "*****". This requires real-time transmission of STD from C to T.

Step 2: U presses the "OK" button on T to finish STD input.

Step 3: T highlights STD for a few seconds, and prompts U to press the "OK" button again for re-confirmation.

Step 4: U presses the "OK" button for a second time to approve the STD.

Step 5: U inputs the non-sensitive transaction data (NSTD), and then clicks a "submit" button.

Step 6: T generates a nonce r_3 and sends (IDU, STD, NSTD, r_3) to S via C.

Step 7: S generates a new nonce r_4 and sends $(r_4, H_3 = h(\text{STD} \parallel r_3 \parallel h(K_T) \parallel r_4))$ to T via C.

Step 8: T checks if $H_3 = h(\text{STD} \parallel r_3 \parallel h(K_T) \parallel r_4)$. If so, T shows a "server verified" message to U via its display and goes to the next step; otherwise it shows a "bogus server" alert to U and stops.

Step 9: T sends $(\text{IDU}, H_4 = h(\text{STD} \parallel r_4 \parallel h(K_T) \parallel r_3))$ to S via C.

Step 10: S checks if $H_4 = h(\text{STD} \parallel r_4 \parallel h(K_T) \parallel r_3)$. If so, S executes the requested transaction and sets M ="success", otherwise sets M ="error". Then, S sends $\widetilde{H}_3 = h(\text{STD} \parallel r_3 \parallel M \parallel h(K_T) \parallel M \parallel r_4)$ to T (via C).

Step 11: T checks if $\widetilde{H}_3 = h(\text{STD} \parallel r_3 \parallel \text{"success"} \parallel h(K_T) \parallel \text{"success"} \parallel r_4)$. If so, it displays a "transaction approved" message, otherwise a "transaction failed" message, to U.

Security Analysis and Implementation

- Security based on a cryptographically secure hash function.
- PIN encrypted in the USB-token against theft and loss.
- The second half of each sub-protocol (hPIN, hTAN) can be replaced by a more complicated protocol if necessary.
- Implementation and user study is currently work in progress.

Table: hPIN/hTAN vs. Some selected hardware-based solutions.

	Mobile phone or PDA	Secure keypad	Encryption	Optical sensor or camera	External dependency	Smart card
hPIN/hTAN (this work)	No	No (one button)	No	No	No	No
mTAN/mobileTAN	Yes	No	No	No	Yes (Cellular network)	Yes
Sm@rtTAN plus	No	Yes	No	No	No	Yes
Sm@rtTAN optic	No	Yes	No	Yes	No	Yes
FINREAD, HBCI-3 (smart card readers)	No	Yes	Yes	No	No	Yes
photoTAN, Fotohandy-TAN	Yes	Yes	Yes	Yes	No	Yes
Sesam-Öffne-Dich ("Open Sesame")	Yes	Yes	Yes	Yes	Yes (Cellular network)	Yes
what-you-see-is-what-you-sign/confirm	Yes	Yes	Yes	Yes	No	No
QR-TAN	Yes	Yes	Yes	Yes	No	No
IBM ZTIC	No	No (two buttons)	Yes	No	No	Optional
"Sichere Fenster"	No	No	Yes	No	No	Yes
AXSionics personal tokens	No	No	Yes	Yes	Yes (Token server)	No
MP-Auth	Yes	Yes	Yes	No	No	No

FC 2010 – 14th International Conference on Financial Cryptography and Data Security, Tenerife, Canary Islands, Spain, January 25–28, 2010