Secure Confirmation of Sensitive Transaction Data in Modern Internet Banking Services*

Thomas Weigold
IBM Research - Zurich
twe@zurich.ibm.com

Alain Hiltgen
UBS AG Zurich
alain.hiltgen@ubs.com

Abstract— **In recent years attacks on Internet banking services have evolved from rather simple credential stealing attacks to advanced content-manipulation attacks by means of malicious software seeded on the client end-devices. This paper presents the risk mitigation approach of secure beneficiary confirmation on a trusted device combined with multi-level whitelist management to selectively authenticate transactions. Furthermore, two real-world implementations offering unique properties with regards to convenience and mobility while maintaining the highest level of security are described, along with practical results gained from deployment to a large user population.**

Keywords: Internet banking, malicious software, trusted device, user authenticaton, transaction authentication

1.1 Introduction

Financial transactions placed over Internet are supposed to be executed in real-time and have become extremely sensitive, since, shortly after their execution, they become very hard to recover. At the time of writing, modern Internet banking services, offering payments to freely selectable beneficiaries, follow the general approach of establishing at login a strongly authenticated and encrypted communication channel between the client end-device and the bank's server. Once established, the reliably identified user can either work freely with his or her account and perform all kinds of transactions, or is requested to additionally authorize transactions with a transaction-independent Transaction Authorization Number (TAN). In recent years, the authentication methods used for establishing such channel or authorize such transactions have been constantly revised and advanced with regards to commonly observed attack vectors; the main goal being to thwart attacks focusing on credential-stealing, for instance, via phishing or man-in-the-middle (MITM) [1,2,3]. However, in the mean time, more sophisticated attacks – specifically, silently manipulating the input and output on the client end-device by means of malicious software or malware (MSW) – have surfaced [4]. Some of these new attacks are built upon a combination of MSW and MITM if, for instance, MSW controls the web browser towards enabling or facilitating a subsequent MITM attack. Others exclusively use MSW towards silently manipulating transaction data submitted through or displayed in the web browser. Both of these attacks are commonly referred to as "man-in-the-browser" attacks [5].

In the light of such advanced content-manipulating attacks, in essence, information either sent to or keyed in on the client end-device can no longer be fully trusted to reach that device or the bank's server without any relevant changes. Therefore, the approach of relying only on user authentication and/or transaction authorization had to be considered insufficient for protecting sensitive transaction data. Many regulators reacted [6,7,8,9] and forced their banks into providing customers with appropriate complementary security means. Transaction-dependent authentication was soon identified as a potential means to thwart the new type of attacks, but only if the process for confirming the sensitive content of transactions could effectively be shielded from the client end-device by moving it to a trusted device not exposed to MSW. In this paper, we discuss such security solutions and point out crucial success factors that eventually determine if the user is able to intuitively understand his or her responsibility and due diligences and therefore can also be deemed liable to take advantage of the new protection mechanisms at his or her disposal. In addition to security relevant aspects, we also consider business relevant properties such as convenience, mobility, integration, administration, and last but not least costs. This finally leads us to an approach for beneficiary confirmation, enhanced with whitelist management and two trusted device implementations that offer a unique combination of properties found to best match the business and customer expectations of a large bank.

The remainder of this paper is organized as follows: After describing relevant attack vectors and solution properties in Section 2, we discuss the state-of-the-art in Section 3. Afterwards, Section 4 presents the enhanced approach of beneficiary confirmation with whitelists, followed by the description of two trusted device implementations in Section 5. Section 6 summarizes by discussing practical results while Section 7 closes with final conclusions.

2 Relevant Attack Vectors and Solution Properties

The plethora of conceivable attacks against Internet banking services can be classified into three main categories: social attacks, physical attacks, and software attacks: (i) Social attacks try to convince users to perform certain actions, for example, to authorize a transaction to an unknown beneficiary or to divulge secret information such as their authentication credentials to an unknown third party. To our knowledge, there is no technical solution that reliably thwarts such attacks without assuming a certain degree of user due diligence. Therefore, in this paper we focus on providing the user with the technical environment that enables him or her to easily comply with due diligence rather than trying to eliminate these social attacks in the first place. (ii) Physical attacks try to steal tokens, crack chip cards or issue fake devices. Such attacks are considered out of scope for this paper, since they cannot be easily launched against a large number of users and can be traced using standard investigative methods. Nevertheless, we recommend that user credentials such as cryptographic keys shall be reasonably protected against physical attacks, for instance, by storing them on tamper resistant smart cards secured by a PIN code. (iii) Software attacks thus are at the main focus of this paper, as they must be considered the most common and most threatening attack vector for several reasons. Firstly, software attacks can be launched easily from remote against a very large number of users, for example, by spreading a virus via email or via infected web sites. Research shows that the latter approach, also known as “drive-by infection”, is gaining large popularity at the time of writing [4]. Secondly, the effort required to implement software attacks is constantly decreasing due to the fact that the required knowledge and easy to use tools are available from the Internet at very low costs and users have problems with keeping their software at the latest patch level [10]. Lastly, software attacks are, if possible at all, very hard to trace back, as the initiators typically hide behind botnets and server infrastructures hosted in various foreign countries and governed by various different jurisdictions.

Such software attacks can be further categorized as follows: credential-stealing attacks, channel-breaking attacks, and content-manipulation attacks. Credential-stealing attacks aim at fraudulently gathering a user’s credentials, for example, via MSW or by tricking the user into voluntarily revealing them, for instance, to a fake login page. On the other hand, channel-breaking attacks, aim either at intercepting messages between the client end-device, typically a personal computer (PC), and the banking server or at taking over the client session as a man-in-the-middle. While these two attack classes along with relevant authentication methods have been discussed in depth in [1], this paper further focuses on content-manipulation attacks targeting the transaction data entered via the client PC. This can be considered the most advanced attack vector, which renders precautions used to thwart the other two attack classes, for instance, short-time passwords generated by hardware tokens or mutually-authenticated SSL/TLS connections ineffective. Content-manipulation attacks are typically implemented by means of MSW seeded on the client PC. Despite the fact that user’s protect their PCs by maintaining firewall and anti-virus software, in practice, MSW finds its way to infect PCs by exploiting either user inadvertence or constantly reappearing software deficiencies. Experience shows that it is impossible to reliably and lastingly protect software running on today’s client hardware and operating systems, lacking trusted computing support, by means of other software running on the same platform [11]. Momentarily observed effectiveness of such software must therefore be interpreted with due precaution and as such be considered more a consequence of its proprietary nature than of any specific hardening [12].

From a long term perspective it is wiser therefore to assume that a user’s PC, that cannot be securely rebooted for e-banking and that is regularly used for browsing and e-mailing over the Internet, will continue to be exposed to MSW able to take over full control. Under this assumption, an attacker does no longer need to steal authentication credentials or break into the secure channel established between the user’s PC and the bank’s server, but will rather take direct advantage of MSW to silently submit or modify transaction data while the genuine user is connected to the bank. Real-world examples of MSW supporting such attack vectors are the Trojans “ZEUS’’, “Trojan.Silentbanker” and others [4]. The only long term solution to this exposure appears to come from complementing strong authentication at login with sensitive data authentication at transaction placing, and using for the additional confirmation step a trusted platform not exposed to the Internet, the client PC, or any MSW potentially spreading through these means [13]. In other words, the process of confirming sensitive transaction data must be moved from the PC to a trusted device that, at minimum, includes a display, one or more buttons, as well as the user credentials. This way, the user can reliably verify critical transaction details on the trusted display, for instance, the beneficiary account of a payment instruction, and then confirm the respective data via the trusted buttons. The device then typically generates a transaction specific signature to be sent to the banking server. Credentials used to do this are never exposed to the client PC, in other words, the trusted device always signs what the user has seen and approved on the same device.

The foremost property of an Internet banking service offering payments to freely selectable beneficiaries should be resistance against the discussed attack vectors. However, in addition to the security aspect, other aspects can be equally important for the applicability and success of a solution. Therefore, in this paper we also consider the following additional business relevant solution properties:

- **Reasonableness:** The solution’s security shall be intuitive, even for a non-experienced user (responsibility and due diligence are easily assumable).
- **Convenience:** The solution shall be easy to use (e.g. fast, well known interaction patterns, mobile, no software installation on client PC).
- **Mobility:** The trusted device shall be easy to carry around, shall not require software installation, and shall work with standard PCs and operating systems.
- **Integration:** The solution shall integrate well with existing infrastructures. This includes client-side hardware and software, communication protocols, and server-side requirements.
- **Administration:** The solution shall be easy to administrate and maintain, for instance, by means of remote configuration and software update.
- **Cost:** Overall costs shall be low to qualify for high-volume deployments, for instance, in retail banking.

3 STATE-OF-THE-ART

At the time of writing, the majority of Internet banking services offering free selection of beneficiaries does not satisfy the security requirements discussed in Section 2. In particular, they do not provide a technical environment that allows a user to reliably detect and thwart content-manipulation attacks carried out by means of MSW. For instance, software-only solutions or trusted device solutions, without secure display, offer some transitional security, but no reliable technical resistance against such attacks. Only a few solutions potentially providing the desired security level have surfaced (c.f. Table 1). All of them comply with the inevitable requirement to confirm sensitive transaction data via a separate trusted device, independent from the display and keyboard of the potentially infected client PC. What differs is whether or not the device is physically connected to the client PC and how data is entering and leaving the device. For example, transaction details to be confirmed might be sent to the device automatically via a Universal Serial Bus (USB) connection or might have to be entered manually by the end user. Similarly, the result of the confirmation process, typically a digital signature or a Transaction-dependent Authentication Code (TAC), might be retrieved either automatically or manually. The way how communication with the security device is implemented also has significant impact on afore mentioned solution properties. Table 1 lists the most familiar of these solutions, together with their PC connection, data input/output methods, and some property ratings.

Solution A is based on a standalone TAC generator device not connected to the PC or any network, as described in [1] or defined by the EMV Chip Authentication Program (CAP) for secure credit card payments over the Internet [14]. Such devices either have built-in cryptographic keys or provide a smart-card interface such that, for instance, EMV compliant credit/debit cards can be inserted. While security is fine, the user has to manually enter the transaction details and PIN via a built-in keypad. The device then generates a TAC that needs to be manually copied to the web browser. Obviously the device is not exposed to MSW, as it is not connected to the PC. But for the same reason, it can also neither be remotely configured nor updated. Furthermore, manual interaction decreases convenience and renders the approach impractical for frequent users, whereas other properties like the low device costs make it an interesting option for occasional users.

Solution B can be considered a variation of solution A where the transaction details are not entered manually. It makes use of the user's mobile phone as a separate end device and the short message service (SMS) available in mobile phone networks as an independent communication channel to that end device. A server-generated TAC is sent to the user's mobile phone via SMS along with the details of the transaction previously placed over the Internet. The user can then verify the transaction details on his or her phone and approve them by copying the associated TAC to the web browser. In contrast to the other proposed solutions, the security prerequisites for this approach already raised some concerns, due to the increasingly vanishing independence between mobile and Internet communication channels and the potentially inappropriate customer authentication used by mobile service providers before issuing secondary SIM cards or forwarding SMS messages to a different number [15]. While the effort required to integrate an SMS service is acceptable and the mobility of the solution is good, the improved user convenience remains somewhat limited, as the TAC still needs to be copied by hand. Further drawbacks are that the SMS delivery is not highly reliable in all countries, e.g. US, the SMS service generates significant recurring costs, and involved network operators can trace the messages.

Solution C can be considered another variation of solution A where the transaction details are not entered manually but transmitted within an encrypted flickering image such that a trusted device can read it via an optical interface from the client PC screen. For that the user must place a special trusted device equipped with optical sensors and a decryption key firmly in front of the flickering image onto the computer screen. Transaction details to be verified and a TAC to be copied by hand are both displayed after local decryption on the trusted device. This approach somewhat increases convenience while the interaction pattern may appear unfamiliar, could require fine motor skills from the user and sometimes also comes with dependencies from screen size and resolution. Furthermore, security is fine, but integrating the flickering image functionality on the server side requires changes to the banking service and the need for an optical interface increases costs.

Solution D makes use of yet another trusted device, a secure smart-card reader equipped with a display and keypad and connected with the PC via USB, as specified for example by the FINREAD reader standard [16]. Such devices with bidirectional communication are sometimes also referred to as class 3, class 4, or Secoder smart-card readers, according to definitions by the Zentraler Kreditausschuss (ZKA) in Germany. Transaction data is transmitted automatically from the web browser to the smart-card reader. The user inserts his or her smart card into the reader device, possibly unlocks the card by entering a PIN code, verifies the transaction details on the trusted display, and finally confirms sensitive transaction data via the trusted keypad. The reader digitally signs the confirmed transaction data with the help of the smart card and automatically sends the result back to the web browser, which forwards it to the server. This represents the most convenient solution as the user can confirm sensitive transaction data without any manual copying by simply pressing one trusted button. Furthermore, it is also the only approach that supports usage of Public Key Infrastructure (PKI) smart cards with a signature of a size too large to be copied by hand and it supports secure remote administration of the reader firmware via the USB connection. Unfortunately, proposed devices are typically quite bulky and require specific device drivers and possibly additional software to be installed on the user's PC. Therefore, these solutions are less mobile and often also face problems in working with different operating systems and web browsers or try to overcome this with an own browser inducing the need for bulky security updates. Last but not least, these devices require an effective shielding from the client PC and are rather expensive.

Obviously, all these solutions despite having their own advantages also reveal some shortcomings, specifically when facing large scale deployments. From a user perspective, and thus critical for the overall success of such a solution, security, convenience and mobility can be considered the most important properties [17]. Therefore, in this paper (c.f. Section 5) we present two solutions that clearly improve the respective properties of the solutions listed in Table 1. The first is based on the low cost Solution A augmented with comprehensive server-side whitelist management to improve user convenience, specifically for occasional users. The second can be considered a variant of the highly convenient Solution D augmented with whitelisting, enhanced mobility and simplified integration properties, towards best supporting frequent users.

Property\Solution	A	B	C	D
Connected To PC	no	no	no	yes
Input	manual	auto	auto	auto
Output	manual	manual	manual	auto
Convenience	+	++	++	+++
Mobility	+++	+++	+++	+
Client-side Integration	+++	+++	++	+
Cost	low	medium	medium	high

Table 1: Contemporary transaction authentication solutions and their properties

4 Beneficiary Confirmation and Whitelists

Confirming critical transaction details on a trusted device for mitigating risks associated with fraudulent payments is a rather intuitive approach but, as with most intuitive things, challenges start to surface only when it comes to implementation. The two main questions to be answered are: (a) ***What** should be displayed?* (b) ***When** should it be displayed?*

Although practice can differ from one country to another, we made the observation that the success of a payment order often does not depend on the correctness of the specified beneficiary name and/or address, but solely on the correctness of the specified beneficiary account and/or payment reference number. For the payee's bank it is mostly impossible to validate the beneficiary name provided together with a beneficiary account, whereas for the beneficiary's bank, as long as the beneficiary account and/or payment reference number allow for an unambiguous crediting, the additionally provided information, like the name and address of the beneficiary, are mostly not taken into consideration. This shows that towards an effective beneficiary confirmation, it is essential to *identify **what** uniquely determines the beneficiary of a transaction*, depending on the beneficiary's bank and the type of payment, and have the user check and confirm only this. Nevertheless, many implementations of such extra confirmation steps seem to miss this crucial aspect as exemplified in Table 2. By displaying too many transaction details, the user is fooled in approving a transaction, irrespective of the correctness of what matters (the beneficiary account) being blinded by the correctness of what does not matter (the beneficiary name). Therefore, we propose to only ask the user to confirm the beneficiary account or payment reference number, if this has been verified to eventually determine the beneficiary..

Beneficiary Name:	John Meyer
Beneficiary Address:	Private Street 10, Hometown
Beneficiary Account:	123-45-67-89
Transaction Amount:	1'000 US$
Transaction Date:	Mai. 12, 2010
Confirmation Code:	1234

Table 2: Displayed transaction details to be confirmed

Another important aspect to take into consideration is the de facto increase in negligence, if users are asked for confirming things too often or for confirming things they already confirmed before. It is therefore also crucial for the effectiveness of beneficiary confirmation to *ask for a confirmation only **when** the beneficiary is neither globally trusted* by the bank *nor known to be trusted* by the user. Keeping track of globally and personally trusted beneficiaries essentially resumes to using global and personal whitelists, managed by the bank and the user respectively. Beneficiary confirmation is then equivalent to the functionality of adding a new beneficiary to the personal whitelist and needs to be complemented by an according functionality enabling the user to also remove a beneficiary from the personal whitelist. Before executing a transaction the banking service in such a case simply checks whether the beneficiary is on the global whitelist of the bank or on the personal whitelist of that user and asks for an extra confirmation only in the exceptional case where a non-globally-trusted beneficiary is used by that user for the first time. For increased convenience during first usage, the personal whitelist can be prefilled either with beneficiaries securely provided by a new user or with beneficiaries collected from prior payments of an existing user.

With such a multi-level whitelist approach, convenience will typically be very high for the large majority of occasional users, primarily dealing with globally trusted beneficiaries. A disconnected solution, like the one described in 5.1, can thus be estimated well serving the purpose for most of these users. Nevertheless, for a small minority of frequent users, with regular payments to new and mostly non-globally-trusted beneficiaries, a connected convenience solution, like the one described in 5.2, is deemed necessary to best support these users in their daily business.

5 Client Integration

This section presents two concrete client-side solutions implementing the beneficiary confirmation approach described in Section 4. Both solutions have been fully implemented and rolled out to a large number of users (c.f. Section 6).

5.1 The Disconnected Reader Solution

The first solution is an extension of the short-time password solution presented in [1]. This means it is based on a standalone TAC generator device not connected to the PC or any network (cf. solution A in Section 3). The device represents a smart-card reader including a simple display and a keypad. Furthermore, the user holds a smart card personalized with a symmetric key and a strictly monotonic counter with which it can generate a response cryptogram for a given input challenge (see [1] for more details). For user authentication the back-end server provides a user specific random challenge in a web page. After card insertion and entering the PIN code the user manually copies the challenge from the web page to the reader, which sends it to the smart card for response calculation. Finally, the user copies the response displayed on the reader back to the web page to submit it to the server. Figure 1 illustrates the overall solution architecture.

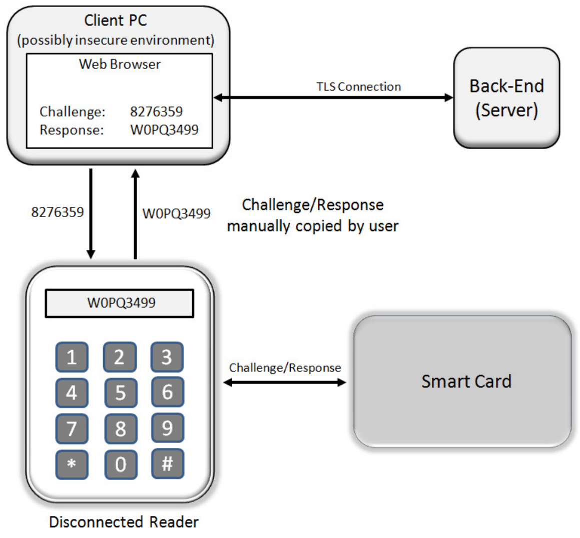

Figure 1: Architecture of the disconnected reader solution

This authentication solution has been extended to support beneficiary confirmation. To achieve this, the server requests a response cryptogram for a transaction related challenge, the beneficiary account, before a transaction request is processed. This means the user enters the beneficiary account into the reader and copies back the response to the web page, just like with user authentication. At this point in time the account number is not displayed on the web page since the PC display is not considered trusted. Instead, the user is forced and gets used to retrieve the number from another supposedly more trusted source, for instance, a paper invoice. In case the account number is too long to be used as a challenge, the web page indicates which part of the account number shall be used. For example, if the account number is 123456789 and 34567 shall be used as the challenge, the web page displays 12■■■■■89 to guide the user. Requiring the user to carry out this rather time-consuming confirmation process for every transaction would be impractical. Only in combination with the whitelist management approach described in Section 4, the solution becomes practical for occasional users.

5.2 The Convenience Reader Solution

The second solution aims to significantly increase user convenience by moving from a disconnected to a connected approach and appropriately addressing the security implications to effectively shield the trusted device from potential MSW on the client PC. As the trusted device is physically connected to the client PC via USB, so that data input and output can be carried out automatically, the solution can be considered similar to solution D described in Section 3. However, the way how the trusted device is integrated into the client-server communication between the web browser and the back-end server is fundamentally different and thus eliminates the drawbacks of solution D with respect to mobility and client-side integration as summarized in Table 1.

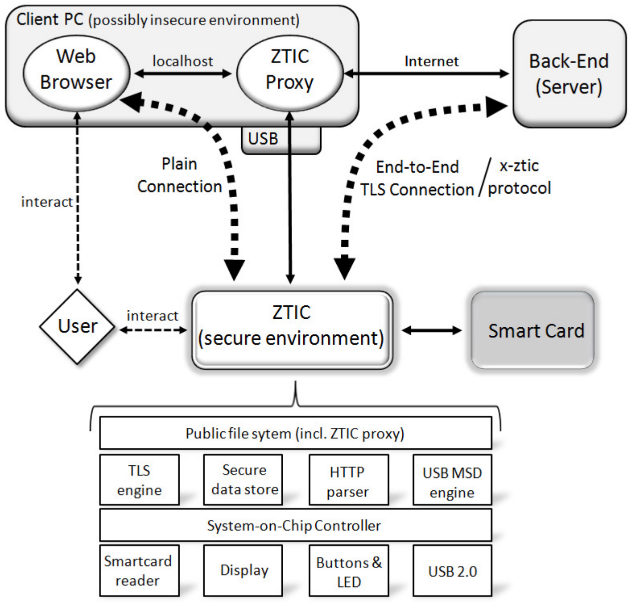

Figure 2: Architecture of the ZTIC-based convenience reader solution

This solution is an enhanced version of the initial Zone Trusted Information Channel (ZTIC) approach described in [18]. Figure 2 illustrates the overall solution architecture. The basic idea of the ZTIC is to hook a trusted device into the communication channel between the web browser and the back-end server and move the TLS protocol endpoint from the potentially insecure client PC to the trusted device. As illustrated in Figure 2, the ZTIC device does this with the help of the *ZTIC proxy* program included in the file system of the ZTIC, which is connected to the PC as a standard USB mass storage device (MSD). The ZTIC proxy, which is started by the user, on one hand starts the web browser and acts as a localhost server to the web browser while on the other hand connects to the back-end server of the bank. By communicating with the ZTIC device via a dedicated file in the MSD file system, the proxy routs all HTTP traffic between web browser and server through the ZTIC. Additionally, the ZTIC proxy helps the ZTIC device to establish an end-to-end TLS connection between the ZTIC and the remote server. The ZTIC device can be considered a local man-in-the-middle that maintains a TLS connection to the remote server and a plain connection to the browser (cf. Figure 2). The main purpose of the ZTIC proxy program is to manage the TCP/IP connections to the server and the browser and blindly relay all communication between the two via the ZTIC device. Additionally, the proxy rewrites all links in HTTP responses to point to its own localhost port such that communication always keeps flowing through itself and thus through the ZTIC device. More details about the basic ZTIC approach can be found in [18].

Each ZTIC device is initialized with a set of trusted server certificates and a device specific client certificate stored in the ZTIC's secure data store located in the internal persistent memory of the system-on-chip controller. These credentials are used within the TLS handshake to carry out mutual authentication. While the server certificates ensure reliable authentication of the back-end server, the device-specific client certificate allows the bank to verify that the TLS connection is established by a genuine ZTIC device. These precautions render channel-breaking attacks such as the classical man-in-the-middle network attack impossible. The same applies for TLS proxy servers or any other third party services that compromise privacy. All asymmetric keys and TLS session keys are maintained by the ZTIC device and are never exposed to the client PC.

Once both connections, the plain connection and the TLS connection as indicated in Figure 2, are established, HTTP communication between the web browser and the server is carried out normally except that it is routed through the ZTIC for TLS encryption and decryption. Building on this, an additional protocol between the ZTIC device and the back-end server, the so-called *x-ztic* protocol, has been introduced. This protocol is embedded within the HTTP traffic by inserting a custom field named 'x-ztic' into the HTTP headers. In other words, the ZTIC device adds an x-ztic field to all HTTP request headers to provide additional information to the server. Likewise, the server adds an x-ztic field to HTTP response headers to control the ZTIC's behavior. This way the ZTIC transmits state information such as the current firmware version, smart-card related information (e.g. card present, PIN verified), contract related information (e.g. preferred language), and cryptograms for authentication or beneficiary confirmation purposes to the server. Similarly, the server makes use of the x-ztic protocol to control the ZTIC's state, send an authentication challenge or beneficiary confirmation data to the ZTIC, or provide a firmware update.

In general, user authentication and beneficiary confirmation is based on the same smart card and related challenge response protocol used with the disconnected reader solution. However, instead of requiring the user to copy challenge and response manually, they are automatically transported to/from the device within the x-ztic protocol. The user however has to confirm or reject every login or beneficiary confirmation requests via the ZTIC's display and buttons. From a user perspective the solution works as follows:

1. The user attaches his or her ZTIC device to the client PC and starts the ZTIC proxy program available in the ZTIC's file system, which in turn starts the web browser.
2. The user inserts the smart card into the ZTIC device and enters the PIN code to unlock the card into an HTML form provided by the server. (When the form is submitted, the ZTIC extracts the PIN code from the HTTP request and sends it to the smart card).
3. The user is requested to confirm or reject the Internet banking login by pressing the OK/Cancel button on the ZTIC device, in reaction to an accordingly unambiguous display on the ZTIC device.
4. As soon as beneficiary confirmation is required, the user is requested to verify and confirm or reject the beneficiary account displayed on the ZTIC device by pressing the OK/Cancel button on the same device.

Whenever the server transmits a challenge to the ZTIC in the x-ztic field of an HTTP response header, the ZTIC removes and stores the challenge until the next HTTP request is issued by the web browser. Only then the user is requested to verify and confirm the transaction on the device if it is a beneficiary confirmation challenge or to confirm the login if it is an authentication challenge. This way the challenge response protocol is embedded within the standard HTTP request/response communication between the browser and server. On the server side, the web application flow is influenced by the state information sent by the ZTIC via x-ztic protocol. For example, if the ZTIC indicates that there is no smart card inserted in the ZTIC device, the server provides an appropriate web page requesting the user to insert the card. Figure 3 shows the ZTIC device comprising a high resolution OLED display, two buttons, a mini USB connector, a scroll wheel, and a smart-card slot at the top side.

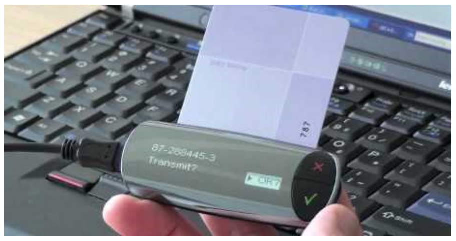

Figure 3: The ZTIC device

6 Practical Results and Discussion

The approach of beneficiary account confirmation with multi-level whitelist management as presented in Section 4 and the two solutions described in Section 5, have been implemented as *UBS Access Card Reader* and *UBS Access Key* solutions at UBS Switzerland and deployed to roughly 650'000 active Internet banking users around the world; essentially using the disconnected Access Card Reader for private customers and the Access Key convenience reader for corporate customers. Every account, personal or corporate, manageable via Internet banking, has also had a whitelist attached to it, so that the UBS implementation at the time of writing works with about 1.2 million of personal whitelists. Most of these lists are very short and hardly change over time, as summarized by Table 3 and 4. This is manly due to the global whitelist currently holding about 50'000 entries. Nonetheless, the figures below also clearly identify the need for a convenience solution to best support a minority of most frequent users, working with whitelists of up to 70'000 beneficiaries and up to 200 new entries a month..

Size	0-9	10-99	100-999	1'000-9'999	≥10'000
Occurrence ratio	60%	38%	1.9%	0.1%	(18)

Table 3: Sizes of personal whitelists at UBS in 2010

New entries	0	1-4	5-14	15-49	50-99	≥100
Occurrence ratio	80%	19%	0.9%	0.1%	(190)	(96)

Table 4: New entries per personal whitelist per month at UBS in 2010

From a security perspective, both solutions reliably thwart credential-stealing and content-manipulation attacks given the user is aware of his or her responsibility and due diligence. Limiting transaction details to be confirmed to the beneficiary account and requesting confirmation only if absolutely necessary has shown to facilitate this awareness, as confirmed by usability tests carried out during the pilot phase. The ZTIC based solution additionally shows stronger privacy properties, since the server certificate is validated by the trusted device and does not rely on a stringent fingerprint validation by the user.

With regards to the other business relevant properties defined in Section 2, the solutions behave as follows:

- **Reasonableness:** The simplicity of beneficiary account confirmation combined with low confirmation frequency due to whitelist management makes the overall solution and due diligence obligations intuitive to understand for all users.
- **Convenience:** While occasionally confirming a beneficiary account using the disconnected reader is convenient, the ZTIC based solution also offers perfect convenience for frequent users. This is essentially due to the fact that the device is kept simple (well readable OLED display and only two buttons for OK and Cancel) and the user does not need to manually copy any data.
- **Mobility:** The ZTIC device has a well-known key fob form factor and works with all standard PCs, operating systems, and web browsers without requiring any software installation. As a result, it is easy to carry around and it can be used almost anywhere. This also applies for the disconnected reader although it is a bit more bulky due to its onboard keypad.
- **Integration:** The ZTIC solution integrates exceptionally well with existing client and server infrastructures because it solely uses standard protocols and interfaces, namely HTTP, TLS, and USB MSD. On the server side, only the x-ztic protocol has to be integrated into the existing web application. Integrating the disconnected reader solution is even easier and works also with client end-devices not equipped or not supporting USB, e.g. mobile devices.
- **Administration:** Due to the fact that there is an end-to-end mutually authenticated TLS channel between the ZTIC device and the server, configuration data and firmware updates can be easily and securely pushed to the device utilizing the x-ztic protocol and support for different cards, e.g. PKI cards, can easily be added. Obviously, this is not possible with the disconnected reader.
- **Cost:** The ZTIC device consists of low-end standard hardware components such that the device cost is at the same level or even lower than the cost of today's secure smart-card reader devices described in solution D (cf. Section 3). Nevertheless, the disconnected reader solution remains much cheaper.

7 Conclustions

Advanced content-manipulation attacks on Internet banking services have become a real threat and require new security solutions focusing on sensitive data authentication to be developed. Regulators increasingly force banks to react, but contemporary solutions, providing the desired security, are lacking with regards to other business relevant properties such as reasonableness, convenience, mobility, or integration.

Towards showing how these secondary but essential factors can be improved, we presented the approach implemented by a larger bank with roughly 650'000 active Internet banking users around the world. More specifically, in this paper we made the following contributions:

- We defined relevant attack vectors and solution properties to be considered in Internet banking services and provided a brief review of the state-of-the-art.
- We presented an approach of beneficiary confirmation on trusted devices combined with multi-level whitelist management, which considerably increases user convenience, compared to contemporary solutions, and also facilitates user awareness of responsibilities and due diligence.
- We outlined two equally secure client integration solutions strongly tailored towards supporting optimization of the cost/benefit ratio with regard to a large customer base of both occasional and frequent users.
- With the ZTIC based solution, we presented a connected reader that uniquely combines security with unparalleled convenience, mobility, administration, and integration properties.

From the level of attacks faced at the time of writing, it is to be concluded, that modern Internet banking services will be able to survive only if banks strongly care about the reasonableness of their solutions and users strongly care about their responsibility and due diligence to protect credentials and validate transaction data whenever needed.

References

[1] A. Hiltgen, T. Kramp, T. Weigold, “Secure Internet Banking Authentication”, IEEE Security and Privacy Journal, vol. 4, no. 2, March/April, 2006, pp. 21-29.

[2] T. Weigold, T. Kramp, M. Baentsch, “Remote Client Authentication“, IEEE Security and Privacy Journal, vol. 6, no. 4, July/August, 2008, pp. 36-43.

[3] B. Parno, C. Kuo, A. Perrig, “Phoolproof Phishing Prevention“, Proc. Financial Cryptography and Data Security, Springer LNCS, vol. 4107, 2006, pp. 1-19.

[4] Reporting and Analysis Center for Information Assurance (MELANI), “Information Assurance – The Situation in Switzerland and internationally”, Semi-annual report (public). http://www.melani.admin.ch/dokumentation/00123/00124/index.html?lang=en

[5] U. Nattakant, “Review of Browser Extensions, a Man-in-the-Browser Phishing Techniques Targeting Bank Customers”, Proc. 7th Australian Information Security Management Conference, 2009, paper 19, http://ro.ecu.edu.au/ism/19.

[6] Monetary Authority of Singapore (MAS), “Internet Banking Technology and Risk Management Guidelines”, June 2008 (public). http://www.mas.gov.sg/legislation_guidelines/banks/guidelines/Internet_Banking_Technology_Risk_Management_Guidelines.html

[7] Hong Kong Monetary Authority (HKMA), “Strengthening Security Controls for Internet Banking Services”, Circular, July 2009 (public). http://www.info.gov.hk/hkma/eng/guide/circu_date/20090713e1.htm

[8] European Payments Council (EPC), “Customer to Bank Security - Good Practices Guide”, March 2009 (public). http://www.europeanpaymentscouncil.eu/knowledge_bank_detail.cfm?documents_id=225

[9] FedFocus: “Bank Revenues and Fraud Detection: A Marriage made in Heaven?”, July 2010. http://www.frbservices.org/fedfocus/archive_risk_management/risk_0710_01.html

[10] Stefan Frei, Thomas Duebendorfer, Bernhard Plattner, “Firefox (In) security update dynamics exposed”, ACM SIGCOMM Computer Communication Review, vol. 39, issue 1, January 2009, pp. 16-22.

[11] D. Challener et. al., “A Practical Guide to Trusted Computing”, Prentice Hall Computing, 2007, ISBN 978-0132398428.

[12] C. Ronchi, S. Zakhidov, “Hardened Client Platforms for Secure Internet Browsing”, in N. Pohlmann, H. Reiner, W. Schneider (Editors): Secure Electronic Business Processes, Vieweg (2008), pp. 367-379.

[13] R. Oppliger, R. Rytz, T. Holderegger, “Internet Banking: Client-Side Attacks and Protection Mechanisms”, IEEE Computer, vol. 42, no. 6, June 2009, pp. 27-33.

[14] MasterCard International, “Chip Authentication Program – Functional Architecture”, OneSmart Authentication, September 2004. https://mol.mastercard.net/mol/molbe/public/login/ebusiness/smart_cards/one_smart_card/biz_opportunity/cap/index.jsp

[15] Aviah Litan, “Where Strong Authentication Fails and What You Can Do About It“, Gartner Research, ID Number: G00173132, December 2009. http://www.gartner.com/DisplayDocument?ref=clientFriendlyUrl&id=1245013

[16] European Committee for Standardization (CEN), “Financial Transaction IC Card Reader (FINREAD)”. http://www.cen.eu/cen/Sectors/Sectors/ISSS/CEN%20Workshop%20Agreements/Pages/FINREAD.aspx

[17] Cormac Herley, “So Long, and No Thanks for the Externalities: The Rational Rejection of Security Advice by Users”, Proc. New Security Paradigms Workshop, ACM, 2009, pp. 133-144.

[18] T. Weigold et al, “The Zurich Trusted Information Channel - An Efficient Defense against Man-in-the-middle and Malicious Software Attacks”, P. Lipp, A.-R. Sadeghi, and K.-M. Koch (Eds.): TRUST 2008, LNCS 4968, pp. 75–91, 2008, Springer-Verlag.