Avoiding Man-in-the-Middle Attacks When Verifying Public Terminals*

Gergely Alpár and Jaap-Henk Hoepman

TNO, The Netherlands and ICIS Digital Security, Radboud University Nijmegen, The Netherlands
{gergely, jhh}@cs.ru.nl

Abstract. An individual who intends to engage in sensitive transactions using a public terminal such as an ATM needs to trust that (a) all communications are indeed carried out with the intended terminal, (b) such communications are confidential, and (c) the terminal's integrity is guaranteed. Satisfying such requirements prevents man-in-the-middle attacks and eavesdropping.
We have analysed several existing transaction schemes and concluded that they tend not to meet all requirements during the entire transaction. We propose a new, generic protocol that provides (a) optional terminal identification, (b) key establishment, and (c) customisable integrity assurance.

1 Introduction

Individuals often have to use public terminals for sensitive communication in which trust is taken for granted; however, establishing trust is not easy in a public environment[1]. Several schemes in the literature have been proposed for the users to verify such terminals and to receive a reliability report before performing the main transaction. This approach is not sufficient as an adversary can place an intermediary platform that relays all messages between the trustworthy terminal and the user during verification. After verification this hostile platform will be trusted by the user, who may communicate sensitive data during the main transaction.

To protect against such a *man-in-the-middle* (MitM) attack, the main transaction has to be linked strongly to the verification. There are two ways to establish this link: key exchange and identification. A key exchange is used to establish a secure channel that is bound to the verification and to the verified honest device, while an identification process of a terminal demonstrates locality, that is, a proof of direct contact between the user and the public terminal. Considering these two methods independently, we can analyse and design protocols more consciously.

The purpose of the present study is twofold. First, we examine whether existing schemes can exclude man-in-the-middle attacks. Second, we design a flexible generic protocol that optionally incorporates these linking methods.

The paper is organised as follows. Section 2 presents preliminary notions in relation to cryptography and trusted computing. The problem statement in Section 3 is divided into three parts.

* This research is supported in part by the research programme Sentinels as project 'Mobile IDM' (10522). Sentinels is being financed by Technology Foundation STW, the Netherlands Organisation for Scientific Research (NWO), and the Dutch Ministry of Economic Affairs. The work described in this paper has been supported in part by the European Commission through the ICT programme under contract ICT-2007-216676 ECRYPT II.

[1] Recently ATMs have become also vulnerable to malicious software, or malware, as they tend to use publicly available hardware, operating systems, and the Internet rather than exclusively private elements. See e.g., ATM crime: Overview of the European situation and golden rules on how to avoid it, ENISA, 2009 August.

First, the model and the main assumptions are described. Second, key exchange and identification are discussed in detail. Third, we discuss the relevant state of the art in this area. Applying our approach and model, Section 4 provides an analysis of two existing protocols and Section 5 reports and analyses our new protocol. Finally, Section 6 discusses conclusions and open questions.

2 Preliminaries

2.1 Cryptographic Tools

Since throughout this paper we use cryptographic concepts, this section aims to describe some relevant notions without their mathematical background. For a more complete description see e.g. [11].

Randomness plays a crucial role in cryptography and, instead of true randomness, computing devices generate pseudorandom values in practice. A *cryptographic hash function* creates a fixed length bit string, often called a hash value, from an arbitrary length bit string. Even if the function is deterministic, a hash value is seemingly random and reveals nothing about the input value. A *nonce* is a (pseudo)random number generated for using it only once in an instance of a cryptographic protocol. The bit length of a nonce usually depends on the system's security parameter.

Symmetric- and public-key encryptions and digital signatures are essential topics in cryptography,. While in case of *symmetric-key encryption* both the sender and the receiver of the message have the same key and therefore encryption and decryption can be performed by both of them, in case of *public-key encryption* only the receiver can decrypt by means of a secret key and the corresponding encryption key can be put publicly available. Since computing a secret key from the corresponding public key is intractable anyone can encrypt, but only the receiver can decrypt. A *digital signature* on a message is an appendix to the message that can only be produced by the signer but can be verified by anyone. The signer needs her private key, and the verifier needs the corresponding public key. As any small modification in the message essentially changes the signature, a digital signature provides authenticity (it is originated from the signer) and integrity (it has not been changed) for the message. A signature of participant A on a message m is denoted by $sig\{m\}_A$.

In order to establish a secure channel, both participants have to hold a symmetric key that enables them to encrypt and decrypt messages. Key establishment can be achieved by either a secure key exchange, or some key distribution mechanism. In this study we make use of the *Diffie–Hellman key exchange* [3] that enables two participants to create a shared secret key after sending certain messages over an insecure channel. Although applying the pure Diffie–Hellman key exchange is vulnerable to the MitM attacks, it can be protected by authenticating the messages during the key exchange.

In the context of public-key cryptography an important issue is how participants receive keys belonging to other participants. To verify a signature or to send a secret message, the public key is often certified by a trusted third party, the certificate authority, or CA. By signing other CAs' certificates, CAs can establish a hierarchy of public signature keys which is called a *public-key infrastructure*, or PKI.

2.2 TPM

A *trusted platform module*, or TPM [10] is the hardware root of trust within a computing device. Besides the TPM, a trusted subsystem of a host computer contains two software components:

A core root of trust for measurement (CRTM) and a trusted software stack (TSS). The CRTM, usually located within the TPM, is the first piece of software to run during the boot process, and the TSS communicates with the rest of the host platform and with other computers. A trusted subsystem can create attestation identification keys (AIKs) certified by a unique endorsement key, to sign its output. A trusted third party, the privacy certificate authority (Privacy CA) may attest in a certificate that an AIK was actually signed by a valid TPM's endorsement key. According to the specification, the TPM's actual identity (the public part of the endorsement key) is not revealed during an attestation. This enables the design of privacy-preserving protocols.

An *integrity report*, created by the trusted subsystem and the host platform, is a statement about the current state of a computer. When an integrity report is requested during a so-called *remote attestation*, the TPM by means of its AIK creates and signs a secure summary about the host computer's current hardware and software configuration.

In the course of the verification of an integrity report, a verifier follows several steps. It checks that the signature is valid and that the signing key (AIK) is certified by a Privacy CA. Moreover, it verifies that the current state of the platform occurs in a previously stored database of "good" states. In certain scenarios the verification requires big resources including large storage space (storing all "good" states of all computers in a system), many computations (verifying large measurement logs), and additional interaction with the Privacy CA.

3 Problem Statement

Verification of a computer is a process resulting in a report about the integrity of that device. It determines the state of the device and is able to detect whether the device contains malicious software, such as viruses or trojans. A special case of verification in which the computer is in the direct vicinity of an individual, is called *local verification*. Local verifications are often recommended to precede security or privacy critical actions. Examples of such actions are withdrawing money from an ATM, paying at a POS terminal in a supermarket, or using a public kiosk computer for internet access.

3.1 Model and Assumptions

A user U, having a trusted personal device M, such as a mobile phone, intends to verify whether a public terminal T is trustworthy enough to be used for a security or privacy sensitive task.

The user trusts his mobile device to perform all computations correctly, and to display the results honestly. Moreover, U relies on M's security, i.e., that it does not leak any information to a third party.

The terminal T is potentially untrusted. However, as in most proposed schemes, T is assumed to have a hardware TPM as the root of trust, which is able to perform integrity checks and to provide an integrity report about the current hardware and software configuration (see Section 2.2). Being a hardware module, T's TPM is able to reveal any software modification. T is presumed to be tamper-resistant and hence no adversary can make hardware alterations.

Although in some scenarios mobile phones can be suitable to verify integrity reports, we incorporate a trusted verification server S which is able to carry out this demanding task requiring much storage, computation, and communication. To convey the resulting signed statement, called an attestation about the state of the terminal in question, an authentic channel has to be established from the server to the user's trusted mobile phone via the untrusted terminal. (This authentic channel is established by a digital signature of the verification server, which is assumed to be efficiently verifiable, unforgeable, and it is assumed to provide message integrity.)

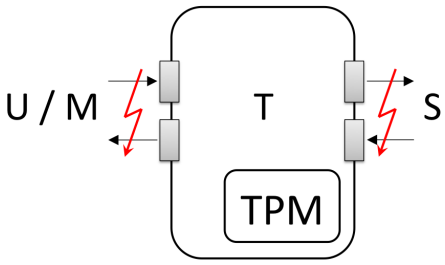

Fig. 1. Attacker model: The adversary has control over all digital channels.

Note that we do not assume a direct channel between M and S; however, we do presume that M stores S's public signature key or uses a matching PKI.

In our model the adversary is reasonably powerful as it controls all digital input and output channels between M and T and between T and S, see Figure 1; all messages transmitted through these channels can be eavesdropped on or modified by the attacker.

3.2 Approach: Key Exchange or Identification

The primary objective of a verification process is to assure the user U that a terminal T is not in a compromised state. However, the positive result of a verification does not necessarily guarantee that it does not originate from another terminal. Some previous protocols, such as [1] and [13], suffer from a potential man-in-the-middle attack. In this type of attack, an adversary places a hostile computer T' between the honest computer and the user. While the verification is done on the honest computer, the result is displayed on the hostile terminal. Consequently, the user is convinced that the hostile computer is honest.

We would like to draw attention to two methods to eliminate man-in-the-middle attacks by binding the communication to the honest terminal. A *key exchange* during the verification produces a secure key between the user's personal device and the honest terminal. This key is used to build a secure logical channel between these participants, so even if a man in the middle is present, it will not be able to extract or insert messages. The channel is tied to the verification result and hence to the honest terminal; therefore, it provides security for the main transaction. *Identification*, on the other hand, is a means to provide locality by determining that the device included in the verification is in fact the same as the one the user is physically interacting with.

Combining these two independent optional methods generates four different types of verification schemes to study (see Figure 2).

1. Schemes of type 1 do not establish a secure channel and provide no strong identification. Since the lack of these features results in the lack of a link between the verification and the main protocol, schemes of type 1 are vulnerable to man-in-the-middle attacks.
2. Schemes of type 2 provide assurance that the public terminal a user is standing at is indeed the honest computer and not a platform in the middle that relays all the messages between the honest host and the user. Although the main transaction does not require the use of a personal device, robust identifications may require to involve it for the verification. This would be a meaningful method for verifying ATMs before using them: (1) identification is obviously required because of the human interaction, and (2) the use of a secure channel is not possible since a user is required to communicate directly with the public terminal when typing the 4-digit PIN.

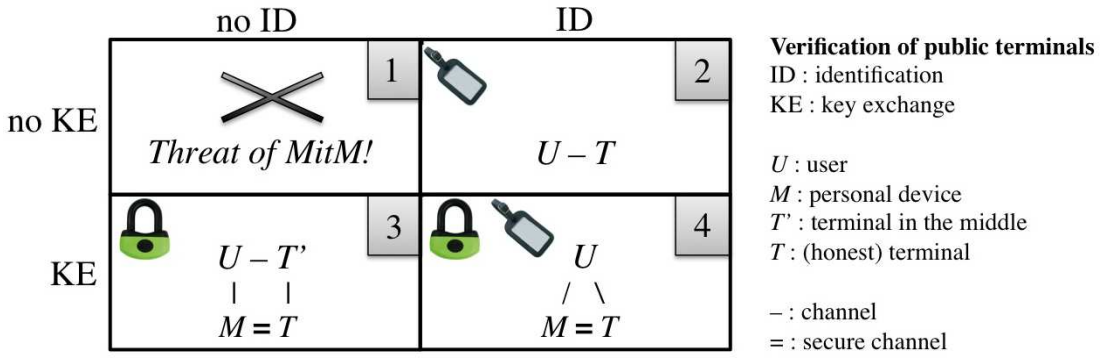

Fig. 2. Binding the verification to the main protocol: key exchange and/or identification.

3. Schemes of type 3 allow computing devices to establish a secure channel (e.g., SSL) during verification. This requires the user to apply a trusted personal device for the whole (secure) communication with the public terminal. As a positive verification result is bound to the honest terminal, the key of the secure channel is also bound to it. Therefore, even if there is a device T' in the middle controlled by an adversary, it is impossible for it to eavesdrop. Consequently, schemes of type 3 provide security without identification. As an example, an enhanced cash withdrawing protocol would enable the user to enter the PIN into her mobile phone (instead of directly into the ATM) that would send it over the secure channel to the honest ATM.
4. Schemes of type 4 establish a key between the user's mobile phone and the public terminal for the whole transaction and, simultaneously, they provide a technique for the identification of the public terminal in the vicinity. By linking the identity of the terminal and the key of the secure channel, these methods guarantee that the user and his personal device are communicating with the same public terminal.

Distinguishing these types helps to understand the requirements of different scenarios and the essential differences among proposed schemes in the literature.

3.3 Requirements

Consider the system model in Section 3.1. A verification scheme must provide an authentic and verifiable integrity report about the current state of the public terminal with the following properties:

- *Completeness*: An honest terminal should be able to convince a verification server and eventually the user about its good state.
- *Soundness*: A cheating terminal should not be able to convince either a verification server, or a user that it is honest.

Furthermore, the scheme must bind the verification to the transaction by one of the methods described in Section 3.2, i.e., it has to be of type 2, 3, or 4.

3.4 Related Work

Several schemes have been proposed in the literature for integrity reporting and locality verification. We can classify proposals using the four types described in the previous section.

Recently, Toegl and Hutter [13] and Bangerter et al. [1] propose verification schemes that output the state of a public terminal using its TPM. Even though these protocols claim to provide identification of a public terminal, they do not as both schemes are vulnerable to man-in-the-middle attacks (see Section 4). Consequently, these schemes are of type 1 instead of type 2.

McCune et al. propose *Seeing-is-Believing* (*SiB*) [7] techniques for demonstrative identification of previously unknown devices by applying bar codes and cameras. SiB protocols, which can securely associate a trustworthy device identifier with the present communication, are able to assure the user that a man-in-the-middle attack does not take place. McCune et al. discuss TPM related protocols, in which identification is based on the TPM identifier which is welded securely on the cover of the host platform. This identifier and the one that provides authenticity in the integrity report are compared and expected to be the same by the user's mobile device. This results in a verification protocol, which is also included as a sub-protocol in our proposal (see Section 5), of type 2.

Locality can also be achieved by distance bounding, which was devised by Brands and Chaum [2] and is still an important topic of research [6]. Distance bounding guarantees an upper bound on the physical distance between the user and the public terminal.

Stumpf et al. [12] show that several remote attestation protocols are vulnerable to man-in-the-middle attacks. To protect against this, they propose a new scheme that they call *enhanced integrity reporting*. This method is based on a simple yet essential observation: The actual goal of a remote attestation (and verifications, in general) is not the valid integrity report for its own sake, but a secure communication that is *bound* to the computer that provides it. Therefore, in the protocol of enhanced integrity reporting, remote attestation is complemented by a session key establishment between the verifier server and the terminal. This key is then used to secure the channel for the whole communication. Although Stumpf et al. study the problem of remote attestation and therefore their scheme does not tackle identification of a local computer resulting in a scheme of type 3, it can easily be incorporated into the verification of public terminals as we show in our proposal (see Section 5).

Garris et al. [5] describe a specific scenario of type 4 in which a user can interact with a public kiosk computer using a trusted mobile phone for encrypting all personal information. The scheme, which uses the SiB concept for identification and TPM technology for verification, enables the user to verify a public computer before using it. This protocol is closely related to our proposal; however, it is not practical in some scenarios (e.g., ATM, POS terminal) as it requires the public computer to reboot during an execution of the protocol and to run a virtual machine. Oprea et al. [8] also design an application of type 4, that is, it guarantees both a secure channel and identification. Their scheme is not just an initial verification for some main interaction, but a complex protocol. It applies two SSL channels for scenarios in which users access a trusted remote computer, such as their home computer, from an untrusted terminal.

In sum, though specific protocols of different types were designed that bind a computer with the secure communication, no general description exists to prevent an adversary from performing man-in-the-middle attacks. In particular, there does not exist a scheme of type 2, 3, or 4 that can provide secure verification in the context of public terminals.

4 Analysis of Existing Schemes

A secure verification scheme does not only provide a trustworthy report about the public terminal in question, but also guarantees that the transaction is properly bound to the verified device. To analyse the design of a verification scheme, we perform the following steps with respect to its security and its binding procedure:

– Determine the type of verification the scheme claims to perform
  - If it is of type 1 (i.e., no identification, no key exchange), then it is vulnerable to a man-in-the-middle attack.
  - If it is of type 2, 3, or 4, then its security has to be studied further.
– Verify security of the messages: proper freshness (to prevent replay attack), inclusion of the identifier of the terminal (to prevent the use of another terminal's report), authentic evaluation result (to prevent a rogue evaluation).
– Verify security of the channels (depending on the type): identification (channel between the user and the terminal) and/or key exchange (channel between the personal device and the terminal) are properly bound to the transaction.

After studying several schemes, we found that proposed verifications of public terminals often fail to provide robust solutions. Below we describe the analysis of two schemes briefly.

The objective of Toegl and Hutter's elegant scheme (Sections 4 and 5 in [13]) was to build a verification protocol of type 2, that is, it should incorporate identification. In their work, locality is provided by using a direct near-field communication (NFC) channel from the TPM to the user's mobile device – according to Parno's [9] suggestion in his theoretical work. They proposed a design modification in the TPM specification by extending it with an NFC interface. Nevertheless, another recent study by Francis et al. [4] shows that NFC is vulnerable to man-in-the-middle attacks, so its use for proving locality is not secure enough. The scheme can be enhanced for each verification type in the following manners:

– Type 2: Apply mitigations against NFC man-in-the-middle relay attacks by location-based information, distance-bounding, or multi-channel communication [4]; however, those are not always possible.
– Type 3: For communicating S's verification result, a secure channel is established between M and T's TPM in the scheme. Either the protocol should be extended and the honest host terminal should learn this key, or the TPM should be involved during the whole secure communication.
– Type 4: By combining the security and locality extensions above, the scheme can be reinforced to be of type 4.

Bangerter et al. [1] proposed a scheme that applied the proprietary AXS system[2], which uses special flickering images, for secure communication from S to M, while T is used only for conveying encrypted messages. Their protocol aims to produce a type 2 verification, but it also turns out to be vulnerable against man-in-the-middle attacks in our model. Our attack works as follows. As the adversary has control over all input and output channels of the terminal, messages from the server can be captured and displayed on the monitor of a dishonest terminal. Even if the flickering image cannot be relayed, as Bangerter et al. assume, the data is seized on a lower level. Because the terminal's identifier is not included in the protocol, there is no way for the user to detect this man-in-the-middle attack. If the protocol is extended by an adequate identification procedure of the terminal, it does become a verification of type 2.

[2] The AXS system contains hardware and software elements. Each user has a small electronic device, the AXS token, that has two input channels (an optical sensor and a fingerprint reader) and an output channel (display). The software element is at the server's side and its main output is an unforgeable flickering pattern that can be sent and displayed on the client side. AXS tokens can read and interpret these patterns as numbers.

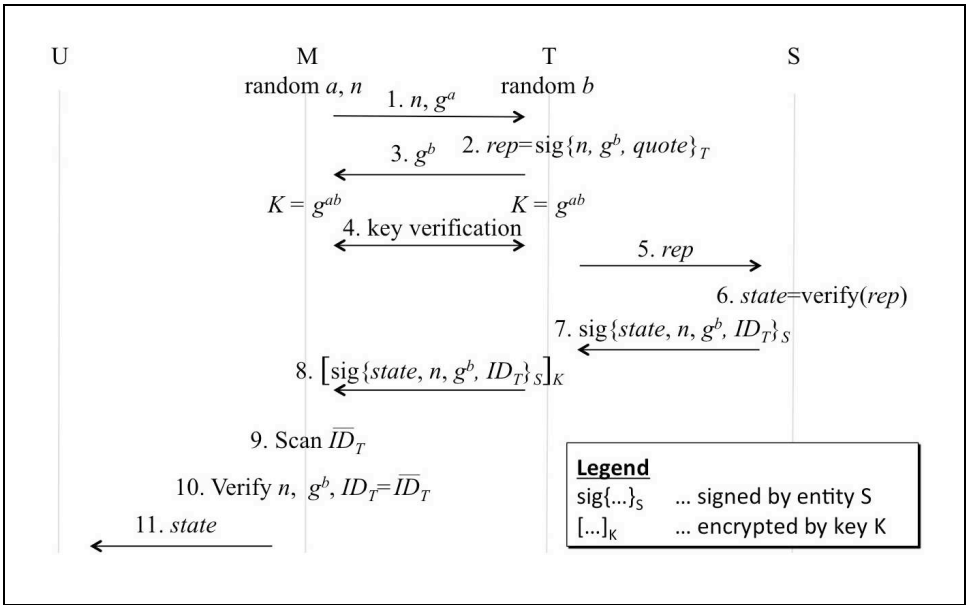

Fig. 3. Protocol for verifying a public terminal.

5 General Proposal

In this section, we propose a resilient verification protocol for public terminals that can provide key exchange (type 3) and optionally identification (type 4) together with the terminal verification process. First, we present a set of requirements that the protocol has to meet, then we show our scheme, and finally, we give a brief security analysis.

5.1 Verification Scheme

Our proposal, shown in Figure 3, is based on two previous proposals. Firstly, the scheme incorporates key exchange that provides a secure channel for further communication according to Stumpf et al.'s robust integrity reporting [12]. Secondly, the scheme applies the SiB principle (see 3.4 and [7]) that enables the identification of the public terminal. A brief description of the scheme is as follows:

1–4 A fresh integrity report is generated with an additional Diffie–Hellman key exchange (in a standard cryptographic setup) as in [12]. (Besides the nonce n and the DH key share g^b, the integrity report *rep* contains a *quote* that describes the measurement of the current state, the public key of the terminal, and a certificate about the AIK that is used to sign the whole report.)

5–8 Having received the integrity report, the server evaluates it ("good" state, signature, key, etc.) and creates a signed message containing the state and the identity of T as well as the nonce.

9–10 (a) Depending on whether direct U–T interaction is necessary in the main transaction or not, the scheme does or does not include identification. In the first case the user, by means of his mobile device, employs SiB techniques to scan and compare the terminal's identifier ID_T from the signature and that on the outside of the terminal ($\overline{ID}_T$). In the latter case,

in which no direct U–T interaction is necessary in the main transaction, the user does not initiate identity verification. (b) To verify freshness, the mobile device confirms that the nonce in the attestation and the original nonce from step 8 are equal. Furthermore, the mobile device also checks equality of the terminal's public channel key g^b from step 3 and 10.

11 Finally, after validating freshness, channel security, identification, and T's verified state report, M displays the result to U whether T is in a trustworthy state.

As a result of this verification protocol, the user knows the state of the public terminal; moreover, M and T have a session key $K = g^{ab}$ that establishes a secure and authentic channel between the mobile device and the honest terminal. If identification supplements the verification (in steps 9–10), the user is also convinced that the honest terminal is the one directly in front of him.

5.2 Security Analysis

Although a comprehensive security analysis has not been carried out in this research, we argue that the protocol above is of type 3 or type 4, depending on the exclusion or inclusion of identification, and it meets the security requirements described in Session 3.3. Basically, there are two cases according to the state of the platform.

If terminal[3] T with its TPM cannot produce a valid integrity report, i.e., either the TPM's signature is not valid, or the state is not in the set of "good" states, then the verification server will output "bad" as the terminal's state in step 6. Since the state – together with the nonce and the terminal's identifier – is sent signed by S, it is impossible to change the values without knowing S's private signature key. Eventually, M will display to the user that T should not be used for processing sensitive data. As a result, we get that the scheme is *sound*, that is, no malicious terminal T' can convince U that it is honest.

If T can produce a valid integrity report, there are further evaluations. If the scheme includes the identification steps in 9 and 10, the ID_T in S's signed message and the identifier $\overline{ID}_T$ on the casing of the public terminal have to match. Moreover, the nonces generated by M in step 1 and signed by S in step 7 should be the same as it is verified in step 10. As a result we get that an honest terminal can convince U that it is honest; therefore, the scheme is *complete*.

We show that the scheme without the identification method in steps 9 and 10 is of type 3. First, we show that the channel established in steps 1–4 is secure between M and the honest terminal T. We need to prevent an attacker to break the channel key, replay previous communications, and being able to mischievously have M establish a key with T' instead of T. Since a and b are only known by M and T, respectively, according to the Diffie–Hellman assumption[4], the key of the secure channel $K = g^{ab}$ can only be computed by these participants. Furthermore, the nonce freshly generated by M (see step 1) prevents replay attacks. Finally, T's share g^b is included in the integrity report (see steps 2 and 8), therefore it is bound to T. The key verification in step 4 guarantees that the two participants share the same key. Second, we show that it does not provide identification. It is easy to see that a terminal T' is able to impersonate T by relaying all messages between M and T (although T' can only see encrypted messages). Therefore, the scheme is of type 3.

By incorporating the identification in steps 9 and 10, the scheme is of type 4 since it provides not only secure communication between M and T as above, but also between U and T. Since ID_T as well as T's key share g^b are included in the integrity report, M and U communicate with the same honest terminal.

[3] Throughout this section, we refer to the honest terminal as T, while to an adversarial terminal as T'.

[4] Informally, the Diffie–Hellman problem is to compute g^{ab} from g, g^a, and g^b without knowing a or b. The Diffie–Hellman assumption states that in certain groups this problem is hard.

6 Conclusions

We have considered the security problem of verification of a public terminal prior to sensitive communication between an individual and the terminal. The findings suggest that two independent security methods, identification and key exchange, can prevent man-in-the-middle attacks.

While the strongest schemes provide both methods, one of them is often sufficient. When direct user interaction is required with the public terminal, key exchange, which is computationally intractable for humans, is not possible. Therefore, in this case terminal identification is preferred. On the other hand, when a private mobile device can be employed for conveying all sensitive information, prior key exchange without actual identification is often adequate.

In order to put the problem of verification of public terminals in a broader context, we construct an overview table (see Table 1) of scenarios that illustrates eight different alternatives depending on which of the three channels are present for sensitive communication in the main transaction between a user U, a personal device M, and a public terminal T. They characterise distinct scenarios, security and channel requirements, and user experience.

U–M	U–T	M–T	Description	Requirment	Example
0	0	0	No comm. among agents	-	-
0	0	1	U is not included	-	-
0	1	0	Only U–T	ID (M–T in verification)	ATM
0	1	1	No U–M channel	ID+KE	M is a smart card
1	0	0	T is not included	-	-
1	0	1	No U–T channel	KE	ATM with secure PIN entry
1	1	0	No trust can be built	low entropy messages	one-time password
1	1	1	Full trust	ID+KE	both U–T and M–T secure

Table 1. All possible scenarios for single-channel communication between a user U, a mobile device M, and a public terminal T.

We have described a way to analyse verification schemes and, in order to demonstrate the analysis, we have studied existing schemes and found that some of them failed to provide either of these methods. We made recommendations to rectify those.

We have designed a generic user-friendly and privacy-friendly protocol that provides both methods to accompany the verification. The protocol design is modular in the sense that, according to the requirements, besides establishing a secure channel identification can be added or removed.

6.1 Further Research

Although this research can be enhanced in several directions including the verification of not only public terminals but also other (mobile) devices or the whole environment a mobile device is placed, we focus on two interesting questions.

As the personal device is often a mobile phone in practice, the presumption that there is no independent channel between M and S may be eliminated. In this case new protocols can be designed in which the server can communicate the evaluation result easier as the untrusted public terminal is excluded.

As mobile phones are ever more disposed to malicious software, the trust assumption about M can be questionable. Using the model in which T and M are both untrusted but independent (i.e., assuming that they are not attacked by the same adversary), new protocols can be designed to verify whether T is in a valid state.

7 Acknowledgements

The authors are grateful to Rieks Joosten for stimulating discussions, advices on writing, and insightful comments, and to Maarten Everts for his helpful remarks.

References

1. Endre Bangerter, Maksim Djackov, and Ahmad-Reza Sadeghi. A demonstrative ad hoc attestation system. In Tzong-Chen Wu, Chin-Laung Lei, Vincent Rijmen, and Der-Tsai Lee, editors, *Information Security*, volume 5222 of *LNCS*, pages 17–30. Springer Berlin / Heidelberg, 2008.
2. Stefan Brands and David Chaum. Distance-bounding protocols. In Tor Helleseth, editor, *Advances in Cryptology — EUROCRYPT '93*, volume 765 of *LNCS*, pages 344–359. Springer Berlin / Heidelberg, 1994.
3. W. Diffie and M. E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, IT-22:644–654, November 1976.
4. Lishoy Francis, Gerhard Hancke, Keith Mayes, and Konstantinos Markantonakis. Practical nfc peer-to-peer relay attack using mobile phones. IACR Eprint archive, April 2010.
5. Scott Garriss, Rámon Cáceres, Stefan Berger, Reiner Sailer, Leendert van Doorn, and Xiaolan Zhang. Trustworthy and personalized computing on public kiosks. In *Proceeding of the 6th international conference on Mobile systems, applications, and services*, MobiSys '08, pages 199–210, New York, NY, USA, 2008. ACM.
6. Chong Hee Kim, Gildas Avoine, François Koeune, François-Xavier Standaert, and Olivier Pereira. The swiss-knife rfid distance bounding protocol. In Pil Lee and Jung Cheon, editors, *Information Security and Cryptology – ICISC 2008*, volume 5461 of *LNCS*, pages 98–115. Springer Berlin / Heidelberg, 2009.
7. Jonathan M. McCune, Adrian Perrig, and Michael K. Reiter. Seeing-is-believing: using camera phones for human-verifiable authentication. *International Journal of Security and Networks*, 4(1-2):43–56, 2009.
8. Alina Oprea, Dirk Balfanz, Glenn Durfee, and D. K. Smetters. Securing a remote terminal application with a mobile trusted device. In *In ACSAC*, pages 438–447, 2004.
9. Bryan Parno. Bootstrapping trust in a "trusted" platform. In *Proceedings of the 3rd conference on Hot topics in security*, pages 9:1–9:6, Berkeley, CA, USA, 2008. USENIX Association.
10. S. Pearson, editor. *Trusted computing platforms: TCPA technology in context*. HP Professional Series. Prentice Hall PTR, 2003.
11. N.P. Smart. *Cryptography, An Introduction (3rd edition)*. on-line, http://tinyurl.com/yeafjcx, 2011.
12. Frederic Stumpf, Omid Tafreschi, Patrick Röder, and Claudia Eckert. A robust integrity reporting protocol for remote attestation. In *Second Workshop on Advances in Trusted Computing (WATC '06 Fall)*, pages 25–36, Tokyo, Japan, November 2006.
13. Ronald Toegl and Michael Hutter. An approach to introducing locality in remote attestation using near field communications. *J. Supercomput.*, 55:207–227, February 2011.