ELSEVIER

Computer Networks 31 (1999) 861–870

COMPUTER NETWORKS

Authenticating public terminals

N. Asokan [1], Hervé Debar [*], Michael Steiner [2], Michael Waidner [3]

IBM Research Division, Zurich Research Laboratory, Säumerstrasse 4, CH 8803 Rüschlikon, Switzerland

Abstract

Automatic teller machines, Internet kiosks etc. are examples of public untrusted terminals which are used to access computer systems. One of the security concerns in such systems is the so called *fake terminal* attack: the attacker sets up a fake terminal and fools unsuspecting users into revealing sensitive information, such as PINs or private e-mail, in their attempt to use these terminals.

In this paper, we examine this problem in different scenarios and propose appropriate solutions. Our basic approach is to find ways for a user to authenticate a public terminal before using it to process sensitive information.

Keywords: Authentication; Mobility; Fake terminal attack; Internet kiosks

1. Introduction

Consider withdrawing money from an automatic teller machine (ATM) using a bank card. In all existing systems, users have to enter a personal identification number (PIN) or pass-phrase in order to reliably authenticate themselves to the bank. However, this authentication is one-way only; there is no way for the user to authenticate the bank. There have been incidents where thieves set up fake ATMs and successfully stole PINs and magnetic stripe information from unsuspecting users [3].

The same *fake terminal* problem occurs in many other settings. We consider the following examples.

- **ATMs and point-of-sale terminals**: In both scenarios, every user is registered with a specific server (e.g., a credit-card issuer). All transactions of the user are eventually authorized by the server. Servers can typically identify and authenticate legal terminals. A typical attack scenario is when the attacker would set up an illegal terminal which waits for the user to type in the PIN code, read any necessary information from the card, and then refuse service, for example by displaying a ''terminal out of order'' message. Unsuspecting users will simply move on to a different terminal. The attacker can later use the stolen information at a legal terminal.
- **Public Internet kiosks**: Short-term access to the Internet from public terminals is an increasingly common feature in malls, airports, the so-called

[*] Corresponding author. E-mail: deb@zurich.ibm.com.

[1] With Nokia Research Center, Helsinki from January 1999. Email: asokan@acm.org.

[2] E-mail: sti@zurich.ibm.com.

[3] E-mail: wmi@zurich.ibm.com.

1389-1286/99/$ - see front matter
PII: S1389-1286(98)00020-6

''Internet cafes,'' and other public places. There is little risk for users who merely want to ''surf'' the web from these terminals. But people can, and do, perform more sensitive transactions such as accessing their personal or business computer systems, making payments, etc., from public Internet kiosks. This scenario differs from the previous ones in some respects:

- the user may access several servers from the same terminal, and
- the types of private information which needs to be protected may not be fixed, or even known a priori.

A similar scenario arises in the case of *virtual mall kiosks*. Virtual mall kiosks allow prospective customers to browse through and purchase the wares advertised by shop-keepers in the virtual mall. Functionally, this scenario is similar to public Internet kiosks.

In specific settings, such as ATMs that use biometrics instead of password to authenticate, the fake terminal problem can be avoided. However, the general problem remains. A solution to this general problem must take into account different scenarios where the resources available to a user may be different: a user may have a trusted personal device with its own display or may have only a standard smartcard with no display attached or, in the simplest and most common case, may not have any personal trusted device at all. In this paper, we describe methods to solve this problem for each of these scenarios. Our solutions are achieved by combining known techniques in novel ways.

The paper is organized as follows. In Section 2, we describe our model and state our goals. In Section 3 we describe our techniques in detail. In Section 4 we survey previous work on related issues. Finally, in Section 5, we conclude with a summary and directions for future work.

2. Model

We target the following general model. Users access server systems from public untrusted terminals. Users have accounts on a central server system which they trust to correctly authenticate public terminals. These terminals are tamper-resistant but an attacker can easily replace a legal terminal with a fake terminal or install a new fake terminal in a plausible location. The central system knows about legal terminals and can authenticate them. Information necessary for users to authenticate the central server (and, where necessary, information needed for the central server to authenticate users) has already been set up during user registration or other initialisation steps (e.g., agreeing on a shared key). Once an entity authenticates another, a confidential, authenticated channel is established as a result. In other words, an attacker cannot hijack a channel resulting from the authentication procedure. We use the symbols U, T, and S to identify a user, a terminal and a central server respectively. When the user has a trusted personal device, it is denoted by D. The notation is illustrated in Fig. 1.

The authentication steps mentioned above are implemented using authentication protocols. There are various well-known authentication protocols for performing both one-way and two-way authentication such as Secure Sockets Layer (SSL) [7], KryptoKnight [5], and Kerberos [10]. The solutions we propose below assume the use of a suitable authentication protocol.

The central server may be replicated thereby avoiding it from becoming a bottleneck. All copies

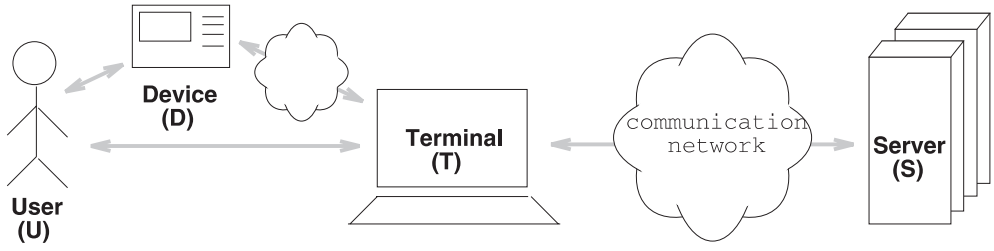

Fig. 1. Model.

of the central server need to be aware of the up-to-date set of legal terminals and the information necessary to authenticate them. There may also be *several* central servers, each responsible for a separate *domain*. In this case, we assume the existence of the infrastructure necessary for central servers to authenticate each other (e.g., a public-key infrastructure). In either case, the number of terminals is likely to be several orders of magnitude higher than the number of servers.

3. Terminal authentication protocols

3.1. Case 1: Personal device with built-in output capability

First we consider the scenario where a user has a full-fledged trusted personal device with its own output channel, such as a display screen. The terminal cannot access the device output channel. Consequently, the user can be sure that any information is communicated to him via this channel does in fact originate from his trusted personal device. In other words, there is a *trusted path* from the trusted personal device to the user. When a user U walks up to an untrusted terminal, he attaches his device D to the terminal T by some means (e.g., infrared link, physical connection) and the following message flows take place.

1. U → D: U requests D to authenticate the terminal it is attached to (e.g., by clicking on a button on D's display).
2. D → T: D requests T to authenticate itself to S.
3. T → S: T runs a one-way authentication protocol to S. If this succeeds, S knows that it has an authenticated channel S–T to T.
4. S → D: S runs a one-way authentication protocol to D via S–T. If this succeeds, D knows that it has an authenticated channel S–D to S, which is tunneled through S–T.
5. S → D: S sends a message to the effect ''T is authentic'' via S–D. In addition, S sends additional information (such as a session key, or one-time certificates) that can be used by D and T to construct a secure channel D–T between themselves.
6. D → U: D displays a message to the effect ''T is authentic according to S'' to U.

As mentioned before there are various well-known authentication protocols that may be used for the one-way authentication flows above (as well as in the scenarios below). In step 3, T could run a two-way authentication protocol. This would foil an attacker masquerading as S. In scenarios where U has to authenticate to S, it can be done in a separate phase following the above exchange or step 4 can be a mutual authentication exchange. In this case, D may need to ask U to provide authentication information (e.g., a pass phrase or PIN) in step 4. Notice also that so far U is not identified to T or S. This helps to keep the itinerary of U confidential from T.

This is essentially an application of the usual three-party authentication protocols [5,10,15]. Abadi et al. describe an authentication and delegation protocol using ''ultimate smartcards'' [2]. This protocol is similar to the one outlined above.

Notice that at the end of the protocol, D has succeeded in authenticating T. But U has no guarantee that the terminal in front of him is the same terminal that has been authenticated by D (for example, an attacker may have placed a fake terminal which is connected to a legal terminal elsewhere). There are various ways of supplementing our protocol to thwart this *fake-terminal-in-the-middle* attack. Abadi et al. mention that the device (D) gives ''enough information'' to the user to check whether the keyboard and display in front of him actually belongs to the terminal that was authenticated. We outline two concrete techniques to do this.

First, at the end of the above protocol, D could generate some random bits, communicate them to T via D–T, and then both D and T could display the random bits to the user on their respective displays. The attacker cannot figure out the random bits from D–T because it is a confidential channel. However he might be able to read it off T's display and present it on the display of his fake terminal.

An alternative approach is for the user to verify that T is physically close to D. If the user or his device is capable of verifying his location (e.g., if D is equipped with a Global Positioning System receiver), then S can indicate T's location in the message in Step 5. Brands and Chaum described *distance-bounding protocols* [4] which can be used

by a verifier to place an upper bound on its distance from a prover. D and T can employ such a protocol as part of the above protocol.

3.2. Case 2: Personal smartcard without output capability

Now we consider the scenario where a user is equipped with a device, such as a smartcard, which has no output capability. We could try to use the same solution for this scenario as well. However, the problem arises in step 6: D does not have its own display; consequently it does not have a trusted path to U. There may be devices with other types of trusted paths (e.g., mobile phones could use a speech synthesizer to communicate the message to U), in which case the previous solution could still be used. Standard smartcards, however, have no such output mechanism. Hence we need to modify the technique.

Customizing security-critical windows is a well-known security measure against Trojan horse attacks. There have been various proposals, see Refs. [1] and [18]. Some variants have also been implemented, for example in the SEMPER Trusted INteractive Graphical User INterface (see url: www.semper.org), or the hieroglyphs in the Lotus Notes login dialog-box. While it is an effective countermeasure against simple-minded Trojan horses, it is ineffective in a scenario where the Trojan horse has read and write access to the display. As soon as a personalized window is displayed to the user, the Trojan horse program can read the personalization information, construct a fake window with the same information on top of the legitimate personalized window [4].

However, currently we are considering a limited threat model where ''legal'' terminals are tamper-resistant while ''illegal'' terminals will not be able to authenticate themselves to the central server. We combine the personalization idea with authentication protocols to achieve an effective solution for this threat model. In our protocol, the personalization information is not revealed to the terminal before the it has been authenticated by the central server. This way, we can protect even against sophisticated fake terminal attackers as long as they are subject to the constraints of our threat model. In our current work (see Section 5) we consider the stronger threat model in which an attacker *may* subvert legal terminals by, for example, installing Trojan horses.

We assume that the user has a trusted (home) base (such as a home PC) where he can prepare his smartcard before beginning his travel. For the preparation, the user selects an *authentication vector*. An authentication vector consists of one or more types of *authenticators*. An authenticator of a particular type is such that

- it can take one of several values,
- each different value can be perceived by an unaided human and distinguished from other values.

Examples of types of authenticators are:

- background colour (of the order of 256 possible values),
- foreground colour (of the order of 256 possible values),
- background pattern (of the order of 16 different patterns),
- sound sequence (of the order of 256 different tunes).

Another example is to include text phrases that can be easily recognized by the user. A variety of means could be employed in order to show the words to the user: e.g., visual (by printing them on a screen), aural (by using a speech synthesizer) or tactile (by ''displaying'' the words in braille). Words and phrases constitute the most powerful type of authenticators since (a) they can be drawn from a relatively large space, and (b) they can be communicated to the user in a variety of ways.

To prepare for his travel, the user picks one combination: for example, a tuple of the form

phrase = abracadabra, bg-color = blue,

fg-color = white, bg-pattern = grid,

tune = jingle-bells

on his trusted home base and stores it on the smartcard. When the user walks up to an untrusted terminal and inserts his smartcard into the reader, the following message flows take place:

1. U → T: U requests T to authenticate itself to S (e.g., by typing in the identifier of S and clicking on a button on T's display).

[4] The Lotus Notes hieroglyphs cannot be spoofed this way because the hieroglyphs depend on the password. But the security of the scheme relies on the algorithm for mapping password characters to hieroglyphs remaining secret.

2. T → S: T runs a one-way authentication protocol to S. If this succeeds, S knows that it has an authenticated channel S–T to T.
3. S → D: S runs a one-way authentication protocol to D via S–T. If this succeeds, D knows that it has an authenticated channel S–D to S which is tunneled through S–T.
4. S → D: S sends a message to the effect ''T is authentic'' via S–D. In addition, S sends additional information (such as a session key, or one-time certificates) that can be used by D and T to construct a secure channel D–T between themselves.
5. D → T: D reveals the pre-selected authentication vector to T.
6. T → U: T shows the received authentication vector to U (e.g, by displaying the selected colours and background pattern, and playing the selected tune).

In other words, D reveals the authenticator to T only after S has certified that T is a legal terminal. The probability of an illegal terminal correctly guessing the authenticator of a user is very small (e.g., of the order of one in $256 \times 256 \times 16 \times 256$ with the parameters suggested above). If a rogue terminal incorrectly guesses the authenticators of several users in close succession, it will likely be reported to the authorities and thus detected as an illegal terminal.

Notice that so far U is not identified to T or S. This helps to keep the itinerary of U confidential from T. As before, a distance-bounding protocol can be used to avoid the possibility of the fake-terminal-in-the-middle attack.

3.2.1. Variations

The following variations are possible:

- Smartcards may not have sufficient memory to store an authenticator in its entirety. However, if the types of authenticators are pre-defined, the smartcard needs to store only an index: the terminal can use the index to look up the authenticator in a table of all possible values for the different components.
- Some smartcards may not be writable by the user. In this case, the following modifications are made:
  - In the preparation phase, the user selects the authentication vector and communicates it to the server via a confidential, authenticated channel from his home base.
  - In the protocol above, steps 5 and 6 are replaced as follows:

    5-V D → S: D authenticates to S via S–D.

    6-V S → T: If the authentication succeeds in step 5, S sends the pre-selected authentication vector to T via S–T.

    7-V T → U: T shows the received authentication vector to U.

  Authentication is necessary in step 5 because S must not reveal the authentication vector to an attacker who is using a legal terminal but pretends to be a user U.
- The same authentication vector could be used several times. The user could also select a set of authentication vectors during the preparation phase. Another variation is where the user challenges T to show a different component of the authentication vector each time. This will also help foil an attacker who ''peeps over the shoulder'' of a legitimate user and learns his authentication vector.
- As before T could run a two-way authentication protocol with S (Step 2). This would foil an attacker masquerading as S.

There can be scenarios that are between Case 1 and Case 2. For example, there are hand held smartcard readers with the single line display. This is not as powerful as a device with a dedicated display available at all times during the authentication protocol. However, it is still powerful enough to afford the following simple solution. Just before starting a session at a public terminal T, U inserts his smartcard D into his personal reader. D chooses a random string and displays it in the reader's display. This string will be used as the authentication vector in the protocol described above for Case 2. The random string is similar to the use of smart-card device identification numbers (DIN) described by Pfitzmann et al. [17], except that the DIN is permanently associated with the device unlike the random string above which is chosen afresh for each session.

3.3. Case 3: No personal device

Smartcards and personal trusted devices may become commonplace in the near future. But to date, their use is relatively limited. Most users are armed

only with simple pass-phrases (e.g., in the case of Internet access) or memory cards (e.g., in the case of credit/debit cards which have memory capacity in the form of magnetic stripes or memory chips but do not have any processing capability of their own). In this section, we investigate this scenario in which the user has no personal computing device at all.

For this we adapt a solution for one way authentication called S/Key [8]. In the S/Key system, the server issues a number of challenge/response pairs to the user during an initialisation stage. The user prints out the list of these pairs. The responses are essentially one-time passwords. In order to access the system, the user identifies himself and the server sends a challenge. The user then looks up the appropriate response from his printed list, sends it back to the server, and strikes off that pair from his list. We use an S/Key like system in *both* directions.

Before beginning his travel, S sends a number of challenge/response pairs to the user via a confidential, authenticated channel to his home base and the user selects a different authentication vector for each challenge and sends them back to S. The user also prints out the entire list of $< challenge, response, authentication vector >$ triples.

When the user walks up to an untrusted terminal, the following message flows take place:

1. U → T: U requests T to authenticate itself to S (e.g., by typing in the identifiers of U and T, and clicking a button)
2. T → S: T runs a one-way authentication protocol to S. If this succeeds, S knows that it has an authenticated channel S–T to T.
3. S → T: S sends a challenge via S–T to T.
4. T → U: T displays the challenge to U.
5. U → T: U looks up the response corresponding to the challenge on his printout and types it in, provided it is not already struck off.
6. T → S: T sends the response via S–T to S.
7. S → T: If the response is valid, S looks up the authentication vector corresponding to the challenge, and sends it via S–T to T.
8. T → U: T shows the received authentication vector to U.

U can verify if this is indeed the authentication vector corresponding to the challenge, according to his printed sheet. If so, he can be confident that T is a legal terminal. U then strikes off the entry corresponding to the challenge from his printed list. If the authentication fails, U as well as S should still cross out the entry corresponding to that challenge and never use it again. As before T could run a two-way authentication protocol with S (step 2). This would foil an attacker masquerading as S.

Since there is no user device, techniques described earlier to avoid fake-terminal-in-the-middle attacks are not applicable in this scenario. We do not know of an effective way to reliably foil this attack in this scenario. Distance-bounding protocols are not suitable because any distance-bounding protocol must be executed between the terminal and the human user, which is impractical. The goal is to prevent the attacker from capturing the output on the legal terminal display and showing it on his fake terminal display. He may do this by capturing the electromagnetic radiation from the legal terminal and display it on his fake terminal. ''Soft Tempest'' techniques described by Kuhn and Anderson [9] may be used to foil this attack. The attacker may also simply place a video camera in front of the legal terminal display. With standard cameras the user will notice the unavoidable refresh rate mismatch in the form of ''rolling lines.'' But a more sophisticated attacker operating with cameras of high refresh rate and high resolution can smooth the effect through signal processing. Further work is required to identify and develop effective techniques.

3.3.1. Variations

The user may want to avoid carrying around a printed list. It can also be a security weakness: if the attacker manages to get hold of the printed list, he can fool the user and/or the central server. In this case, he can make do with a single authentication vector. Steps 3–6 above are dropped. In step 7, S sends the authentication vector to T without any further checks. Obviously this simplification is not secure against targeted attacks where the attacker obtains the authentication vectors of specific users (e.g., by interacting with a legal terminal), sets up a fake terminal, and waits for these users to come in. But it is useful against untargeted attacks (i.e., setting up a fake terminal without specific users in mind). If users change their authentication vectors regularly, large scale targeted attacks are not feasible.

Table 1
Notation summary

Notation	Explanation
n	Size of the entire authenticator space
m	Number of transactions before a fake terminal is detected and removed
p	Probability that the fake terminal cannot correctly guess the authentication factor of even a single transaction out of m

A second variation is, as in the previous scenario, we can allow the user to challenge T to show a different component of the authentication vector each time: i.e, the user specifies the type of the authentication vector as the challenge since it may help the user remember the challenges. For example, it is easier for a user to remember a colour, a tune, and a word rather than to remember three colours.

Note that a user need not necessarily *remember* his entire authentication vector, but need only be able to *recognize* incorrect authentication vectors. One possibility to construct authentication vectors with high entropy is to arrange them by themes. For example, the user could issue a challenge on the theme ''car,'' and ask for specific attributes of his car. A car has several attributes which are easy to recognize.

3.4. Analysis

Suppose an attacker installs a fake terminal which is used for m transactions before it is detected and removed. Let p be the probability that the terminal was not able to correctly guess the authentication vector in *any* of the m transactions. Let n be the total size of the authenticator space (the notation is summarized in Table 1). If we assume that the authentication vectors for the m transactions were independent,

$$p = \left(1 - \frac{1}{n}\right)^m$$

.

We can set p and m as system parameters and ask how big n should be. If the expected rate of

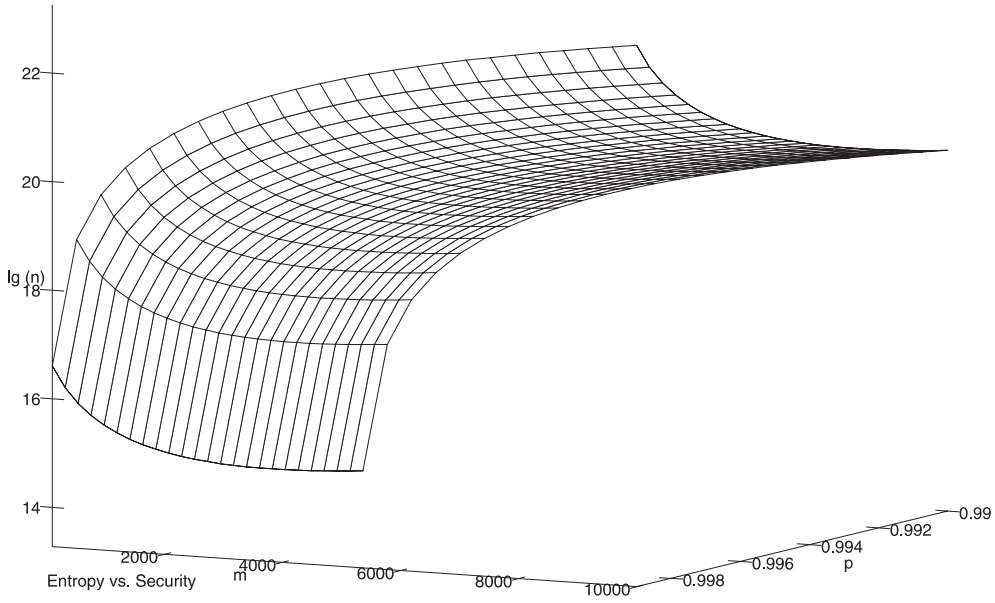

Fig. 2. Relationship between entropy of authenticators and security (Table 1 explains notation used).

usage of a typical terminal is known, m is a measure of the length of time before a fake terminal is detected and disabled. We can view p as the desired security guarantee, presumably determined by the risk we are willing to take. Fig. 2 shows the relationship between these parameters for the range $p = 0.99 \ldots 0.999$ and $m = 100 \ldots 10000$. For this range, the required entropy in n ranges from 13.3 bits (for $m = 100, 0.99$) to 23.25 bits (for $m = 10000, p = 0.999$). An entropy of 24 bits can be obtained by using an authentication vector with three components, each of which has 256 different possibilities.

Notice that this is a pessimistic analysis: we assumed that users in *none* of the m transactions bothered to inform the authorities about the fake terminal even though they would have been able to clearly detect the fact. [5]

4. Related work

Techniques in visual cryptography [14,16] are used to address the problem of a communication between a human user and a trusted digital entity (such as a smartcard or a central server) via an untrusted terminal under the control of an attacker. The techniques require the user to carry a secret key shared between the user and the digital entity in the form of a transparency. When the digital entity wants to send a message to the user, it ''encrypts'' the message using the shared key and gives the resulting random-looking image to the terminal to display to the user. The user can see the original message by placing his transparency over the displayed image. These techniques solve a more general problem than ours: they allow the digital entity to send *arbitrary* messages to the user, whereas we are interested in sending only a single bit of information. The generality comes at the cost of ease of use.

Haller et al. [8] and Matsumoto et al. [11,12]. addressed the problem of authenticating human users at untrusted terminals to a digital system. In this paper, we considered the opposite problem.

Our solution in the first scenario is essentially an application of well-known solutions for three-party authentication [5,10,15]. Authentication protocols for mobile users [13] are another example of a three-party authentication protocols. In Ref. [2] Abadi et al. considered the problem of delegating a user's privileges to a public terminal. They too described different protocols for different scenarios, starting from a user equipped with an ''ultimate smart-card'' that has its own input and output devices. Their solution for this case is similar to our solution for the first scenario. However, in the remaining scenarios where the user is equipped with a less capable device, their approach differs from ours. In their solutions, the user does not find out if a terminal is in fact a legal terminal. Instead, the user relies on a remote server to verify the terminal before approving the delegation of rights. This approach is sufficient where the issue is delegation of a set of pre-defined user rights to a terminal: a remote server can verify the delegation of these pre-defined rights. In contrast, our protocols focus on allowing the user to learn whether a terminal is a legal terminal. The server limits itself to authenticating the terminal *and* communicating the result to the user. It does not concern itself with how the result is going to be used; the user himself makes that decision. Consequently, we do not have to pre-define the contexts where the protocols are going to be employed in.

Customization of displayed security-critical data is a well-known technique. In our solutions, we combine customization with authentication to communicate to the human user whether a terminal is a legal terminal. Pfitzmann et al. [17] outline a similar approach. In their scenario, the user has two devices: a small security module (similar to D in our case) which does not have its own display, but which the user always carries with himself, and a larger, less trusted, device (similar to T in our case) with its own display. Before the first ever use of D, the user personalizes D by assigning it a permanent device identification number (DIN). An attacker may succeed in replacing T with a fake one which can steal the user's PIN. To protect against this, the user inserts D into T. T and D mutually authenticate each other. If mutual authentication succeeds, D reveals

[5] This is in contrast to today's ATMs where the user cannot tell the difference between a fake terminal and a legal terminal that is out-of-order.

the DIN to T which displays it to the user. The user enters the PIN only if he sees the correct DIN displayed to him.

Our solution can be viewed as a generalization of this approach for the following reasons. First, the authentication vector can be changed as often as the user wants. Second, we allow many different types of authenticators. A user's authentication vector may contain only a subset of these. A legal terminal could substitute random choices for the *other* types of authenticators (e.g., if a user's authentication factor does not include background color, a legal terminal can choose a random background colour). Also, as mentioned previously, a user may challenge T to show only a subset of the components of his authentication vector. Each of these generalizations help make our technique resistant against an attacker who ''peeps over the shoulders'' of users at legal terminals. Third, we allow the possibility of a central server being used to authenticate terminals. This is useful in the case where a central authority can remotely detect compromised legal terminals but cannot physically prevent unsuspecting users from continuing to use the terminal.

5. Conclusions

We addressed the problem of how mobile users can authenticate public terminals before using them for confidential transactions. We described different solutions for different scenarios based on the level of computational resources available to the user. We assumed that legal terminals are tamper-resistant. A useful extension is to address the case where an attacker *can* succeed in subverting legal terminals. In our current work [6], we are developing intrusion detection techniques that can be used with the techniques described in this paper.

We mention various types of authenticators which may be suitable for our purposes. It will be quite useful to carry out experiments with real users to identify which types are suitable, and how large a space can they be drawn from.

Acknowledgements

We thank Mehdi Nassehi, Luke O'Connor, Harry Rudin and Peter Klett for reading previous versions of this paper and giving us valuable feedback. We thank Luke for interesting discussions on related topics.

References

[1] N. Asokan et al., Deliverable D02: Preliminary report on basic services, architecture and design, Technical Report, SEMPER Consortium, 1996, SEMPER Project deliverable submitted to the European Commission (available from the authors); see url: http://www.semper.org for related information.

[2] M. Abadi, M. Burrows, C. Kaufman, B. Lampson, Authentication and delegation with smart-cards, Science of Computer Programming 21 (2) (October 1993) 91–113.

[3] oss Anderson, Why cryptosystems fail, in: V. Ashby (Ed.), 1st ACM Conf. on Computer and Communications Security, Fairfax, Virginia, November 1993, pp. 215–227.

[4] S. Brands, D. Chaum, Distance-bounding protocols, in: I.B. Damgard (Ed.), Advances in Cryptology – EUROCRYPT'93, Lecture Notes in Computer Science, vol.765, Springer, Berlin, May 1993, pp. 344–359.

[5] R. Bird, I. Gopal, A. Herzberg, P. Janson, S. Kutten, R. Molva, M. Yung, Systematic design of a family of attack-resistant authentication protocols, IEEE Journal on Selected Areas in Communications 11 (5) (June 1993) 679–693.

[6] H. Debar, N. Asokan, M. Steiner, Remote code execution on semi-trusted environment, 1998, in preparation.

[7] A.O. Freier, P. Kariton, P.C. Kocher, The SSL Protocol: Version 3.0, Technical Report, Internet Draft, 1996, Will be eventually replaced by TLS.

[8] N. Haller, The S/Key one-time password system, in: D. Nesset (General Chair), R. Shirey (Program Chair) (Eds.), Symposium on Network and Distributed Systems Security, San Diego, CA, February 1994, Internet Society.

[9] M.G. Kuhn, R.J. Anderson, Soft tempest: hidden data transmission using electromagnetic emanations, Workshop on Information Hiding, Portland, OR, April 1998.

[10] J.T. Kohl, B.C. Neuman, The Kerberos Network Authentication Service (V5), Internet Request for Comment RFC 1510, 1993.

[11] T. Matsumoto, Human-computer cryptography: an attempt, in: C. Neuman (Ed.), 3rd ACM Conference on Computer and Communications Security, New Delhi, India, March 1996, pp. 68–75.

[12] T. Matsumoto, H. Imai, Human identification through insecure channel, in: Advances in Cryptology – EUROCRYPT'91, Lecture Notes in Computer Science, vol. 547, Springer, Berlin, April 1991, pp. 409–421.

[13] R. Molva, D. Samfat, G. Tsudik, Authentication of mobile users, IEEE Network 8 (2) (March/April 1994) 26–35.

[14] M. Naor, B. Pinkas, Visual authentication and identification, in: B.S. Kaliski Jr. (Ed.), Advances in Cryptology – CRYPTO'97, Lecture Notes in Computer Science, vol. 1294, Springer, Berlin, August 1997, pp. 323–336.

[15] R.M. Needham, M.D. Schroeder, Using encryption for authentication in large networks of computers, Communications of the ACM 21 (12) (December 1978) 993–999, also Xerox Research Report, CSL-78-4, PARC.
[16] M. Naor, A. Shamir, Visual cryptography, in: I.B. Damgard (Ed.), Advances in Cryptology – EUROCRYPT'94, Lecture Notes in Computer Science, pages 1–12. Springer-Verlag, Berlin Germany, 1994.
[17] A. Pftizmann, B. Pfitzmann, M. Schunter, M. Waidner, Trusting mobile user devices and security modules, IEEE Computer 30 (2) (February 1997) 61–68.
[18] J.D. Tygar, A. Whitten. WWW electronic commerce and Java Trojan horses, in: 2nd USENIX Workshop on Electronic Commerce, Oakland, CA, November 1996, pp. 243–250.

N. Asokan is a senior R&D engineer with the Nokia Research Center in Helsinki, Finland. Prior to joining Nokia, he was a research scientist at the IBM Zurich Research Laboratory from 1995 to 1998 and a software systems specialist at the University of Waterloo from 1990 to 1995. His research interests include applied cryptography, security in distributed systems, mobile computing, and electronic commerce. Dr. Asokan received his Ph.D. from the University of Waterloo, and MS from Syracuse University, both in computer science and his BTech in computer science and engineering from the Indian Institute of Technology at Kharagpur. He is a member of the ACM, IACR, and IEEE.

Michael Steiner is a research scientist in the network security research group at the IBM Zurich Research Laboratory, where he works on security in network management and electronic commerce. His interests include secure and reliable systems as well as cryptography. He received a Diplom in computer science from the Swiss Federal Institute of Technology (ETH). He is a member of the ACM.

Hervé Debar is a research scientist in the global security analysis laboratory at the IBM Zurich Research Laboratory, where he works on system and network security (in particular intrusion detection) as well as system management. His interests include secure systems and artificial intelligence. Dr. Debar holds a Ph.D. from the University of Paris and is a telecommunications engineer from the Institut National des Télécommunications in Evry (France).

Michael Waidner is the manager of the network security research group at the IBM Zurich Research Laboratory. His research interests include cryptography, security, and all aspects of dependability in distributed systems. He has coauthored numerous publications in these fields. Dr. Waidner received his diploma and doctorate in computer science from the University of Karlsruhe, Germany. He is a member of the ACM, GU, IACR, and SIAM.