Authentication Method with Impersonal Token Cards†

Refik Molva
EUREKOM Institute
Sophia Antipolis, 06560 Valbonne, France
molva@eurecom.fr

Gene Tsudik
IBM Zürich Research Laboratory
CH-8803 Rüschlikon, Switzerland
gts@zurich.ibm.com

Abstract

Traditional methods of user authentication in distributed systems suffer from an important weakness which is due to the low degree of randomness in secrets that human beings can use for identification. Even though weak secrets (passwords and PINs) are typically not exposed in the clear over the communication lines, they can be discovered with off-line brute force attacks based on exhaustive trials. Since such secrets are chosen from a relatively small key space, a determined adversary can try all possible values until a match is found between the trial value and the message recorded from a genuine authentication session. Authentication devices like smartcards and token cards offer an attractive solution by providing a user with a cryptographically strong key for authentication. In contrast to passwords and PINs, the device's key can be chosen from a much larger key space thus making a brute force attack computationally infeasible or, at least, difficult.

In this paper we present a novel authentication method whereby the authentication device (a token card) is used solely to provide a secure channel between a human user and an authentication server (AS). Since the communication channel is secured by the card, the user can still utilize weak secrets for authentication purposes, but, without any risk of exposure. Furthermore, the card's and the user's secrets are mutually independent, i.e., the card is not associated with any particular user. Since the card is impersonal, *it can be freely shared by several users. This eliminates the high cost of administration which is typical of existing designs requiring fixed user-device relationship. Moreover, our method does not require any coupling between the token card and the workstation (e.g., a galvanic connection) which would be difficult to implement on a global scale and retrofit onto existing equipment.*

† In Proceedings of 1993 IEEE Symposium on Research in Security and Privacy.

1 Introduction

Users of distributed systems need to authenticate themselves in order to gain access to system resources and to delegate their rights to programs that will be accessing system resources on their behalf. Traditional user authentication mechanisms are based on the user's knowledge of a password or a Personal Identification Number (PIN). An important vulnerability common to most such methods is due to the lack of randomness in weak secrets such as passwords and PINs. Not only the methods whereby the password or PIN value is transmitted in plaintext are vulnerable but also the ones limiting the exposure of the password or PIN to their utilization as an encryption key are subject to *verifiable plaintext* attacks as described in [5]. In [13] an authentication scheme based on passwords and public-key cryptography is presented. Even though this scheme is more secure than traditional schemes based on passwords or PINs, it is not resistant to verifiable plaintext attacks nor to the disclosure of the password or PIN value by trojan horse programs residing in the public or shared login terminal. Many contemporary authentication designs involve a trusted hardware device called a *smartcard* or a *token* that addresses this weakness by giving a user the ability to authenticate using a strong cryptographic key.

Many existing designs based on such devices use a delegation technique (e.g., [1]) whereby the device acts on behalf of the user by deploying its strong cryptographic capability. However, before the device can act on user's behalf, the user has to authenticate himself to the device. This requires some interaction between the user and his device which may present a vulnerability if the user is still authenticated on the basis of a weak secret.[1]. An intruder in possession of a user's authentication device can try to exhaust all possible choices and break the weak secret. However, the effectiveness of such an attack is limited by the speed

[1] Of course, authentication devices employing biometric authentication are not susceptible.

of manual entry at the device's keypad (or workstation's keyboard if there is an interface between the device and the workstation). Moreover, an authentication device can be programmed to refuse any usage after a reasonably small number of (e.g. 3-4) failed authentication attempts.

An inherent disadvantage of most existing authentication schemes based on such devices is the administrative burden incurred by the fixed relationship between the authentication device and the user. Since the device is expected to act on behalf of the user with respect to external parties, the relationship between the device and the user must be corroborated and protected by a trusted authority so that all transactions performed by the device can be accounted for and traced to the associated human user. In order to maintain this relationship, the administration must maintain a mapping between users and devices. Moreover, it must assure a means for secure distribution, update and revocation of the devices or of the personal secret keys to be installed in the devices.

In this paper we present a novel authentication method whereby the fixed relationship between the user and the device is avoided. The authentication device is used solely to provide a secure channel between a human user and an authentication server (AS). Since the communication channel is secured by the device, the user can still utilize weak secrets for authentication purposes, but, without any risk of exposure. Furthermore, the device's and the user's secrets are mutually independent, i.e., the device is not associated with any particular user. This "impersonal" quality allows a single device to be shared by several users. This new concept eliminates the high cost of administration which is typical of existing designs that employ fixed user-device relationship. Moreover, because the device involved is of the so-called *token card* variety, there is no interface (coupling) between the device and the workstation, e.g., a galvanic connection. Therefore, realization of this method on existing systems is logistically simpler and much less costly than in the case of existing smartcard-based solutions which require equipping all workstations with card readers or similar hardware gadgets.

2 Target Environment

Throughout the remainder of the paper, a particular operating environment is assumed. We consider a very large distributed system environment, e.g., a global Internetwork, with a dynamic user and workstation population. The system offers a number of services accessible via public workstations. Workstations are geographically dispersed. They are not only large in number, but also *anonymous* in a sense that a particular workstation may not necessarily be registered in any way. Furthermore, the workstations themselves, the software they run, and the underlying network are all completely untrusted. For example, consider public terminals currently offered to attendees of some conferences or to guests in some hotels. Prospective users are ill-advised to make any assumptions about security in such an environment.

There are a number of generally trusted, highly-secure ASs situated throughout the internetwork. An AS has two main duties: 1) to maintain a database of user records, and, 2) to act as a *certification authority* (TGS, *a la* Kerberos [9]) for its constituent users

Every workstation is assumed to know of at least one (perhaps, nearest) AS and every human user is registered with at least one AS. Finally, users are considered to be highly mobile; a user may request system access from any public workstation.

3 Traditional User Authentication

In a distributed systems environment, user authentication is typically attained by exchanging a number of messages between the user (or a workstation acting on the user's behalf) and the authentication authority. In the course of the authentication process the user has to prove his identity to the authenticator by demonstrating the knowledge of a secret which is shared with the authenticator.

Many user authentication protocols are susceptible to masquerading attacks whereby an intruder spoofs, intercepts and replays authentication messages. In older protocols where the user's secret is sent in cleartext simple spoofing and replay is sufficient to break the protocol. More recent protocols (e..g, Kerberos which is based on Needham-Schroeder protocols [6, 9]) use the user's secret as an encryption key or a seed from which an encryption key is derived. This measure is not very effective because, as described in [5], such an encryption key is still weak and can be easily broken by wiretappers. The weakness is due to the lack of randomness in the way human users choose their secrets and to the humans' difficulty of remembering truly random numbers. In other words, the user's secret is chosen out of a key space which is relatively small in comparison with the minimum key space required by a good cryptographic algorithm. Usually, the secret is a password chosen from a dictionary the size of which (at most 10^5) is several orders of magnitude smaller than, for example, the one required by the Data Encryption Standard (DES[3]) (2^{56}).[2] Cryptographic keys derived from such weak secrets can be easily broken by brute force attacks with an exhaustive search in the relatively small key space from which the secret is chosen.

A practical mechanism for recovering strong cryptographic keys using weak secrets without exposure is provided by additional authentication devices like smartcards and token cards [1, 4].

4 Device-based Authentication

Some current approaches address the exposure described above by using an authentication device with limited processing capability that contains a cryptographically strong key. The purpose of the key is to aid user authentication in a *hostile* environment. Unlike the weak keys (passwords and PINs) used by human beings, the device's key is randomly chosen out of the total key space of the underlying cryptographic algorithm. The probability of success with a brute force attack based on exhaustive search in the key space is therefore negligible. The user must activate the device operation by authenticating himself using a weak initial secret but this interaction takes place directly between the user and the device (via device's keypad or a protected card reader device) without any involvement of untrusted media. Thereafter, all data exchanged over the untrusted network is sent under the protection afforded by encryption using the device's strong secret.

Existing device-based authentication methods differ widely in many respects. We are particularly interested in the differences as they translate into the requirements for the underlying device (be it a "true" smartcard loaded with features or a "dumb" token card with a primitive interface.) The following device features influence both the cost and the security of the associated authentication methods:

- Device-Workstation Coupling
  One way for the device and the workstation to communicate is through an electronic interfaces, e.g., a card reader. This is an expensive solution as it requires card readers on all workstations. Alternatively, a user can act as an *intermediary* between the device and the workstation by copying and entering the necessary information by hand.

- Interaction Complexity
  It is clearly desirable to keep the volume of the information that user must exchange with the device down to a minimum. Electronic coupling eases the problem since the interface between the smartcard and the workstation allows for fast information transfer without user's involvement. For token cards (without electronic coupling), the user must act as an intermediary. In order to ensure ease-of-use, functional complexity (of the device) is can be traded off in return for minimal interaction complexity.

- Keypad
  A keypad may be needed to enter into the device the user's secret like a password or a PIN. If a device is not equipped with an electronic interface, other information may need to be entered via the device's keypad.

- Clock
  A clock may be required for generating timeliness indicators and, possibly, nonces [6].

- Display
  A display is imperative when there is no coupling between a device and a workstation. With electronic coupling, however, a workstation's display may be utilized as described in [1].

- Non-volatile Storage (ROM)
  Stable, non-volatile read-only storage is needed to store the device's secrets, e.g., a key or a nonce generator seed. It may also be needed to store public key(s) of the Certification Authority (CA) or the Authentication Server (AS). Some designs may also require a non-volatile RAM to store secrets or sequence numbers generated at run-time. The drawback of maintaining a non-volatile RAM is the amount of power needed to refresh the memory that is relatively high in comparison with the power required by a clock.

- Encryption/Decryption Ability
  The complexity of the encryption algorithm influences the cost and the performance of the device. One possibility is to restrict the device's computation power to perform only secret one-way function (i.e., encryption but no decryption). Some recent smartcard schemes employ asymmetrical (public key) cryptography (e.g., [1, 11]). This has two main drawbacks:

[2] The size of the PIN space is usually at most 10^6.

- Public key encryption remains quite expensive in terms of both implementation and performance.
- Most public key cryptosystems are covered by patents, e.g., Diffie-Hellman[2] and RSA[8].

Traditional smartcard techniques employing more conventional, symmetrical encryption do not suffer from these problems but, instead, suffer from heavy administrative burden owing to the need to maintain a per card record at the AS (in addition to user records containing passwords).

Finally, as alluded to above, all current device-based methods are based on fixed user-device relationship. This relationship originates from the basic concept of delegation through which the user delegates the device to perform an authentication procedure with the AS on his behalf. The major drawbacks of this relationship are the potential for masquerading by breaking into a stolen device and the administrative burden of maintaining this relationship that consists of safe distribution, updating and revocation of the devices like in the case of PIN-based ATM cards. Besides, practical solutions based on token cards like the *SecureID* card [12] suffer from the exposure of plaintext password or PIN to trojan horse programs residing in the workstation.

5 Authentication Method with *Impersonal* Token Cards

We begin with the discussion of the salient characteristics of the token card as required by the new authentication method.

5.1 Card Features

The token card has the following features:

- *No card-user relationship*:
  The token card is completely decoupled from the user. It has no PIN or password checking capabilities and acts only as a means for providing a secure channel between the user and the AS.[3]
- *No keypad*:
  The token card has no keypad but only a single button. This button controls the state of the card (ON/OFF).
- *No coupling to the workstation*:
  There is no interface (electronic or otherwise) between the token card and the workstation.
- *Display*:
  There is an LCD display on the card.
- *Internal clock*:
  The card has a built-in clock. The clock has no dedicated display. The time is displayed (i.e, the display is active) only when the card is *ON*. The clock does not need to have high resolution; second precision is sufficient for reasons explained below.
- *Cryptographic ability*:
  The token card implements a strong secret one-way function, such as DES encryption (but no decryption) [3] or MD5 with secret suffix [7, 10].
- *Card's secret key*:
  Every token card, C, possesses a secret, K_c which is computed as[4]:
  $$K_c = E(K_{as}, SN_c)$$
  where SN_c is the unique *Serial Number* of C and K_{as} is the *Card Key Generation Key*, a secret key known only to the AS(s). At the time of manufacture, each token card is assigned a unique SN_c and a corresponding K_c. While K_c is a secret value, SN_c is not. For example, SN_c may be etched on the back of every token card, not unlike serial numbers on other electronic merchandise (e.g., workstations, VCRs). Even the means for generation of SN_c's is not necessarily kept secret; it may simply be a monotonically increasing 32-bit (ten-digit) number.

  This method for generating K_c is an application of a well-known approach usually referred to as **name-based** (or **id-based**) scheme [4].

The resulting token card is depicted in Figure 1.

5.2 User Requirements

Every legitimate user is identified by a combination of: i) a unique user (or login) name, and ii) a password

[3] We envisage that such a token card can be purchased over-the-counter in a retail shop. No buyer registration would be necessary and users would be free to resell, exchange, discard or lend their cards to anyone.

[4] We use the notation $E(K, A)$ to denote encryption of a value A under a key K.

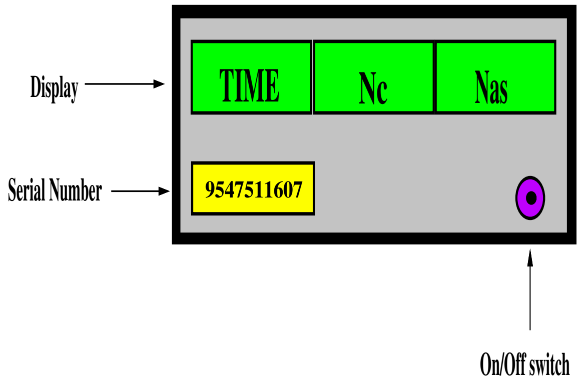

Figure 1: An *impersonal* token card

or a PIN. A password may be an alphanumeric string of, say, eight characters, while a PIN is generally a numeric string (i.e., a decimal number) of at most five-six digits in length. For clarity's sake, the term *PIN* is used hereafter to mean both password and PIN in their traditional sense.

5.3 AS Requirements

Every AS is responsible for keeping the records of its constituent users. A user's record includes, among other things, the name and the PIN of the user. For further refinement, the AS may know only a one-way function of the PIN similar to the way most modern operating systems store only a one-way function of the users' passwords. However, for simplicity's sake, we will assume in the remainder of this paper that the PIN itself is stored by the AS. The only information the AS has to know about *all* public token cards is the Card Key Generation Key, K_{as}. The AS does not keep track of the identities or secrets of individual token cards.

5.4 Protocol Steps

The goals of the protocol are to achieve:

- mutual authentication of the AS and the user (i.e., workstation authentication is not taken into account but our protocol can be easily extended to provide it),
- delegation of the user's identity to the workstation for a limited duration.

The protocol consists of the following steps:[5]

1. The user begins by turning on the token card using the sequence button. The token card immediately computes and displays two values:
   (a) $TIME$ - Current date/time. Hereafter, the term $TIME$ is used to denote both time of day and date (e.g., *time: 12:35.02 date: 02-29-92*).
   (b) $E[K_c, TIME]$ - Encryption of current time under K_c. Throughout the rest of this section, this value is referred to as N_c. Here we utilize the property that, although time is predictable or easily-guessed, its encryption under a strong secret key is random and unpredictable. Furthermore, as a clock does not run backwards, N_c is guaranteed to be unique. Hence, N_c can be considered a "good" nonce.
2. The user supplies the following values to the workstation:
   (a) U - User name.
   (b) SN_c - Serial number of the token card.
   (c) $TIME$ - Current date/time taken from the token card's display. (Actually, the entire date/time string need not be entered into the workstation. See Section 5.5.)
   (d) K_u - User's authenticator computed as: $K_u = (N_c + PIN_u)$, where PIN_u is the user's PIN. Our principal assumption in this step is that it is fairly easy for the human user to compute the sum of N_c and PIN_u. Moreover, it is not the *arithmetic* sum that the user has to compute. For each digit of N_c it suffices to compute the modulo 10 addition of that particular digit and the corresponding digit of the PIN.[6] Addition is no the only example of an operation that can be used to compute K_u; digit-by-digit modulo 10 subtraction can be used as can other methods (see Section 5.5 below).

   K_u is a one-time credential that delegates the identity of the user to the workstation without disclosing the PIN to the workstation. The workstation can use K_u to act on behalf of the user beyond the user's authentication session, for instance, as a key encryption key to get pair-wise keys from

[5] The protocol is illustrated pictorially in Figure 2.

[6] An assumption is made here that an average user is able to compute the modulo 10 addition of two single-digit numbers in his head.

the AS to communicate with other systems. The validity of K_u is limited in time, because it is computed as a secret function of the current time value. The lifetime of K_u can be defined as $TIME + lifetime$ or included as an explicit value in the expression of K_u.

3. Now the workstation sends to the AS:
   (a) U, SN_c, $TIME$ - All unmodified from the previous step.
   (b) $E[K_u, TIME]$ - Encryption of $TIME$ under K_u. Here, it is assumed that K_u forms a valid encryption key. By sending this value the workstation proves to the AS that it was granted a valid one-time credential by the user, i.e. K_u, without disclosing the value of the latter.

4. AS uses SN_c and K_{as} to compute K_c. Next, it computes $E[K_c, TIME]$ in order to obtain a candidate N_c value, $\hat{N}_c$. Using U, AS looks up PIN_u and obtains a candidate K_u value, $\hat{K}_u = [N_c + PIN_u]$. Then, AS recomputes $E[\hat{K}_u, TIME]$ and compares it with $E[K_u, TIME]$ supplied by the workstation in the previous step. If there is a match, AS replies with:
   (a) $E[K_u, f(TIME)]$
   Encryption of $f(TIME)$ under K_u where the function f is a simple arithmetic function, e.g., one's complement. This value is intended to authenticate AS to the workstation and, to the user if (s)he trusts the workstation.
   (b) $N_{as} = E[K_c, f(TIME)]$
   Encryption of $f(TIME)$ under K_c. This value is optional; its purpose is to authenticate AS to the user if the user does not trust the workstation (see step 6 below).

   In this step, AS is simultaneously assured of the freshness and the authenticity of the message it received. The authentication of both the token card and the user is attained by recomputing $E[K_u, TIME]$. This is because K_u is uniquely dependent on SN_c, N_c and PIN_u. Freshness is confirmed as a part of the same sequence of checks since N_c depends on a particular $TIME$ value. Furthermore, the cleartext $TIME$ field can be validated before any other checks are made. (Recall that we assume loose time synchronization between token cards and ASs. i.e., there is a maximum time skew.)

5. The workstation verifies $E[K_u, f(TIME)]$ sent by the AS. This step assures the workstation that someone (presumably, the AS) possesses K_u. Finally, the workstation displays N_{as} on the screen.

6. In order to authenticate the AS, the user pushes the token card's sequence button and reads the authentication value expected from the AS, i.e., $N_{as} = E[K_c, f(TIME)]$, on the token card display and performs a visual comparison of this value with the candidate value sent by the AS which is displayed on the workstation's screen.

   If the two values match, the authentication process is completed. The goal of the last comparison is to assure the user that he/she has been communicating with the real AS (since no one but the AS and the token card at hand can compute $E[K_c, f(TIME)]$). We note that it may be considered laborious for the user to compare two random-looking 64-bit (8 byte) values in this step. In that case, instead of comparing $E[K_c, f(TIME)]$, it will suffice to compare shorter values, e.g., 6-digit truncation of $E[K_c, f(TIME)]$ or even a digest of it (using MD5[7], for example).

   It is important to clarify the meaning of the last step. Most (if not all) existing authentication protocols based on token cards only provide for the authentication of **user-to-AS** but not **AS-to-user**. The protocol above provides for **bidirectional** authentication. However, if AS-to-user authentication is not desired, the user is free to forgo the last step entirely.

5.5 Usability Concerns

The main usability concern in the above scheme has to do with the interaction complexity of the authentication protocol, i.e., the number operations imposed on the human user. These operations include:

1. Entering SN_c and $TIME$ into the workstation.

2. Composing K_u from PIN_u and N_c and entering K_u into the workstation.

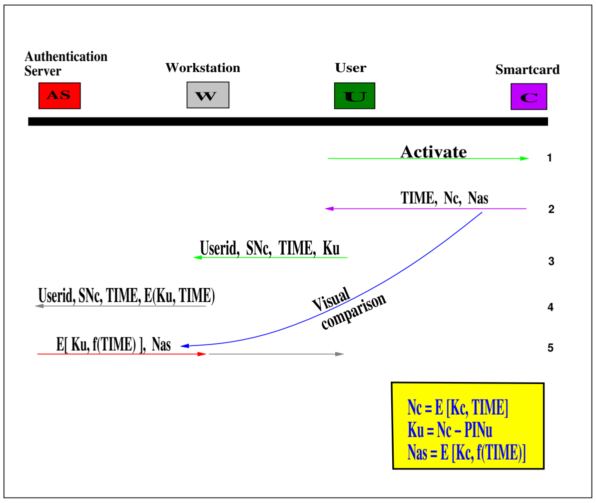

Figure 2: Authentication Protocol

3. (Optional) visual comparison of $E[K_c, f(TIME)]$ displayed by the workstation and its counterpart displayed by the token card.

Of these three operations, only the first two are labor-intensive; the third is strictly optional. In the first operation, SN_c is read directly from the token card as a decimal number of, say, 10 digits. Current date/time can also be entered directly as a decimal number. Alternatively, the workstation can be programmed to display its own time (which is assumed to be fairly close to the time kept by the token card) and the user can use the four $ARROW$ keys to modify the displayed value to match the one shown by the token card.

The biggest usability concern is the composition of K_u since it requires the user to somehow combine PIN_u and N_c to obtain K_u. This difficulty is a direct consequence of our *minimalist* design. If we augment the token card with a digit-only keypad, the user can simply enter his PIN via the keypad and leave it to the token card to compute K_u. However, a keypad would not only increase the production costs but would also result in the token card having larger surface area. Moreover, volatile storage would be necessary in order to store the user's PIN. Therefore, rather than add new features to the token card, we intend to make the composition of K_u easier for the user.

Assuming a 6-digit (decimal) PIN, the user can obtain K_u in two alternative ways (many other variations are possible as well):

1. The user adds digit-by-digit his PIN_u with the first 6 digits of N_c. (The first 6 digits of N_c can be displayed highlighted in order to ease visual operations.) The user then enters K_u into the workstation as the 6 decimal digits resulting from the addition followed by the 14 remaining digits of N_c. (See Figure 3 for an illustration.) This method requires the ability to perform modulo 10 addition of 6 decimal digits. Also, we note that, while in the process of adding the two numbers, the user is expected to keep in his head a "running index" of the current PIN digit involved in the addition. This may prove to be more difficult that the actual addition.

   There is, of course, the danger of a user performing the addition on a piece of paper (or using a workstation-provided calculator software). One simple solution is to have each workstation display on its screen (or have attached to it physically) a simple 10-by-10 table of single-digit decimal numbers and their modulo 10 sum (e.g., row 9, column 6 will display 5).

2. Alternatively, the display area of the token card can be modified so that each digit of N_c is labeled with a fixed index carved on the token card surface or printed on the LCD. The first 10 digits of N_c are thus numbered from 0 to 9. Using each digit of his PIN as an index the user reads the N_c digit displayed below the label corresponding to the value of the PIN digit. Each PIN digit thus points to an N_c digit that is entered to the workstation to form the first 6 digits of K_u. The main advantage of this approach is that no arithmetic is required from the user. (See figure 3 for an example.)

Another issue of concern is the length of K_u. In the protocol described in Section 5.4, K_u plays two roles: i) authenticator of the user to the AS (flow 3 in Figure 2), and ii) encryption key in the AS's reply to the user (flow 5 in Figure 2). Furthermore, subsequent to the authentication protocol, it is possible to use K_u as a one-time (or session) key. However, if K_u was only used as a one-time authenticator (i.e., its use would be limited to the flows of the authentication protocol), it would be possible make it the same length as the user's PIN (6 decimal digits).

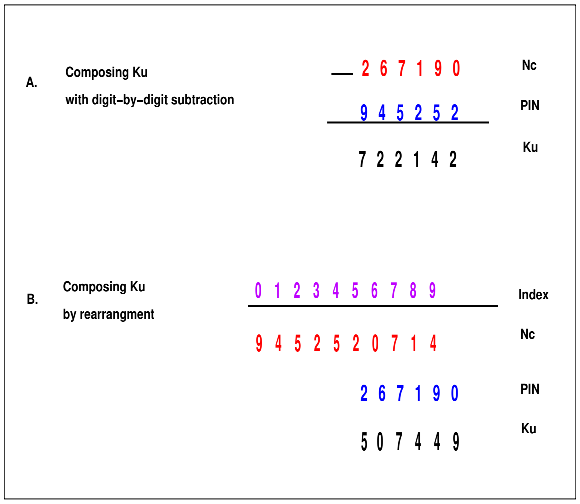

Figure 3: Two examples of composing K_u

5.6 Informal Analysis

In this section we analyze the protocol presented above. The following assumptions are made:

- Every token card's secret, K_c, is a strong key. The derivation of K_c from K_{as} can be designed to be as strong as required by a particular application because there is no limit on the complexity of this operation that is performed only by the AS as opposed to other operations that are also performed by the token card.

- The token card is trusted to faithfully display appropriate values.

- It is difficult to subvert the token card's hardware and obtain the secret (K_c) or otherwise manipulate the token card (e.g., set back the clock).

- The token card clock is assumed to be monotonically increasing.

- The token card can be stolen.

- Any public workstation can be taken over by a hostile party. All communication involving a workstation is subject to interception and divulgement and the workstation may contain trojan horse programs that disclose all the information entered by the user into the workstation or sent by the AS.

- A *bona fide*, registered user may turn malicious – his purpose may be to discover other users' secrets (for instance, by letting them use his token card).

The main purpose of this analysis is to show that the protocol achieves its goals, i.e.:

● $\longrightarrow$ *AS believes that it is talking to a particular user, U, at time T through a token card C.*

More formally, this can be stated as: **AS believes that U recently generated K_u using C.** Applying our assumption that only U and the AS know PIN_u and no two token cards share the same key, only C can generate $N_c = E(K_c, T)$ and, hence, only U can compute $K_u = N_c + PIN_u$.

● $\longrightarrow$ *U believes that he/she is talking to AS at time T.*

This can be re-stated as: U **believes that** AS **recently generated** $E[K_c, f(T)]$**.** The issue at hand is wether the same value, $E[K_c, f(T)]$, can be somehow obtained by the adversary, presumably from the previous flows of the protocol. This requires us to look more closely at the function f.

- First, f must be a one-to-one function as the protocol relies on the condition that a unique value of time T correspond to an equally unique value of $f(T)$.

- Second, f must have a property that, given a *plausible* time value T, $f(T)$ can **never** amount to a plausible or valid time value.[7]

One example of a function satisfying both of these conditions is the one's complement function. (Recall that our time value T includes the date; thus, one's complement of T can never amount to a valid date/time value).

We have now established that $E[K_c, f(T)]$ could not have been generated by either the card C or the AS in the past. Since only the AS and C know K_c, the user U is assured that AS's communication (i.e., $E[K_c, f(T)]$) is both authentic and timely.

There is, however, a small caveat. Since the token card has no knowledge of its immediate user, its challenge value, $E[K_c, f(T)]$, cannot be dependent on the user; neither can the AS's response to that challenge. Therefore, if two well-meaning (and even mutually trusting) users decide to "save time" by activating

[7] This is necessary since, otherwise, the adversary can use the pre-recorded $E[K_c, f(T)]$ value at some later time when $f(T)$ becomes "timely" in order to break the PIN; of course, the adversary has to be lucky in sense that the user will have to be logging in at that particular time, i.e., $f(T)$.

the token card only once for both logins, the second part of the message returned by the AS ($E[K_c, f(T)]$) will be identical in both cases. This implies that a malicious workstation can mislead one of the two users into believing that he/she is talking to the AS.

Another important protocol property is that:

● $\longrightarrow$ *PIN_u cannot be discovered by an intruder (resistance to verifiable plaintext attacks)* .

The user enters PIN_u *indirectly* in Step 3 of the protocol. Since K_u is the only value dependent on the PIN in the entire protocol, the only venue for obtaining the PIN is from K_u. This is reasonable, since one of our assumptions is that a workstation may turn malicious and try to misuse K_u. However, in order to extract the PIN from K_u, the knowledge of N_c is required (recall that $K_u = N_c + PIN_u$). But, N_c is known only to: i) the AS, ii) the token card, and iii) the user.

One of our principal assumptions is that the token card clock never runs backward. It guarantees that N_c are never "recycled", i.e., every N_c is unique, and unpredictable. Therefore, although a workstation may accumulate a number of K_u's for the same user or many different users, it is not able to extract a single PIN, since in all K_u-s, a PIN is masked by a nonce.

We now consider the case of a shared token card. Suppose that two users, Alice and Bob, share the same token card. The issue is whether or not it is possible for one of them (say, Bob) to discover the other's PIN. It is fair to assume that Bob can subvert a public workstation and discover Alice's K_u^{Alice}. Then, in order to extract the PIN, he will need to obtain the same N_c as was used for K_u^{Alice}. He cannot obtain it by manipulating the token card after the fact (since N_c is time-dependent). Hence, the only viable method of attack is to to look over the shoulder and record N_c the "hard way". However, we note that this is also an issue in current non-token card techniques.

Finally, this protocol offers protection against some denial-of-service attacks since an authentication request sent by the user is answered only if the AS is able to verify the authenticity, integrity and timeliness of the request. Thus, bogus or otherwise fraudulent requests are not serviced.

6 Summary

The mechanism described in this paper offers the following advantages over existing authentication designs based on special devices:

- **The token card is not personalized, i.e., it is not associated with a particular user.** This property implies several advantages:
  - **No administration cost:** the token card does not need to be registered under a user's name or sent to a particular user with a safe courier. Token cards can be freely purchased over the counter (with no special registration procedure) and subsequently shared, discarded or exchanged.
  - **Prevention of potential masquerading:** Since a token card, by itself, does not represent any user, its theft carries no danger. In other words, a stolen token card cannot be misused in any way to obtain the rights of any of its past or future users.
  - **No PIN storage on the token card:** the user's secret does not need to be stored on the token card. This eliminates the need for entering, updating and storing user specific secrets (passwords, PINs, biometric patterns, etc.) on the token card. This feature is conducive to a low-cost design.
- **The token card's secret key is not stored in the AS.** This offers the following advantage:
  - **Minimum key management requirement:** Since we use a well-known **name-based** scheme for generating token card keys [4], no per card records ever need to be maintained. The AS has to keep only one key to be able to retrieve all the token card keys. The management of the token card keys has therefore a minimal complexity. The key storage in the AS is independent of the existing token card population; addition, update, revocation of token cards and/or their keys have no effect on the AS.
- The token card protocols achieve the above mentioned goals with minimum requirements for token card and protocol features:
  - The token card design and protocols require no hardware modifications to existing (workstation) equipment. In other word, there is no electronic coupling between the token card and the workstation.
  - The design does not rely on public key cryptography or other sophisticated encryption algorithms that impose significant

execution overhead. Only a secret one-way function is required, e.g., DES encryption.

In this paper we discussed a number of issues in today's authentication methods based on special devices like smartcards and token cards. We introduced a new method which, in addition to being both simple and secure, lends itself to immediate deployment with very little cost. The token card design and protocols we envisage require no hardware modifications to existing terminal (workstation) equipment (i.e., no card readers or galvanic connections), incur no significant overhead due to time-consuming encryption (e.g., public key) and, at the same time avoid the administrative burden typical of other device-based authentication solutions.

Acknowledgements

The authors are grateful to Todd Arnold, Phil Janson, Els Van Herreweghen, Ralf Hauser and Liba Svobodova for their helpful comments and discussions.

References

[1] M. Abadi, M. Burrows, C. Kaufman, B. Lampson, *Authentication and Delegation with Smart-cards*, DEC SRC Technical Report #67, October 1990.

[2] W. Diffie and M. Hellman, *New Directions in Cryptography*, IEEE Transactions on Information Theory, November 1976.

[3] National Bureau of Standards, *Federal Information Processing Standards*, National Bureau of Standards, Publication 46, 1977.

[4] H. Konigs, *Cryptographic Identification Methods for Smart Cards in the Process of Standardization*, IEEE Communications Magazine, June 1991.

[5] T. Lomas, L. Gong, J. Saltzer, R. Needham, *Reducing Risks from Poorly Chosen Keys*, Proceedings of ACM Symposium on Operating System Principles, 1989.

[6] R. Needham and M. Schroeder, *Using Encryption for Authentication in Large Networks of Computers*, Communications of the ACM, December 1978.

[7] R. Rivest, *The MD5 Message Digest Algorithm*, Internet DRAFT, July 1991.

[8] R. Rivest, A. Shamir and L. Adleman, *A Method for Obtaining Digital Signatures and Public Key Cryptosystems*, Communications of the ACM, February 1978.

[9] J. Steiner, C. Neuman, J. Schiller, *Kerberos: An Authentication Service for Open Network Systems*, Proceedings of USENIX Winter Conference, February 1988.

[10] G. Tsudik, *Message Authentication with One-Way Hash Functions*, Proceedings of IEEE INFOCOM 1992. May 1992.

[11] T. Woo and S. Lam, *Authentication Protocols* Proceedings of ACM SIGCOMM 92, September 1992.

[12] Security Dynamics, *The ACE System Access Control Encryption* Product Information, 1992.

[13] J. Linn, *Practical Authentication for Distributed Computing* Proceedings of the IEEE Symposium on Security and Privacy, May 1990.