Secure and Inexpensive Authentication with *Minimalist* Smartcards

Refik Molva
EURECOM Sophia Antipolis
molva@eurecom.cica.fr

Gene Tsudik
IBM Zurich Research Laboratory
gts@zurich.ibm.com

April 16, 1992

Abstract

Traditional authentication of human users suffers from an important weakness owing to the low degree of randomness in secrets that a human being can use for identification. Even though weak secrets (e.g., 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, an intruder can try all possible values until a match is found between the trial value and the message recorded from a genuine authentication session. Smartcards offer an attractive solution by providing a user with a cryptographically strong key for authentication. In contrast to passwords and PINs, a smartcard'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 smartcard design and associated authentication protocols where the smartcard is used solely to provide a secure channel between a human user and an authentication server (AS). Since the communication channel is made secure by the smartcard, the human user can still utilize weak secrets for authentication purposes, but, without any risk of exposure. Furthermore, the smartcard's and the user's secrets are mutually independent, i.e., the smartcard is not associated with any particular user; its only responsibility is to assist the user in the authentication process. Since the smartcard is not personalized, it can be shared by several users. This new concept eliminates the high cost of administration which is typical of existing designs that employ fixed user-smartcard relationship. Moreover, our design does not require any interface between the smartcard and the workstation (e.g., a galvanic connection) which is difficult to implement on a global scale and retrofit onto existing equipment. We discuss two smartcard variants: one with a built-in clock and one without, and analyze the associated authentication protocols.

1 Introduction

Human 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 the 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 weakness common to most such protocols is due to the lack of randomness in weak secrets such as passwords and PIN's[1] Many contemporary authentication designs involve a trusted hardware component called a *smartcard* that addresses this weakness by giving a user the ability to authenticate using a strong cryptographic key.

Many existing smartcard designs use a delegation technique (e.g., [1]) whereby the smartcard acts on behalf of the user by deploying its strong cryptographic capability. However, before the smartcard

[1] As discussed in Section 2 below.

can act on user's behalf, the user has to authenticate himself to the smartcard. This requires some interaction between the user and his smartcard which may present a vulnerability if the user is still authenticated on the basis of a weak secret.[2]. An intruder in possession of a user's smartcard can try to exhaust all possible choices and break the weak secret. However, the effectiveness of such an attack is limited by the speed of manual entry at the smartcard's keypad (or workstation's keyboard if there is a smartcard-workstation interface). Moreover, a smartcard 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 smartcard-based authentication schemes is the administrative burden incurred by the fixed relationship between the smartcard and the user. Since the smartcard is expected to act on behalf of its human user with respect to external parties, the relationship between the smartcard and the user must be corroborated and protected by the higher authority (administration) so that all transactions performed by the smartcard can be accounted for and traced to the associated human user. In order to maintain this relationship, the administration must maintain a one-to-one mapping from users to smartcards. Moreover, it must assure a means for secure distribution, update and revocation of the smartcards.

In this paper we present a novel smartcard design and associated authentication protocols where the smartcard 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 smartcard, the human user can still utilize weak secrets for authentication purposes, but, without any risk of exposure. Furthermore, the smartcard's and the user's secrets are mutually independent, i.e., the smartcard is not associated with any particular user; its only responsibility is to assist the user in the authentication process. Since the smartcard is not personalized, it can be shared by several users. This new concept eliminates the high cost of administration which is typical of existing designs that employ fixed user-smartcard relationship. Moreover, our design require no interface between the smartcard and the workstation (e.g., a galvanic connection) which is difficult to implement on a global scale and retrofit onto existing equipment. We discuss two smartcard variants: one with a built-in clock and one without, and analyze the associated authentication protocols.

The rest of the paper is organized as follows. In the next section we describe the target operating environment. Section 2 provides additional background on traditional authentication schemes, smartcard-based solutions, smartcard features and issues involved with smartcard protocols in general. Section 3 describes the target environment. Section 4 and Section 5 describe the two variants of our smartcard design, the associated authentication protocols and their analysis. Finally, the results of this paper are summarized in Section 6.

[2]Of course, smartcards employing biometric authentication are not susceptible.

2 Background

2.1 Traditional Authentication

In the course of an authentication protocol, the user proves his identity to the authenticator by demonstrating his knowledge of a secret, i.e. a password or a PIN, that is shared with the authenticator. Many authentication protocols suffer from an inherent exposure to masquerading by malicious parties that can spoof, intercept and replay authentication messages. In case when the user's secret is sent in cleartext as in the traditional login procedures, simple spoofing and replay is sufficient to break the protocol. More recent protocols [6, 9] use the user's secret as an encryption key or as a seed from which an encryption key is derived. This measure is only partly useful because, as described in [5], such an encryption key is weak and can be easily broken by wiretappers. Since remembering perfectly random numbers is difficult for most humans, this weakness is based on the lack of randomness in the way humans choose their secrets. In other words, users choose their secrets out of a space that is relatively small in comparison with the minimum key space required by a good cryptographic algorithm. In a typical case, the secret is a password[3] chosen from a dictionary the size of which (on the order of 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}). 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 smartcards [1, 4].

2.2 Smartcard-based Authentication

A smartcard is a device with limited processing capability that contains a cryptographically strong key. Its purpose is to aid user authentication in a ***hostile*** environment. Unlike the weak keys (passwords and PINs) used by human beings, the smartcard's key is randomly chosen out of the total key space of the cryptographic algorithm in use. 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 smartcard operation by authenticating himself using a weak initial secret but this interaction takes place directly between the user and the smartcard (via smartcard'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 smartcard's strong secret. Since the smartcard is a simple, non-programmable device (not unlike a calculator), it is trusted by principals involved.

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

2.3 Smartcard Features

In this section, we address the physical characteristics of the smartcard design. The following features influence both the cost and the security of the smartcard protocols and, the smartcard itself:

- Smartcard-Workstation Coupling
  There are several ways of coupling the smartcard to the workstation, e.g, galvanic, capacitive, inductive and RF interfaces. (Hereafter, we use the term *electronic coupling* to denote all of the above). All of them give the smartcard a means to communicate directly with the workstation without involving the user in transfers of information between the smartcard and the workstation. Also, electronic coupling allows the smartcard to be built without a power supply (battery) of its own since the workstation can provide it the necessary power. Unfortunately, the cost of equipping all workstations with a secure *card reader* (and all smartcards with appropriate receptors) may turn out to be prohibitively high, especially in a cost-conscious environment.

- Interaction Complexity
  A relevant factor is the volume of information that user must exchange with the smartcard. Electronic coupling eases the problem since the interface between the smartcard and the workstation allows for fast information transfer without human involvement. For smartcards without electronic coupling, the user must act as an *intermediary* between the smartcard and the workstation. In order to ensure ease of use, it is typical to trade increased functional complexity in return for reduced interaction complexity. In this respect, an ideal protocol with no electronic coupling would require the input of one bit on the smartcard (e.g., an on/off button and no keypad) and the reading of a number by the user.

- Keypad
  A keypad may be needed to enter into the smartcard the user's secret like a password or a PIN. If a smartcard is not equipped with an electronic interface, other information may need to be entered via smartcard's keypad (i.e., in this case, the user acts as a conduit between the workstation and the smartcard).

- Clock
  A clock may be required for generating timeliness indicators and, possibly, nonces[6]. However, a clock requires a battery which has to be replaced or recharged periodically. In [1], the authors suggest that: "having a clock is particularly difficult because it requires a battery". While a battery is indeed required, a built-in clock does not have to present difficulties. Nowadays, many personal electronic gadgets operate on dry cell batteries without any significant

penalty in cost or performance. Wristwatches, pocket calculators and hearing aides are the most widespread of these. Such devices can either: i) require a change of battery every 2-3 years, or ii) be disposable.

- Display
  A display is imperative when there is no coupling between a smartcard 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 smartcard'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.

- Volatile Storage
  Temporary, volatile storage is necessary to store certificates, session keys, etc., for the duration of an authentication session. It is, of course, desirable to minimize the size of volatile storage.

- Encryption/Decryption Ability
  The complexity of the encryption algorithm influences the cost and the performance of the smartcard. One possibility is to restrict the smartcard's computation power to perform only secret one-way function (i.e., encryption but no decryption). Not only does this make the implementation simpler, but it also eases export restrictions since, unlike a full-blown encryption algorithm, a one-way function cannot be misused to obtain secrecy.

2.4 Symmetrical *vs* Asymmetrical Cryptography

Some recent smartcard schemes employ asymmetrical (public key) cryptography (e.g., [1, 10]). This has two main drawbacks:

- 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).

3 Target Environment

Throughout the remainder of the paper, a particular operating environment is assumed. We consider a very large distributed system (e.g., a global Internet) with a dynamic user and node (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 soes not have to be registered in any way.[4] Furthermore, workstations, the software they run, and the underlying network are completely untrusted. As an example, consider public terminals currently offered to attendees of some conferences or to guests in some hotels. Typically, users have little trust in such devices.

In contrast to workstations, users are *not* anonymous. Every user is assumed to be registered with at least one authentication server (AS), a highly-secure and universally trusted node. However, users are considered to be very mobile, i.e., a user may request system access from any public workstation.

There are a number of ASs situated throughout the internetwork. An AS has two main duties:

- maintain a database of user records
- act as a *certification authority* (TGS, *a la* Kerberos [9]) for its constituent users

Every workstation is assumed to be aware of at least one AS.

4 *Minimalist* Smartcard With Internal Clock

In this section we present a novel smartcard design aimed at addressing some of the issues and problems discussed above. The motivation for this work is twofold:

- Explore the design of a smartcard with a minimum number of features, hence, one that can be produced at minimal cost.
- Minimize the administrative overhead common to existing smartcard designs.

4.1 Smartcard Features

The smartcard has the following features:

[4] Even in more tightly controlled environments (where workstations are registered and accounted for) workstations are not necessarily trusted by their users.

- *no card-user relationship*: The smartcard we envision 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. A smartcard can be purchased over-the-counter in a retail shop. There is no buyer registration required and users are free to resell, exchange, discard or lend the smartcard to anyone.

- *no keypad*: Since the user enters no data into the smartcard, the smartcard has no keypad but only a single button. This button controls the state of the card (i.e., ON/OFF) as well the sequencing of displayed information within a single authentication session.

- *no electronic coupling*: The smartcard and the workstation function in complete independence of each other;

- *display*: There is an LCD display on the smartcard.

- *clock*: The smartcard has a built-in clock. The clock has no dedicated display. The time (and date) is displayed (i.e, the display is active) only when the smartcard is *ON*. The clock does not need to have high resolution; second precision is sufficient for reasons explained below.

- *cryptographic ability*: The smartcard implements a one-way function, e.g., DES [3] encryption with a secret key. In general, regardless of the cryptosystem used, the smartcard does not need to incorporate the entire algorithm, encryption alone is sufficient.

- *smartcard's secret*: Every smartcard, C, possesses a secret, K_c which is computed as[5]:

  $$K_c = E(K_{as}, SN_c)$$

  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 smartcard 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 smartcard, 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.

The resulting smartcard is depicted in Figure 1.

4.2 User Requirements

Every legitimate user is identified by a combination of: i) a unique user (or login) name, and ii) a password or a PIN. A password may be an alphanumeric string of, say, eight characters, while a

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

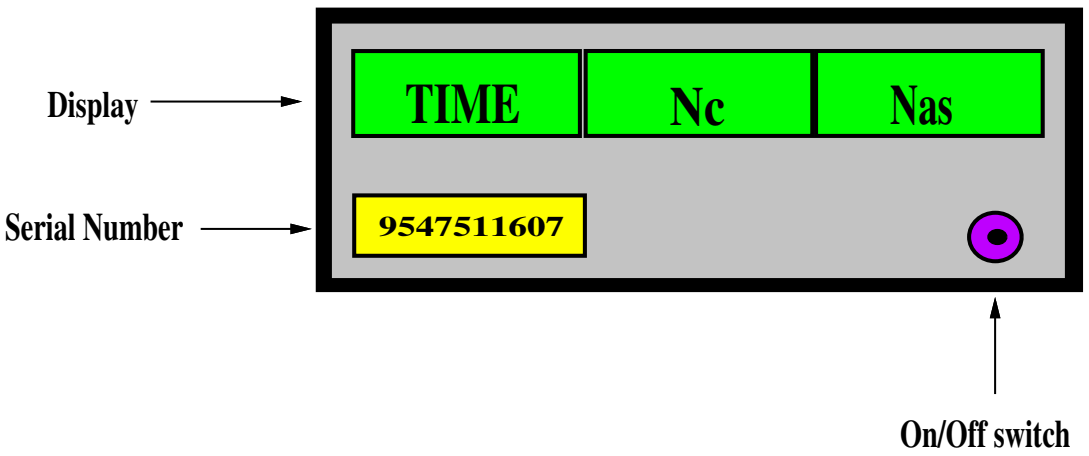

Figure 1: A *minimalist* smartcard

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.

4.3 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* smartcards is the Card Key Generation Key, K_{as}. The AS does not keep track of the identities or secrets of individual smartcards.

4.4 Protocol Steps

The protocol aims to achieve:

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

The protocol consists of the following steps:

1. The user begins by turning on the smartcard using the sequence button. The smartcard 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 "strong" nonce.

2. The user supplies the following values to the workstation:

(a) U - User name.

(b) SN_c - Serial number of the smartcard.

(c) $TIME$ - Current date/time taken from the smartcard's display. (Actually, the entire date/time string need not be entered into the workstation. See Section 4.5.)

(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 difference between N_c and PIN_u. Moreover, it is not the *arithmetic* difference that the user has to compute. For each digit of N_c it suffices to enter the difference between that particular digit and the corresponding digit of the PIN. We use subtraction only as an example operation in computing K_u; digit-by-digit modulo 10 addition can be used as can other methods (see Section 4.5 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 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. As will be seen later, this value authenticates the AS to the workstation and to the user (if the user trusts the workstation).

(b) $N_{as} = E[K_c, f(TIME)]$ - Encryption of $f(TIME)$ under K_c.

In this step, the AS is simultaneously assured of the freshness and the authenticity of the message it received. The authentication of both the smartcard 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 smartcards 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 smartcard's sequence button and reads the authentication value expected from the AS, i.e., $N_{as} = E[K_c, f(TIME)]$, on the smartcard 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 smartcard 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 hashed digest of it (using MD4[7], for example).

   It is important to clarify the meaning of the last step. Most (if not all) existing smartcard-based authentication protocols 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.

7. Finally, the user turns the smartcard *OFF* by pushing the sequence button the last time for this session.

The protocol is illustrated pictorially in Figure 2.

4.5 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.
3. (Optional) visual comparison of $E[K_c, f(TIME)]$ displayed by the workstation and its counterpart displayed by the smartcard.

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 smartcard 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 smartcard) and the user can use the four $ARROW$ keys to modify the displayed value to match the one shown by the smartcard.

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 smartcard with a digit-only keypad, the user can simply enter his PIN via the keypad and leave it to the smartcard to compute K_u. However, a keypad would not only increase the production costs but would also result in the smartcard 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 smartcard, we attempt to make K_u composition easier for the user.

In the protocol description above, N_c is assumed to be an 8-byte number that can be represented by 20 decimal digits. 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 subtracts digit-by-digit his PIN_u from 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 subtraction followed by the 14 remaining digits of N_c. (See Figure 3 for an illustration.) This method requires the ability to perform subtraction of 6 decimal digits digit-by-digit (mod 10). Also, we note that, while in the process of substracting, the user is expected to keep in his head a "running index" of the current PIN digit involved in the subtraction. This may prove to be more difficult that the actual subtraction.

There is, of course, the danger of a user performing the subtraction 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 attached to it physically) a simple 10-by-10 table of single-digit decimal numbers and their differences (e.g., row 9, column 6 will display 3).

2. Alternatively, the display area of the smartcard can be labeled so that each digit of N_c is associated with a fixed number (index) carved on the smartcard 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 remaining 10 digits of K_u are copied directly as the last 10 digits of N_c. The advantage of this scheme is that no arithmetic is required from the user. (An example of composing K_u is shown in Figure 4.)

   The resulting K_u is only 16 digits which reduces its width to 56 bits, from N_c's 64 bits. However, we note that Data Encryption Standard (DES) specifies 56-bit encryption keys [3].

Another usability concern has to do with the requirement that the user enter K_u into the workstation. In the two examples above, K_u is a 16-20 digit decimal number. Entering such long numbers by hand is difficult for most humans. This *feature* is, again, due to the minimalist design. (A smartcard-workstation coupling would certainly alleviate it.)

Another issue of concern is the length of K_u. In the protocol described in Section 4.4, 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 for the duration of the authentication protocol, it would be possible to shorten it to the length of the PIN (i.e., 6 digits).

4.6 Analysis

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

- Every smartcard'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 smartcard.
- The smartcard is trusted to faithfully display appropriate values.

- It is difficult to subvert the smartcard's hardware and obtain the secret (K_c) or otherwise manipulate the smartcard (e.g., set back the clock).
- The smartcard clock is assumed to be monotonically increasing.
- The smartcard 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 smartcard).

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 smartcard 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 smartcards 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.*

Again, more formally, this can be stated as: U **believes that** AS **recently generated** $E[K_c, f(T)]$**.** Assuming that f is one-to-one (e.g., one's complement) $f(T)$ does not form a valid timestamp. Therefore, $E[K_c, f(T)]$ could not have been generated by C in the past. Using the assumption that only the AS and C know K_c, U is assured that AS's communication is authentic and timely.

There is, however, a small caveat. Since the smartcard 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 the smartcard 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.*

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 smartcard, and iii) the user.

One of our principal assumptions is that the smartcard 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 smartcard. Suppose that two users, Alice and Bob, share the same smartcard. 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 smartcard 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-smartcard techniques.

5 Smartcard With External Clock

Earlier in the paper, we argued for making the clock one of the key smartcard features. The protocol presented in the previous section is particularly dependent on the clock (and its correct operation) for computing nonces.

Although we have until now downplayed the difficulties associated with having to maintain a running clock (specifically, a continuously operating battery), it is interesting to consider a smartcard without an internal clock.

Given our reluctance to use galvanic connections for reasons alluded to before, it is imperative for a clock-less smartcard to have a keypad. This is because the smartcard *cannot* maintain any running state without a continuous power supply. Thus, nonce generation is not possible within the smartcard itself, hence, the user must be able to provide some input to the smartcard; the input can then aid in nonce generation.

Apart from the absence of a clock and the presence of a keypad, the other features of our smartcard remain unchanged from its counterpart in the previous section except that the smartcard may function with a solar cell instead of a battery since the power supply requirement is restricted to the time intervals when the smartcard is explicitly on. The resulting smartcard is depicted in Figure 5.

The protocol is as follows:

1. The user begins by switching on the smartcard and entering two values via the keypad:

   (a) ***TIME*** - Current date and time. This can be obtained from the user's watch or from the workstation. In either case the user is expected to recognize grossly inaccurate time

values (e.g., thirty minutes off).

(b) PIN_u - the user's PIN.

2. The smartcard computes and displays: K_u - User's authenticator. However, unlike the previous protocol, $K_u = E[K_c, g(TIME, PIN_u)]$[6].

3. Now the user enters the following into the workstation:

   (a) U - User name.

   (b) SN_c - Serial number of the smartcard.

   (c) $TIME$ - Current date/time; must be the same as in Step 1. Also, the user can avoid this step if workstation-supplied date/time was used in Step 1.

   (d) K_u - User's authenticator.

4. Now the workstation sends to the AS:

   (a) U, SN_c, $TIME$ - all from the previous step.

   (b) $E[K_u, TIME]$ - Encryption of $TIME$ under K_u.

5. AS uses SN_c and K_{as} to compute K_c. Then, it looks up PIN_u (using U), computes $E[K_c, g(TIME, PIN_u)]$ and compares it with the corresponding value 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 (as in the previous protocol).

   (b) $E[K_c, g(f(TIME), PIN_u)]$ - This is as a response to the *challenge nonce* computed by the smartcard in Step 2.

6. The workstation optionally verifies $E[K_u, f(TIME)]$. This gives assurance that the remote party (presumably, the AS) possesses K_u.

7. In order to perform his own verification of the AS the user pushes the smartcard's sequence button and reads the authentication value expected from the AS, that is, $E[K_c, g(f(TIME), PIN_u)]$, on the smartcard display and performs a visual comparison of this value with the corresponding value sent by the AS and displayed by the workstation (see previous step). If the two values match, the authentication is completed. The goal of this comparison is to assure the user that he/she has, in fact, been communicating with the AS (since no one but the AS and the smartcard at hand can compute $E[K_c, g(f(TIME), PIN_u)]$).

The protocol is illustrated pictorially in Figure 6.

[6] g can be any of a number of simple functions, e.g., $\bigoplus$ (exclusive OR).

In addition to the modifications in the smartcard design, the "clockless" protocol differs slightly from protocol described in Section 4.

- The user is required to use the smartcard to input the time and the PIN. This increases the *interaction complexity* of the protocol.

- There is no need to generate a nonce like N_c in the previous protocol to protect the PIN from possible disclosure by the workstation since the user now enters the PIN directly into the smartcard. This makes the user's task easier.

- The structure of the user's authenticator, K_u, has been changed. Simple arithmetic functions such as "-" or "$\bigoplus$" cannot be used for K_u's computation. More generally, only a strong one-way function of PIN_u may be used. The danger of doing otherwise is best illustrated by an example.

  Suppose that a legitimate user, $U1$, uses the smartcard at time $TIME_1$ and obtains an authenticator, $K_{u1} = E(Kc, TIME_1) - PIN_{u1}$. Using the same shared smartcard at a later time, another user (say, U_2) can also supply $TIME_1$ to the smartcard along with PIN_{u2} and obtain $K_{u2} = E(Kc, TIME_1) - PIN_{u2}$. Assuming that U_2 is able to collaborate with the public workstation (e.g., by means of a trojan horse), it can discover K_{u1}. Now, U_2 is able to compute $E(K_c, TIME_1) = K_{u2} + PIN_{u2}$ and, extract $PIN_{u1} = E(K_c, TIME_1) - K_{u1}$. This type of attack is impossible with the previous protocol since the time values are guaranteed to be unique.

- The *challenge nonce* also has been changed from $E[K_c, f(TIME)]$ to $E[K_c, g(f(TIME), PIN_u)]$. The change was made for the following reason. If, as before, the challenge nonce is not a function of the user's PIN, it can be misused. Suppose that U_1 uses the smartcard at $TIME_1$ and obtains a challenge nonce, $E[K_c, f(TIME)]$, which is recorded by a malicious workstation. If U_2 decides to use the smartcard shortly thereafter, it is conceivable that he/she may use $TIME_1$ again. Since time is not kept or checked by the smartcard, the workstation may display the pre-recorded $E[K_c, f(TIME)]$ instead of going to the AS. Hence, it may trick U_2 into believing that the AS was contacted.

  The problem can be solved by making the challenge nonce dependent on the user's PIN. Then, even if identical time values are entered into the smartcard by two successive users, the workstation is not able to achieve any "crosstalk" between the two protocol instantiations.

- An "insider" attack by one of the *legitimate* users sharing the same card with others should be considered in the clockless smartcard case. Having placed a trojan horse program in the workstation and recorded the K_u value generated by the smartcard for user U and time T, an intruder X may use the smartcard to exhaustively try all possible PIN values until a

value identical with K_u is displayed by the smartcard. The matching PIN value would be PIN_u. This attack is not possible in the case of the smartcard with a built-in clock because a card never displays the same N_c value twice. On the other hand, even in the clockless case this attack should not be considered serious because the speed at which the intruder can try different candidate PIN values is limited by the speed of a human entering trial values into the smartcard's keypad. Such an attack could become a serious flaw only if the intruder could try candidate values using computer programs the way a password-based protocol can be broken. In this case, however, only the smartcard itself can be utilized. Assuming a 6-digit PIN, and at least ten seconds to complete each session with the smartcard, approximately 500 days (and nights) of continuous trials are required in order to exhaust half of the possible PIN values. In addition to this extremely low probability of success, in order to perpetrate this attack, the intruder must possess a smartcard used by others during a session spoofed by his trojan horse program.

Some of the attacks described above are made possible because the smartcard allows its users to enter non-contiguous time values. In other words, an intruder attempting to break a user's PIN can enter the same time value as entered by the *victim* user some time in the past. One possible countermeasure is to include a small EEPROM memory cell in the smartcard. Each time a smartcard is used, the time is stored in the EEPROM. Subsequently, the smartcard would reject any time value that is not greater than the one stored. This gives a guarantee that time will always increase. The main drawback of this feature is that it can be misused by an intruder who, by entering a grossly incorrect time (e.g., two months into the future), can deny access to the smartcard for legitimate users.

6 Summary

The mechanisms described in this paper offer the following advantages over existing smartcard-based authentication designs:

- **The smartcard is not personalized, i.e., it is not associated with a particular user.** This property implies several advantages:
  - **No administration cost:** the smartcard does not need to be registered under a user's name or sent to a particular user with safe courier. Smartcards can be freely purchased over the counter (with no special registration procedure) and subsequently shared or exchanged.
  - **Prevention of potential masquerading:** Since a smartcard, by itself, does not represent any user, its theft carries no danger. In other words, a stolen smartcard cannot

be misused in any way to obtain the rights of any of its past or future users.

  - **No PIN storage on the smartcard:** the user's secret does not need to be stored on the smartcard. This eliminates the need for entering, updating and storing user specific secrets (passwords, PINs, biometric patterns, etc.) on the smartcard. This feature is conducive to a low-cost design.

- **The smartcard's secret key is not stored in the AS.** This property offers the following advantage:

  - **Minimum key management requirement:** the AS has to keep only one key to be able to retrieve all the smartcard keys. The management of the smartcard keys has therefore a minimal complexity. The key storage in the AS is independent of the existing smartcard population; addition, update, revocation of smartcards and/or their keys have no effect on the AS.

- The smartcard protocols achieve the above mentioned goals with minimum requirements for smartcard and protocol features:

  - The smartcard design and protocols require no hardware modifications to existing (workstation) equipment. In other word, there is no electronic coupling between the smartcard 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 summary, we discussed a number of issues and shortcoming in today's smartcards and smartcard-based authentication protocols. We introduced a new smartcard design which, in addition to being both simple and secure, lends itself to immediate deployment with very little cost. The smartcard 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, completely avoid the administrative burden typical of smartcard-based authentication solutions.

7 Acknowledgements

This paper has benefited greatly from comments and suggestions made by Todd Arnold, Phil Janson, Liba Svobodova and Don Johnson.

References

[1] M. Abadi, M. Burrows, C. Kaufman, B. Lampson, *Authentication and Delegation with Smartcards*, 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 MD4 Message Digest Algorithm*, Proceedings of CRYPTO'90, August 1990.

[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] T. Woo and S. Lam, *Authentication Protocols* (DRAFT) To appear in ACM SIGCOMM 92, September 1992.

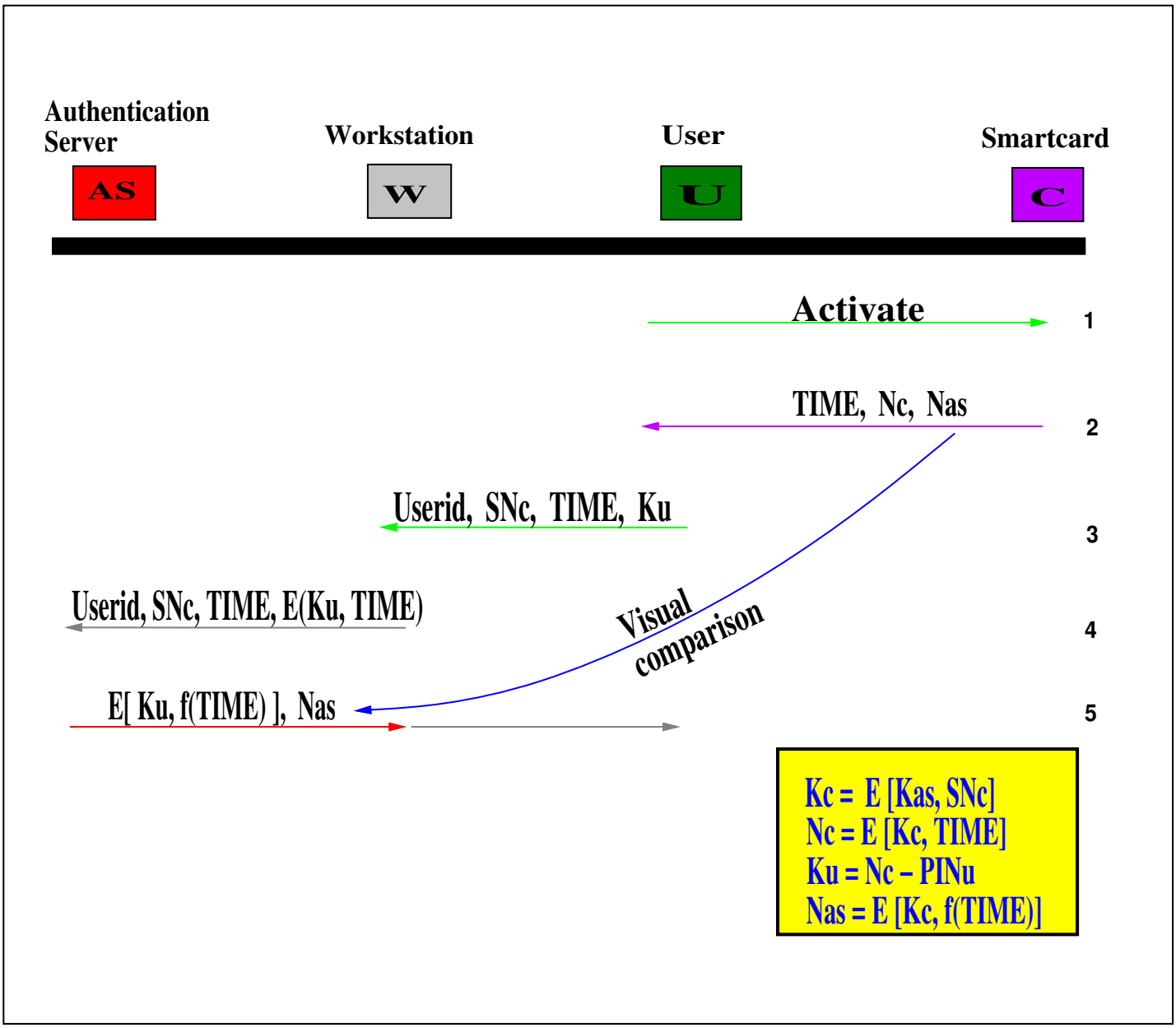

Figure 2: Protocol with internal clock

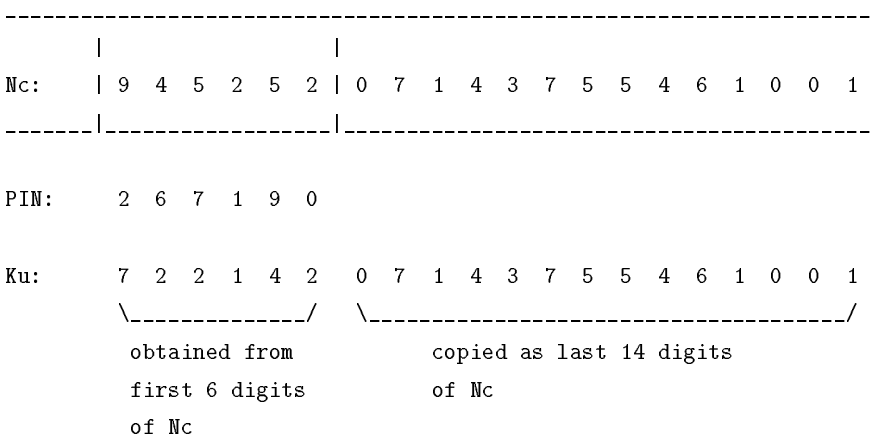

Figure 3: Example of composing $\boldsymbol{K}_u$ (a)

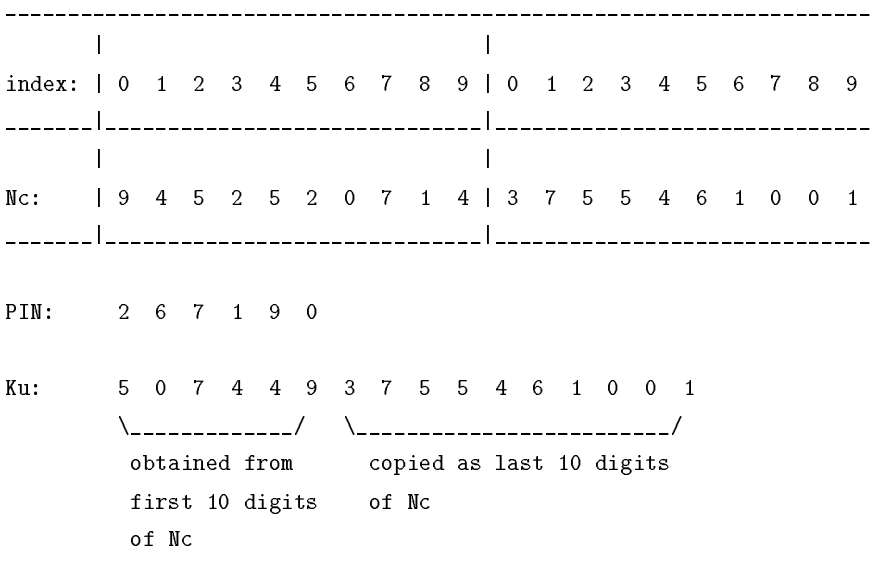

Figure 4: Example of composing $\boldsymbol{K}_u$ (b)

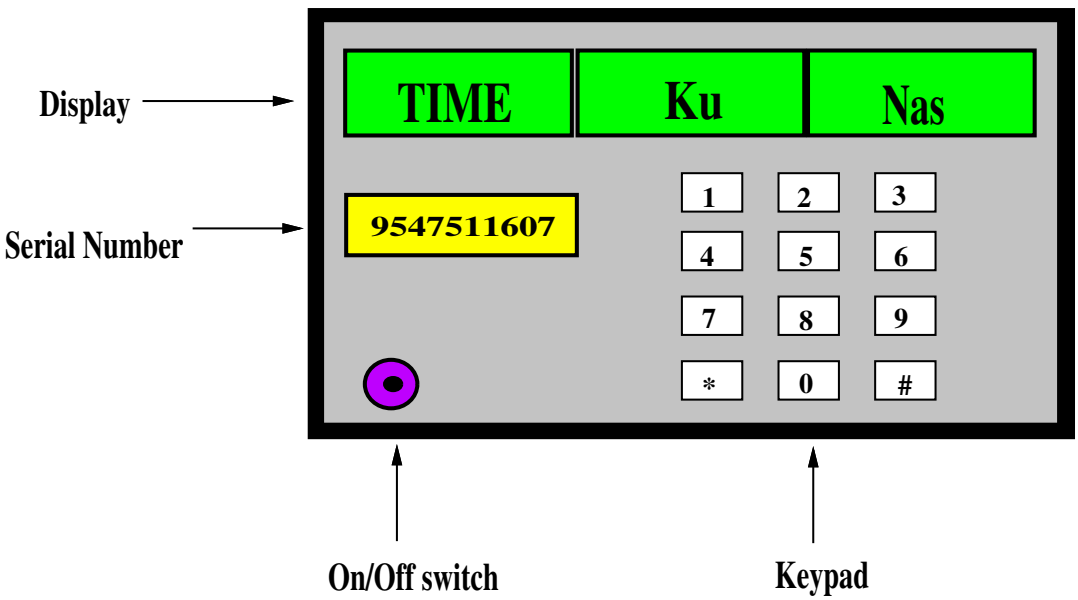

Figure 5: A clockless *minimalist* smartcard

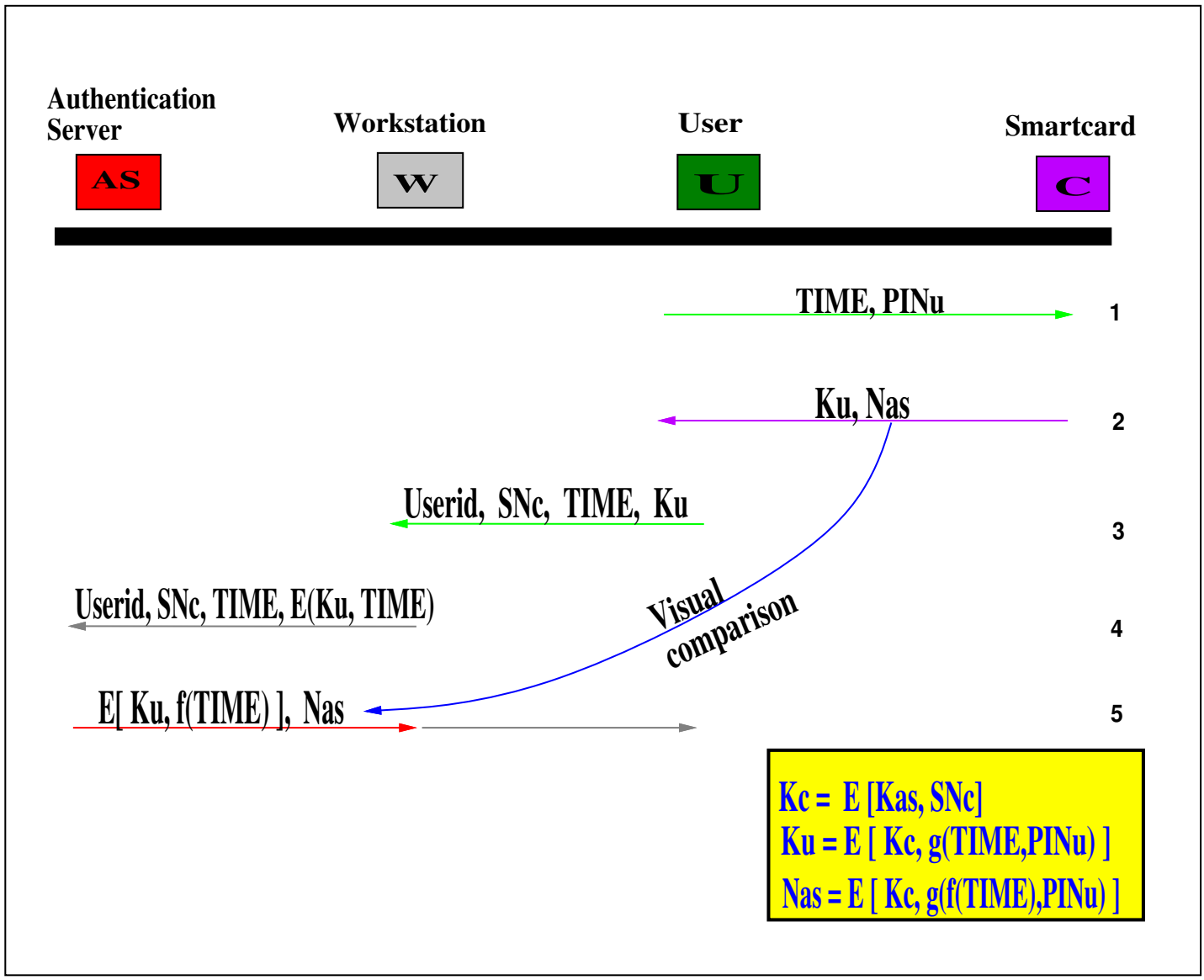

Figure 6: Protocol without a clock