A Personal Information Management Scheme

Jong-Hyeon Lee

Computer Laboratory, University of Cambridge
Pembroke Street, Cambridge CB2 3QG, England

Abstract

There have been many studies on the management of personal information such as PIN code, password, etc. Most schemes include a server containing explicit personal information, and if one can attack this server successfully, then he/she can reach whole personal information on the server. Since information is centralised, the responsibility for services and information is also centralised.

There was an example mentioned by R. M. Needham for the Personal Identification Number (PIN) management scheme for Automatic Teller Machine (ATM) transactions [3]. Implicitly, this example represents design concept for enhanced privacy and the responsibility separation. The concept *responsibility separation* can be regarded as an important design idea for security protocols. Based on this concept, we rebuild his example for practical application. Our scheme provides enhanced privacy, separated responsibility, and nonrepudiation of a bank. It is designed to obtain a cost effective secure system without additional investment to existing structure.

Key words: cryptography; distributed computing; nonrepudiation; personal information management; privacy; responsibility separation; security in digital systems;

1 Introduction

In this paper, the personal information management means controlling mechanisms for users' secret information to obtain access for systems such as password, PIN code, etc.

As an example, R. M. Needham presented an example for the Personal Identification Number (PIN) management for Automatic Teller Machine (ATM) transactions [3]. This example was described as a rough example

which supports his idea on security research under the changing computing environment. However it also represents possibility to achieve enhanced privacy and the responsibility separation implicitly.

The main differences of his example from current ATM systems are the generation and the handling of the PIN code. Most ATM systems use encryption to protect customers' PINs. The detail varies from one bank to another, but many use variants of a system originally developed by IBM, in which the PIN is derived from the account number by encryption [1]. That is, PINs are generated by banks.

In his example, banks do not know customers' PINs, and definitely they do not need to maintain PINs. The PIN code is generated by the customer for him/herself, and is not stored in the bank's database. From the bank's point of view, his example removes bank's responsibility for internal leakage of PINs and its maintenance complexity. Actually, theft by bank staff is a big problem and we can see some cases about leakage of PINs in many way [1]. Banks have a solid defence against an allegation that they negligently permitted the PIN to become known. For customers, they can obtain more privacy by generating and keeping their own PINs for themselves.

2 Analysis of Needham's Example

His example is described under the assumption that a customer has a personal computer and a card writer.

Let H be a one-way hash function. At first, the customer writes on the card a random R and a hash $H(N, B)$ of his/her name N and date of birth B. He/she writes $H(N, B)$ and $H(R, PIN)$ on a floppy disk, where the PIN is chosen by him/herself. He/she then takes the floppy to the bank and says "Please connect $H(R, PIN)$ to my personal details $H(N, B)$ and my account number is 401608 80614874".

A cash machine accepts the card, reads the two quantities on it, works out $H(R, PIN)$ where PIN is the PIN as entered, and sends the two hashes $H(N, B)$ and $H(R, PIN)$ to the bank as shown in Fig. 1. Note that there is an assumption that the hash is good, the PIN is never sent to the bank even in encrypted form. The bank looks up $H(R, PIN)$ where in a substantial in-memory table. If it is found, the table yields the $H(N, B)$ for checking and also gives the account number.

In this example, there are two principals and one intermediate: a cus-

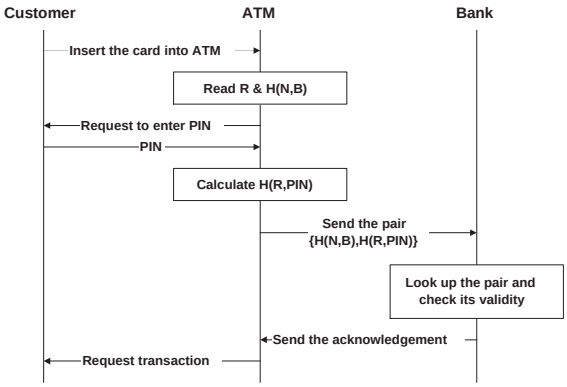

Fig. 1. Needham's example – ATM transaction

tomer and a bank are principals and an ATM is an intermediate. A banking transaction is performed between the customer and the bank, but authentication procedures are done by the ATM and the bank. The ATM checks validity of owner of the card by PIN entered, the bank checks customer's information and account number by $H(R, PIN)$.

A replay attack for the pair of hash values $\{H(N, B), H(R, PIN)\}$ is possible on the communication line between an ATM and the center. If an attacker takes the hash pair from the line, he/she can replay this pair on the same line to be authorised as a valid customer by the center. Since the pair does not include temporarily changing quantum, it can be used at any time by the attacker. The temporal quantum should be combined with the pair not to be separated from the pair. The mechanism to avoid such an attack should be considered.

The hash $H(R, PIN)$ is used for the searching key in bank's database, but it is possible to obtain multiple tuples with the same hash, because the random and PIN are generated by customers, not by centralised one body. Even though the hash function is good enough and collision-free, it is possible to have another tuple with the same pair $\{H(N, B), H(R, PIN)\}$, since the random R and PIN are chosen by a user. We need a different searching key for the database.

When the ATM is not involved in transaction body or card issuing bank, it is possible for the transaction bodies or card issuing banks to insist that they did not confirm transactions although they confirmed them. If the ATM company cannot prove their confirmation, the company is responsible to pay for odd transactions. In this respect, it is necessary to have nonrepudiation feature for the transaction.

In an engineering aspect, it is necessary to define the specification of memory map in the card such as addresses for random, hash pair, and so on. Some identifiers are needed such as the bank identifier, the customer

identifier, or the card number. If there is no standard of memory map for ATM transactions, the applicable card and the way how to write on the card are provided by the bank. Then it is also necessary for the bank to verify whether it is issued by the bank or not. The signature of the bank can be a solution for such a need.

3 The Modified Scheme

Let $S_B(M)$ denote the signature of a bank B on a message M. To have the unique searching key in bank's database, when an account is open, the bank generates the serial number SN for the account. If a customer asks a blank card for ATM transactions, the bank signs on SN, stores this signed serial number $S_B(SN)$ on the card, and issues the card to the customer. The value $S_B(SN)$ is also used to verify that the card is issued by the bank.

Against replay attacks, we adopt a nonce. When a transaction is invoked, the ATM generates a transaction id T. This id consists of the ATM id, transaction time, the date, and a random. After obtaining random R and PIN, the ATM calculates $H(T, H(R, PIN))$ and sends the triple $\{T, H(N, B), H(T, H(R, PIN))\}$ to the bank. The bank checks validity of $H(T, H(R, PIN))$ for customer authentication.

For nonrepudiation of the bank, the above transaction id is used. When the bank completes customer authentication, it signs on T, stores T in the transaction log, and sends the id with bank's signature to the ATM. Then the ATM checks signature of the bank and stores this signed transaction id for proof of the transaction. If there is a trusted third party in existing banking transaction, this transaction id and its signed value can be stored in the trusted third party. Otherwise, we do not require additional authority to provide nonrepudiation of the bank.

When the card issuer is not the same as the dedicated transaction bodies, some transaction bodies can be involved in the transaction between the ATM and the issuer such as credit card companies, mileage service providers, etc. In this case, the transaction id can be used in these all transaction bodies: The id T is tossed to next body with the signature of the current body. Eventually, all signatures of participating bodies can be found in this value, when the ATM receive this signed id. The returned id is of the form $S_{issuer}(S_{cardcompany}(S_{bank}(T)))$. Unless there is missing signature, the ATM perform customer's request. Otherwise, the ATM sends back transaction failure to all the participating bodies and retries the same transaction.

The modified scheme is described in two procedures: the card generation procedure and the ATM transaction procedure. The card generation procedure is almost the same as that of Needham's example: First, a customer obtains a blank card with $S_B(SN)$ from the bank. The customer generates random R and his/her PIN, calculates $H(N, B)$, and writes $H(N, B)$ and R on the card. The customer then sends the pair $\{H(N, B), H(R, PIN)\}$ and SN to the bank in a secure way such as face-to-face off-line transfer, secure mail system, or encrypted email using PGP [1] . Then the bank links this pair to the customer's bank account by matching SN, and sends the acknowledgement to the customer.

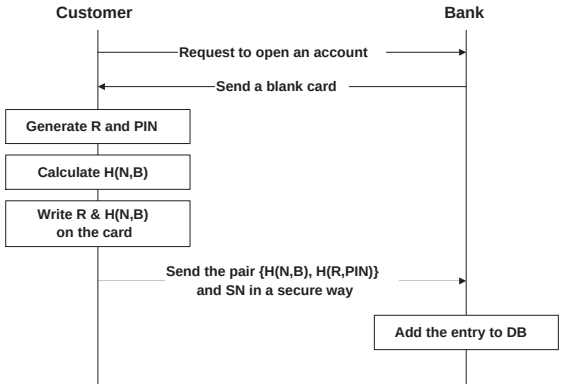

Fig. 2. The modified scheme – Card generation

The ATM transaction procedure is as follows: When a card is inserted to the ATM, the ATM requests the PIN to the customer and calculates $H(R, PIN)$, where PIN is the PIN as entered. Then, the ATM generates a transaction id T, calculates $H(T, H(R, PIN))$, and sends the triple $\{T, H(N, B), H(T, H(R, PIN))\}$ to the bank. The bank checks $S_B(SN)$ to verify whether this card is issued by the bank, extracts SN from the signature, and checks the validity for $H(T, H(R, PIN))$ and the pair $\{H(N, B), H(R, PIN)\}$ by using SN.

The re-issuing procedure is the same as that for new customers. The serial number for the card also should be generated again, because previous card can be found and be used by attackers.

4 Discussion

Needham assumed that the PIN is never sent to the center even in encrypted form. It requires the customer's trust in both the ATM and its

[1] Pretty Good Privacy — a collection of public key cryptographic tools which was released by P. R. Zimmermann

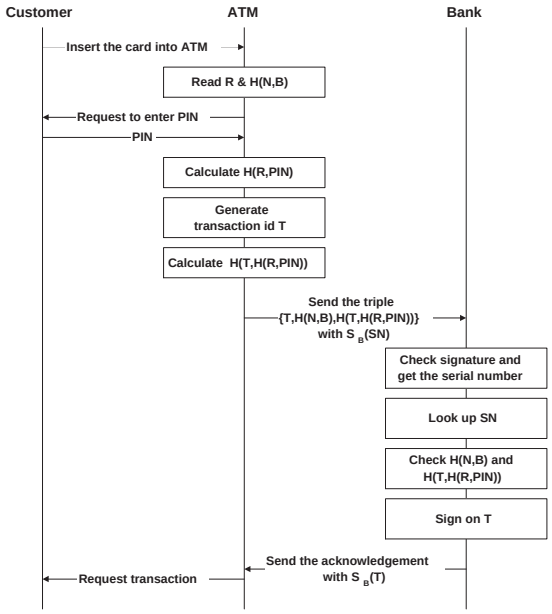

Fig. 3. The modified scheme – ATM transaction

operator such as a bank or a credit card company. In his example, the role of the ATM is more important than that of existing ATMs. A false-terminal attack using a corrupted ATM is also available. Needham mentioned this attack and said that some kinds of smart card can be used to defend such an attack.

If we adopt a smart card in our scheme, and the calculation and the PIN handling can be done by the card in the ATM, then we can separate the role of the ATM, delegate it to the smart card, and do not need to trust the ATM. It means that we can build a trust-free architecture. Without additional processor in customer's side, there is no way to construct trust-free architecture. But the use of smart cards contradicts minimal change of current ATM systems, because all card readers in ATMs have to be changed. It also can raise an argument to use the public key cryptosystem rather than a hash function. For Needham's example and our modification, either memory card or magnetic card is enough for implementation.

Consider UNIX password management scheme as an application of this scheme. The scenario is as follows: First, the user generates a random R and a password P, calculates $H(R, P)$ and $H(N, U)$ for the user's name N and the user id U, make the file `.pin` with the permission `400` in his/her home directory containing the entry $H(N, U)$ and R, and registers $H(N, U)$ and $H(R, P)$ to the administrator of the system. Then the administrator adds the user's information in the password database file. When a user enters his/her user id and password, the login process reads `.pin` file from the user's home directory, calculates $H(R, P)$

with the password P as entered, and checks the validity of the pair $H(N, U) \oplus H(R, P)$.

This password management scheme has a similar mechanism to *salt*-added password management scheme [6], but the random is generated by a user and it is maintained by the user unlike salt-added mechanism. Actually, there is no more benefit of salt-added mechanism for preventing attacks on single user's password than the conventional UNIX password management scheme. With our scheme, attacks for single user's password are harder than in salt-added mechanism. Furthermore, we can build a user-responsible password management scheme.

There have been many trials to use hash functions rather than public key cryptosystem. Some of them have accomplished similar level of security to a widely used implementation using public key cryptosystem. IBM's KryptoKnight is such an example [2]. From a user's perspective, KryptoKnight provides services and facilities which are very similar to those of Kerberos [5], but it uses protocols based on well-known Needham-Schroeder scheme [4] and Message Authentication Code (MAC).

With simple design, our scheme provides similar services which can be provided with public key cryptosystem. If we adopt public key cryptosystem, we have to adopt additional facilities such as certification authority, revocation authority, etc. Although basic function of public key cryptosystem is easy to implement and secure, it requires additional functions to certify valid public keys.

5 Summary and Future Work

In order to fill up gaps in Needham's scheme, we adopted some procedures: nonce generation, signatures on the nonce, the serial number for a card, and the bank's signature in a cards. Due to checking with transaction id including random, the replay attacks are not available in the modified scheme. Bank's signature on the transaction id provides nonrepudiation of the bank for each transaction. The signature of bank is used for verifying that the card is issued by the bank. As a result, the modified scheme provides protection against replay attacks, nonrepudiation of a bank, the unique DB searching key, and bank's card verification.

The Needham's example is a light-weight and efficiently-organised scheme with enhanced privacy. Furthermore, it provides responsibility separation between a client and a server. It reflects the changing paradigm in the application of security protocols.

We believe that the concept *separated responsibility* is one of the most important concept in the resilience of security protocols. Our scheme can be an example which can provide high resilience without high complexity.

Acknowledgements

We are grateful to Dr. Ross J. Anderson, Geraint Price, Kan Zhang, and Harry Manifavas for their helpful comments and advice.

References

[1] R. J. Anderson. Why cryptosystems fail. *Communications of the ACM*, 37 (11):32–40, 1994.

[2] R. Molva, G. Tsudik, E. van Herreweghen, and S. Zatti. Kryptoknight authenication and key distribution system. In *ESORICS '92*, volume 648 of *LNCS*, pages 155–174. Springer-Verlag, 1992.

[3] R. M. Needham. The changing environment for security protocols. *IEEE Network*, pages 12–15, May/June 1997.

[4] R. M. Needham and M. Schroeder. Using encryption for authentication in large networks of computers. *Communications of the ACM*, 21(12):993–999, 1978.

[5] B. C. Neuman and T. Ts'o. An authentication service for computer networks. *IEEE Communications*, 32(9):33–38, Sept. 1994.

[6] B. Schneier. *Applied Cryptography*. John Wiley & Sons, New York, 2nd edition, 1996.