**Zeitschrift:** Bulletin des Schweizerischen Elektrotechnischen Vereins, des Verbandes Schweizerischer Elektrizitätsunternehmen = Bulletin de l'Association suisse des électriciens, de l'Association des entreprises électriques suisses

**Herausgeber:** Schweizerischer Elektrotechnischer Verein ; Verband Schweizerischer Elektrizitätsunternehmen

**Band:** 77 (1986)

**Heft:** 1

**Artikel:** Requirements for Modern Security Systems

**Autor:** Beker, H. J.

**DOI:** https://doi.org/10.5169/seals-904136

# Requirements for Modern Security Systems

H.J. Beker

***Until recently almost all electronic security systems in communications and computers have been retrofits to existing networks. As a result they are often cumbersome in protocol and expensive. In order to reduce the cost and processing overhead it is necessary to design computer and communication networks with the security system as an important aspect of that design. In this paper the concept of a modern security system will be described by the example of a financial institution network.***

***Bis heute resultieren fast alle elektronischen Sicherheitssysteme in Kommunikations- und Computersystemen aus nachträglichen Verbesserungen von existierenden Netzwerken; ein schwerfälliges Konzept und ein hoher Preis ist die Folge. Eine Verbesserung kann nur erreicht werden, wenn das Sicherheitssystem als wichtiger Aspekt in die Planung eines Computer- und Kommunikationsnetzwerks miteinbezogen wird. Als Beispiel eines modernen Sicherheitssystems wird in diesem Aufsatz das Kommunikationsnetz eines Finanzinstituts beschrieben.***

**Address of the Author**
Prof. *H.J. Beker*, Royal Holloway College,
University of London.

## 1. Data Security in Financial Institutions

Until recently almost all electronic security systems designed for network security have been implemented as retrofits. This leads to compromises in design, cumbersome protocols and usually an expensive solution. The keyphrase for this paper is *system design*. To provide a cost-effective solution for data security on a network it is imperative to design that network with security as one of the requirements. In this way not only can cost be reduced, but the processing overhead minimized and normally a higher security level achieved.

To begin, let us consider a typical financial institution and its networking and security requirements. On 13 May 1985, the Financial Times in its 'World Banking Survey' said:

> Security and integrity of electronic transactions, in particular, has become a hot topic for international bankers as the full significance of the consequences of a failure in any of the existing or proposed new systems sinks home.

What are some of the networks that require security? We can identify six distinct areas currently envisaged (and in some cases already adopted) by many Financial Institutions. These are:

1. Inter Financial Institution payment systems, e.g. consortium banking, clearing facilities such as CHAPS,
2. Intra Financial Institution systems, e.g. branch to central computer communications,
3. alarm traffic,
4. corporate banking,
5. retail banking,
6. personal banking.

Clearly with so many networks under consideration an evolutionary systems approach is essential. The alternative will involve a number of expensive gateways and more important an incorrect distribution of security which may result in the whole system being rendered vulnerable because of a single weak link somewhere in the system.

Before considering the specific requirements for any part of this system, let us consider some of the implications of the new technology that is available to us. The clear advantages of this technology are improved and cheaper data security facilities which in turn are leading toward agreed standards. This in itself could substantially reduce the cost still further. The disadvantages, however, include easier access to data for hackers, the wide availability of bugs, the increasing power of home computers and perhaps, most important, the amount of success and resulting publicity of present-time hackers. This is undoubtedly encouraging would-be hackers to have a go.

It cannot be overemphasized that when an institution or corporation sets up an electronic service and its associated security system it will not be easy to change. The cost of updating or replacing all the equipment will be enormous. Thus a security system installed today must remain secure during the lifetime of the equipment. The institution will almost certainly be unable to take advantage of further technology improvements. The hacker suffers from no such handicap. He can fully utilize whatever technology becomes available to him, for instance, the increasing power of home computers. The vulnerabilities of a system lie everywhere: the hardware, the software, the databases, the archives, the people running the system and of course, the communications.

In general, the communications risks can be partitioned into two classes. Firstly, there are the *passive* attacks. These are attacks where the 'frandster' simply monitors the communications. The second class of attack is that called *active*. This involves the hacker tampering with the information transmitted. It might include alterations to the information, deletion of information or falsely originating information. It might even include

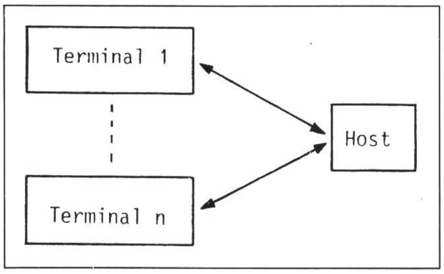


**Fig. 1 Terminals Communicating with a Host**

totally blocking the transmission channel and hence denying the service.

Whilst emphasizing that the system must be considered in its entirety, we shall concentrate on one particular aspect of this typical Financial Institutions network; namely the communications security for corporate banking. We shall see that within a particular environment with knowledge and understanding of that particular application, a simple applications specific protocol may be conceived.

Let us consider the simple scenario shown in Figure 1. Within this architecture we are assuming a multiplicity of terminals communicating with a host. These terminals may be specific to an application or might be more general PCs. They are characterized as follows:

1. They (and their associated security) must be low cost,
2. they will be in an insecure environment,
3. there will be a large number of terminals deployed,
4. there will be a large transaction volume at the host.

The security requirements can be stated as follows:

1. Part of the transaction (at least any password employed) must be protected from an eavesdropper, i.e. encrypted,
2. the transaction must be protected from change, it should not be possible to insert or delete transactions into the network without this being detected,
3. all parties involved in a transaction must authenticate each other,
4. if a terminal is able to communicate with a number of hosts, then no one host should suffer from the effects of security failures in other hosts,
5. the system must require a minimum of manual intervention and should be as automatic as possible; this applies in particular to key management.

It should be noted that 1...4 require a cryptographic protection of messages and this involves the use of cryptographic keys applied to all messages and transactions. Ideally we require a uniquely defined key for every transaction. The logistic problem of handling these keys can be enormous. They must be generated, distributed, stored and managed (e.g. destroyed at the right time). Key management has long been acknowledged as one of the most important and most difficult problems to solve. Almost all general key management schemes are cumbersome, expensive, and in themselves often difficult to manage. The result may be insecurity.

There is one further point to appreciate in this particular environment. Low cost terminals in an insecure environment lead to a vulnerability of the terminals themselves; they will be stolen, they will be tampered with. Some tamper-resistance will of course be provided, i.e. it will not be easy for someone to break into a terminal and discover key variables—and in so doing they may well need to destroy the terminal—but it will not be impossible.

Let us now consider a protocol that can be applied in this environment to not only satisfy all the above criteria, but also to provide automatic management of the keys while restricting any would-be hacker and cryptanalyst of the system to at most a single transaction.

The protocol we shall describe is based on a unique-key-pertransaction scheme originally developed for use within retail banking environments. In order to discuss the general principles of this system, the remainder of this paper will be divided into the following sections:

1. The use of MAC (Message Authentication Code) residues
2. Formulation and use of transaction key
3. Advantages of the technique

We shall begin our discussion on achieving the stated security requirements by assuming that we have a current key at both the terminal and the computer which will be securely changed after each transaction (the key management problem itself will be addressed in Section 2). We shall also assume, although this is clearly not mandatory, that our system is based on the encipherment procedure defined by the Data Encryption Standard (DES) developed at the National Bureau of Standards and by IBM.

## 2. Description of a Special Message Protection Protocol

### *2.1 The Use of MAC Residues*

Given DES and a key, achieving encryption and message protection is straightforward. In fact, standards already exist for this, e.g. ANSI X9.8 and ANSI X9.9. Thus, assuming that we adhere to those already existing standards we need only consider how this might be extended to achieve our remaining requirements. In order to do this, let us examine ANSI X9.9 a little more closely. Avoiding the detail, ANSI X9.9 involves the generation of a cryptographic (i.e. key dependent) check sum to be appended to a message. The recipient of the message can compute this check sum and if it agrees with that received, then he knows, with a high probability, that this message has not changed in any way since its origination. The check sum is called the Message Authentication Code (MAC). ANSI X9.9 describes a procedure for generating a 64-bit check sum of which 32 bits form the MAC and the other 32 bits are discarded. What I shall now describe is how use can be made of these discarded 32 bits which I shall term the MAC residue. The basic idea is to ensure that no unauthorised messages or parts of messages can be inserted into a transaction. It will also allow detection of deletions and furthermore enhance the authentication under the assumption that the key changes securely with each transaction. Essentially the procedure is as follows:

*Step 1:* The terminal generates and sends its message in the normal way but retains a copy of the MAC residue.

*Step 2:* The computer in responding to this message generates its check sum on a concatenation of the response message and the MAC residue of the request message.

*Step 3:* The computer transmits only the message with its MAC.

*Step 4:* The computer retains a copy of the MAC residue from the response message.

*Step 5:* The terminal can compute the MAC for the response message from the received message and the stored MAC residue. If it agrees with that received, not only has this message not been changed, with a

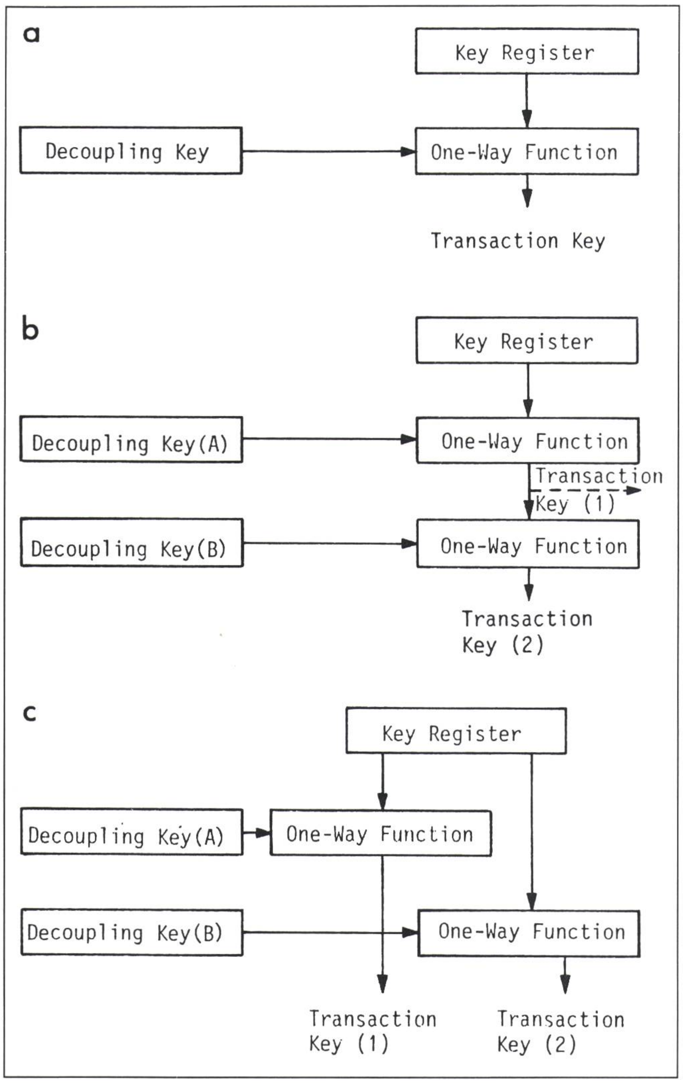


**Fig. 2**
**Transaction Key**

high probability, but furthermore the terminal can have confidence in the entire transaction so far since the MAC is now effectively computed on the entire transaction, not only on the message just received. The terminal stores the MAC residue corresponding to the response message.

*Step 6:* The transaction proceeds in the same way with, in each case, a message MAC being produced from the concatenation of the message to be sent and the MAC residue from the previously received message.

This technique provides the required transaction protection at almost no extra computational cost and without the requirement for any message extension; whilst still conforming to all existing standards.

### *2.2 Formulation and Use of Transaction Key*

We now turn our attention to the mechanism for ensuring that the key is securely changed for each transaction. We assume that the terminal holds a key register (i.e. 64 bit) for the computer with which it must communicate. We also make the assumption that the terminal has access to a decoupling key. A decoupling key is simply a piece of information known at the two ends of the link (i.e. at the computer and terminal). We shall now address this problem in three parts:

a. the formulation of the transaction key at the start of a transaction,
b. the updating of the key register at the completion of a transaction,
c. the decoupling key.

#### a. The Transaction Key

The transaction key is formed simply by combining the key register contents with the decoupling key through a one-way function. The result is a transaction key. The one-way function which may, or may not, be DES-based, is simply a two-input, one-output function which ensures that given the output, the input cannot be determined, even if one of the inputs is already known (Fig. 2a). Clearly, iterations of the above process can also be used, if necessary using the output from the one-way function as an input together with a second decoupling key through the same one-way function, thus producing a transaction key. Similarly, a second transaction key can be generated by utilizing the key register with a second decoupling key and so on (Fig. 2b, 2c). Thus, one can generate more than one transaction key if the system requires it. An example of this might be the requirement for separate keys for message encipherment and message protection.

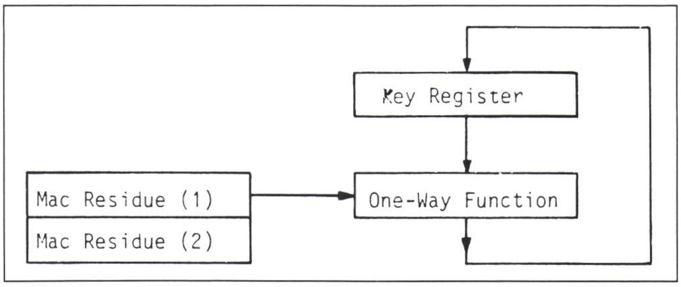
#### b. The Updating of the Key Register at Completion of a Transaction

At the end of a transaction, the key register is updated by taking the key register contents together with the last two MAC residues as inputs to a one-way function. The output is the new key register contents (Fig. 3).

#### c. The Decoupling Key

The decoupling key can be produced in many ways. Four examples

**Fig. 3**
**Updating of the Key Register**



are given below. Other techniques are currently being developed for use in particular applications.

1. The decoupling key could be based on a card-key; i.e. some data on the magnetic stripe of a plastic card that is read by the terminal but not transmitted; and held by the computer. (This relies on a card reader at the terminal.)
2. The decoupling key could be produced off-line, on a personalised token of some kind, as part of the log-in password to authenticate the user; i.e. to produce the correct decoupling key would require the personal token plus a personal password.
3. The decoupling key could be generated during the previous transaction using a Diffie-Hellman key exchange as described in "New Directions in Cryptography".
4. The decoupling key could be based on some transmitted data; e.g. the User Account Number. The true decoupling could then be effected by use of the MAC residue, making it dependent on some non-transmitted data. One example, in an EFT-POS environment might be a cryptographic one-way function of a card-key from the previous transaction, which can be computed by the terminal and can be securely delivered to the computer by a third party, e.g. a card issuer. This is advantageous in the case when the card-key is not held by the computer with which the terminal communicates.

### *2.3 Advantages of the Technique*

1. The transaction key is end-to-end and unique to the particular transaction.
2. If an unauthorised person gained knowledge of a transaction key and the next decoupling key that would *not* be sufficient to deduce the next transaction key. Similarly no information would be gained regarding the previous transaction. Thus the rewards for breaking a single key are indeed small.
3. Key management is automatic and a transaction key is unpredictable since it depends on:
   a. the key register contents,
   b. a decoupling key,
   c. the entire contents of the previous transaction.
4. Confirmation that a transaction completed is inherent in the next communication between terminal and computer. Of course nothing precludes the users from using confirmation messages if they desire.
5. Log-ons are not required for the security system itself and the system needs no interruption for key updates.
6. If someone breaks into the terminal and obtains the key register this compromises no past transactions.
7. If someone breaks into the terminal and obtains the key register this compromises *no* future transactions or at most one future transaction provided the MAC residues are dependent on some non-transmitted information which cannot be deduced by an interceptor with feasible computation power.

In this paper we have addressed one problem within a complex financial institution's security system. This example illustrates how, with knowledge of the application, an applications specific protocol can be devised to provide increased security at reduced cost.