Source: National Institute of Standards and Technology, National Computer Security Center (1989) Proceedings. 12th National Computer Security Conference, Baltimore, Maryland, October 10-13, 1989 (NIST, Gaithersburg, MD).

June 30, 1989

A TOKEN BASED ACCESS CONTROL SYSTEM FOR COMPUTER NETWORKS

Miles Smid, James Dray, and Robert B.J. Warnar

NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY

Supported by the Defense Advanced Research Projects Agency under order 6373.

ABSTRACT

This paper describes a Token Based Access Control System (TBACS) developed by the Security Technology Group of the National Institute of Standards and Technology (NIST). TBACS replaces traditional password based access control systems which have often failed to prevent logins by unauthorized parties. A user's access to network computers and resources is mediated by a smart token implementing a transparent cryptographic three-way handshake with the target computer. The token's onboard processor and memory are exploited to provide sophisticated security mechanisms in a portable device. In addition to access control, the TBACS token may be used for random number generation, cryptographic key generation, data encryption, data authentication, and secure data storage.

I. INTRODUCTION

A computer is a valuable resource which should be protected. The information within the computer should be protected from unauthorized disclosure and modification, and the computing power should be limited to authorized users. The recent rash of computer viruses and the previous successes of hackers in gaining access to computer systems indicates that many computers are not properly protected. Inadequate protective measures are not justified by the statement that the computers contained only unclassified data. The failure to control access to a computer's resources (whether classified or not) has serious consequences.

Most computer systems attempt to protect their resources by authenticating the identity of each user attempting to login. Once the user's identity is established, the system then controls the access of the user to resources based upon some predetermined access control policy.

Unfortunately, as we progressed from localized stand-alone systems to distributed processing systems on large networks, it became easier to subvert the traditional protective mechanisms. At one time, access to a computer's resources could be controlled by limiting the access to the room where the computer was physically located. Today computers are networked so that remote users may take advantage of distributed resources without having to be physically co-located. We now rely on password systems which are not up to the task of protecting computer resources.

In theory, there are three types of information for authenticating the identity of computer users [1,2]:

1. Something the user KNOWS (such as a password)

2. Something the user POSSESSES (such as a token), and

3. Some PHYSICAL CHARACTERISTIC of the user (such as fingerprints or other biometric data).

In practice, most computer systems use only the first type of information (e.g. passwords) to authenticate user identities. Password systems predominate because they are inexpensive and they appear, upon first examination, to be easy to use. Password systems do not provide the highest level of security. If properly implemented, password systems can provide effective security [3]. However, these systems are seldom properly implemented. Time and time again we hear about cases where a user selected a trivial password, the user wrote down or shared a password, the operating system debuggers left well known passwords in the system, or the passwords were transmitted over an unprotected channel in the clear.

The owners and users of most computer systems have not been willing to suffer the expense and the effort associated with

token and biometric based authentication systems. A major exception to this rule has been the retail banking community. Most users of Automatic Teller Machines are accustomed to the fact that in order to obtain their money, they must produce a bankcard as well as a password known as a Personnel Identification Number (PIN). These systems have had some security problems but it is generally acknowledged that they are superior to password-only applications. If all computer systems required tokens for access, most hackers would be prevented from entering systems to which they were not authorized.

The cost of electronic technology has decreased substantially over the last ten years making biometric based authentication much more feasible. Biometric systems are now being considered for limited high security applications. Although, biometric systems still have a significant cost, they may some day become the standard in authentication systems.

Password systems alone are not as easy to use, in a secure manner, as some previously thought.

1. If passwords are randomly generated, they are written down. If passwords are generated by humans, they can often be guessed.

2. If a user needs a different password for each computer to which access is permitted, then the user becomes frustrated and writes the passwords down.

3. If the communications link between the user terminal and the host computer is unprotected, then a line tapper can determine the password and later login as the user.

This paper describes a Token Based Access Control System (TBACS) which is being developed by the National Institute Of Standards and Technology (NIST). The first version of TBACS will use a single user password and a smart token containing cryptography to reduce or eliminate several of the drawbacks associated with password systems. Later versions may employ biometrics for increased security as that technology becomes more cost effective.

II. DESIGN REQUIREMENTS

TBACS was designed by NIST to satisfy the following requirements:

1. TBACS shall be easy to use. A TBACS user only needs to remember one password for all computer systems to which the user has access. The TBACS user authenticates to the token via the password, but does not have to type any challenges or responses. The token authenticates the user to all computers (the user workstation and remote hosts).

2. TBACS shall implement the mechanisms for cryptographic authentication as well as cryptographic key storage on the token itself. The closer the security to the user the better. Once inserted, keys will not leave the token.

3. TBACS shall be consistent with existing government and American National Standards Institute (ANSI) standards. The token implements the Data Encryption Standard (DES) cryptographic algorithm specified in Federal Information Processing Standard (FIPS) 46 [4], and could also be used to authenticate computer data and messages as specified in FIPS 113 [5], ANSI X.9 [6], and ANSI X9.19 [7]. TBACS is consistent with Draft American National Standard for Financial Institution Sign-On Authentication for Wholesale Financial Systems (ANSI X9.26) [8].

4. TBACS tokens shall have the capability to store additional information such as sensitivity labels and other access control information.

5. TBACS shall be capable of serving multiple security needs. Although TBACS token is primarily designed for user authentication, it can also be used for random number generation, cryptographic key generation, low speed encryption, low speed Message Authentication Code calculation, and secure data storage. Future versions of TBACS could function with biometric authentication devices.

NIST decided that the best way to ensure that all its requirements were met was to specify the exact command set that the token would implement. In addition to implementing the desired capabilities, security could be improved because only a limited well defined command set was allowed.

III. SYSTEM DESCRIPTION

The NIST secure computer network research model consists of a Sun workstation connected to an Ethernet with one or more hosts (Figure 1). Each computer on the net is interfaced to a token reader/writer system. Access to the net is granted after a predefined sequence of authentications have been completed between the user, the token, the workstation, and any selected computers on the network.

When the token is inserted into the reader/writer, a C-language program in the workstation starts the login sequence by making calls to commands implemented in the token. The user is prompted for the user identifier (ID) and a Personal Identification Number (PIN) which, if correct, authenticates the user to the token. From this point on, the token acts for the user to perform a mutual DES based cryptographic authentication with the workstation and any other hosts to which the user is permitted access (Figure 2).

A. Hardware

The smart token consists of a plastic carrier containing a microprocessor and nonvolatile memory. The carrier has the same major dimensions as a standard credit card, with six recessed metallic contacts along one edge. The reader/writer provides the following electrical connections to the token via the six contacts: power, ground, hardware reset, clock, serial data in, and serial data out. The reader/writer connects to the workstation through a standard asynchronous serial communications port, eliminating the need for a custom communications interface.

TBACS is designed to operate with workstations operating under UNIX (TM), which implement the DES in hardware using a crytographic chip set. The use of personal computers (PCs) as workstations will also be supported.

B. Software

NIST designed a set of sixteen individual token commands. Several of these commands must be executed in a predefined sequence. The sequence is controlled by a set of flags which are checked each time a command is performed. If the flags are not in the expected state, the system will return an error and the current command will not be executed.

The commands are grouped into three classes: the Security Officer (SO) commands, the user to workstation commands, and the user to host commands. The SO commands provide for the initialization of tokens including the loading of cryptographic keys, host IDs, and PINs. The token is ready to be issued to the user after the SO has completed the "loading process".

The token key table contains the host IDs and the

Figure 1.

NIST Secure Computer Network Model

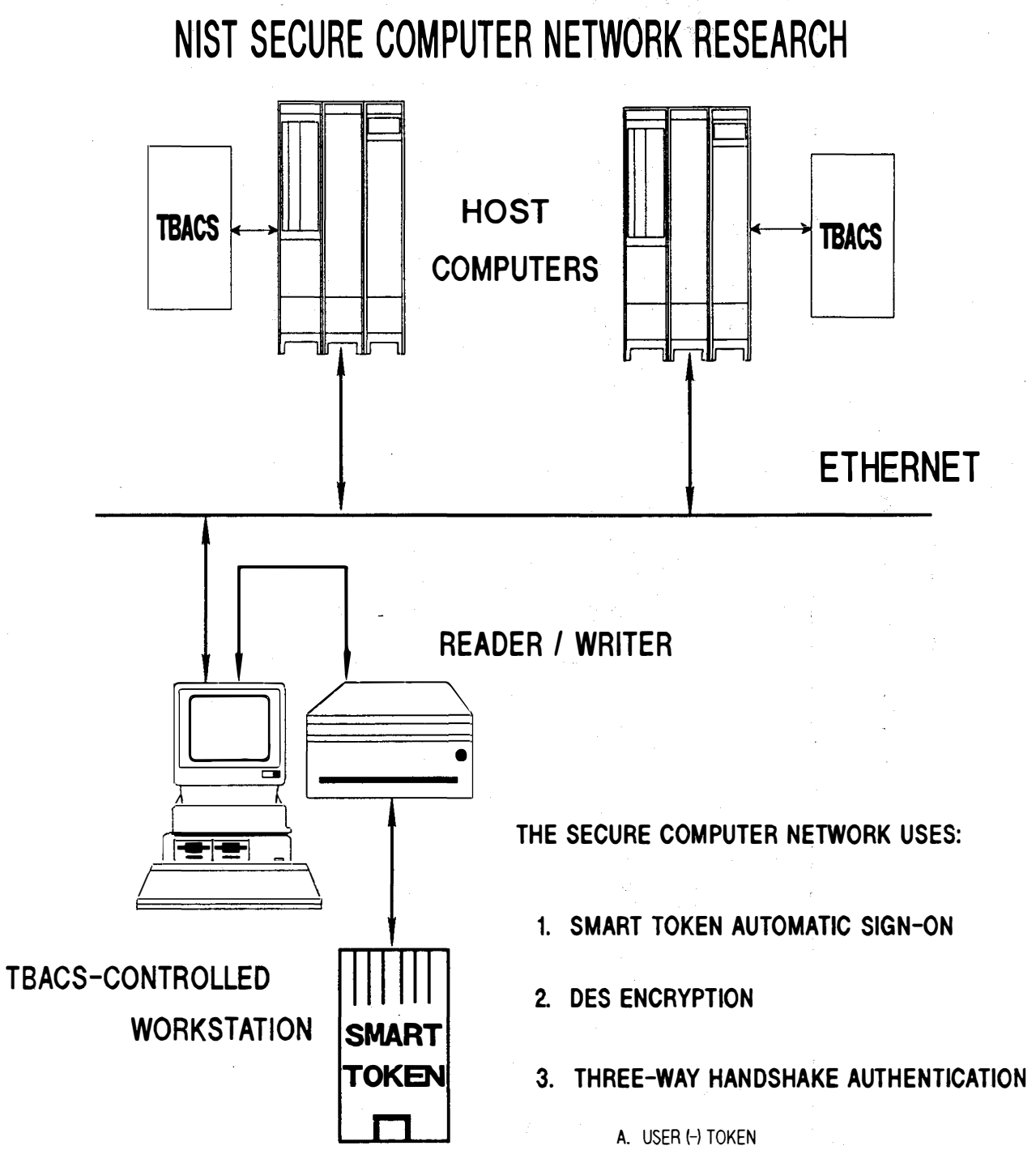
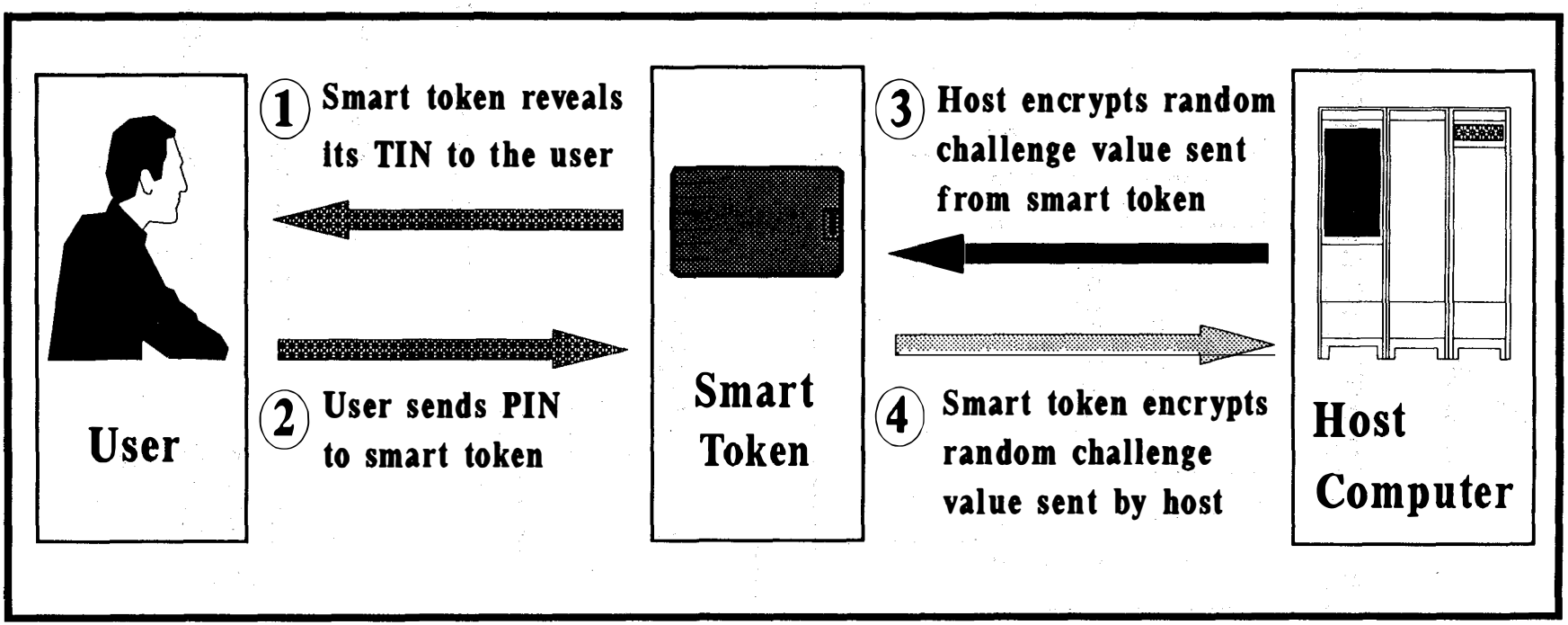


Figure 2.

Mutual Authentication in the TBACS



corresponding cryptographic keys. The design supports 100 cryptographic keys for 100 different hosts connected on the Ethernet. The host IDs and keys are part of a set of the parameters that must be entered by the SO during token initialization. The token uses the keys in this table to perform encryption and decryption processing during workstation and host authentications.

A software simulation program has been written in C which implements the operations of the token as defined by its command set. The simulation forms the main part of the detailed system specification and is used to specify the system. The simulation consists of sixteen functions, one for each token command, plus a small number of internal functions. The total simulation consists of about 2500 lines of code.

The workstation software must interact with the user token through the reader/writer. It must also act as an intermediary in the authentications between the user and the token and between the token and the workstation cryptographic module. If the user wishes to login to a remote host, the workstation software must implement the necessary communications protocols and prompt the token to perform authentication functions as required. The workstation will have security officer controlled software for enrolling new users. The workstation software will store or be able to calculate keys for all valid workstation users.

The software of the network host computers must be able to communicate with the user workstation. Like the workstation, it must have security officer controlled software for enrolling new users and maintaining keys.

IV. AUTHENTICATION PROCESSES

In order for a user to gain access to computing resources on a network using TBACS, a series of authentications between the smart token, the user, and various host computers must be performed. TBACS selectively controls access to all computers on the network, including the user's local workstation. By taking advantage of the processing capabilities of the smart token, the login process can proceed transparently to the user while providing a high level of security. The DES algorithm, operating firmware, and critical data are stored internally on the smart token.

A. USER/TOKEN AUTHENTICATIONS

When a user begins the login process on a workstation, the user should have some means of determining the identity of the token. A program called the "login manager" is executed on the workstation when the user initiates a login, and is responsible for mediating the required series of authentications between the user, the token, and the workstation. The first step performed

by the login manager is to request the token identification number (TIN) from the token and display it on the user's screen for visual verification. The user can choose to either continue the login process or abort. If the user chooses to continue, the user must prove his identity to the token. The login manager prompts the user for the user PIN, which is then encrypted by the workstation and transmitted to the token along with the user ID. The token decrypts the user PIN and uses it as the key to encrypt the user identity. The result is then compared to the value stored on the token, and if these values match the token accepts the identity of the user as authentic. From this point on, TBACS uses the token to authenticate the user's identity to other computers.

B. THREE-WAY HANDSHAKE

The three-way handshake is the authentication protocol used between the token and the workstation and between the token and the remote host(s). This protocol allows each party to prove that it possesses the same cryptographic key as the other party [9] (Figure 3). This protocol works as follows:

1. Party A generates a 64-bit random number and transmits it to party B.

2. Party B encrypts the random number using its DES key, generates a second random number, and transmits it to party A.

3. Party A decrypts the first number and verifies the result. Party A then encrypts the second random number and transmits it to party B.

4. Party B decrypts and verifies the second random number. At this point, each party is satisfied that the other party possesses the DES key corresponding to the claimed identity. Therefore both parties are implicitly authenticated.

C. USER/WORKSTATION AUTHENTICATIONS

After the user and token authenticate to each other, the token must authenticate to the workstation. To perform the authentications between the workstation and the token, the login manager requests a random number from the token. The three-way handshake then proceeds with the token acting as party A and the workstation as party B. If this handshake is completed successfully, the login manager terminates and the user is logged in to the system.

D. USER/REMOTE HOST AUTHENTICATIONS

At some point during a session, the user may decide to connect to a remote host via the network. The user activates a remote login

manager, which requests a table of the allowed TBACS hosts for this user from the token and displays this table in a menu format. After the user selects the desired remote host from this menu, the remote login manager connects to the remote login server on the remote host. At this point, the local remote login manager acts primarily as a transparent communications path between the token and the remote login server. The token is provided with the host ID, which it uses to select the proper key for subsequent cryptographic operations. The steps of the three-way handshake are then performed between the token and the remote login server on the remote host. Finally, the remote login server terminates and the standard remote login process connects the user to the remote host.

E. SEQUENCE CONTROL

In order for the steps which accomplish the authentications required by TBACS to function, some mechanism for ensuring that these steps are executed in the correct order must be provided. This is a critical design consideration, since the overall security of the system is dependent on this order. TBACS controls the order in which the authentication steps are executed through a set of "sequence flags" stored internally on the token. These flags are individual bits in the token's memory, which are set in sequence upon successful completion of each step. The flags are checked at the beginning of the next step. Since the flags and the mechanism for controlling them are internal to the token and no external access is provided, it is difficult to defeat the correct sequencing of steps.

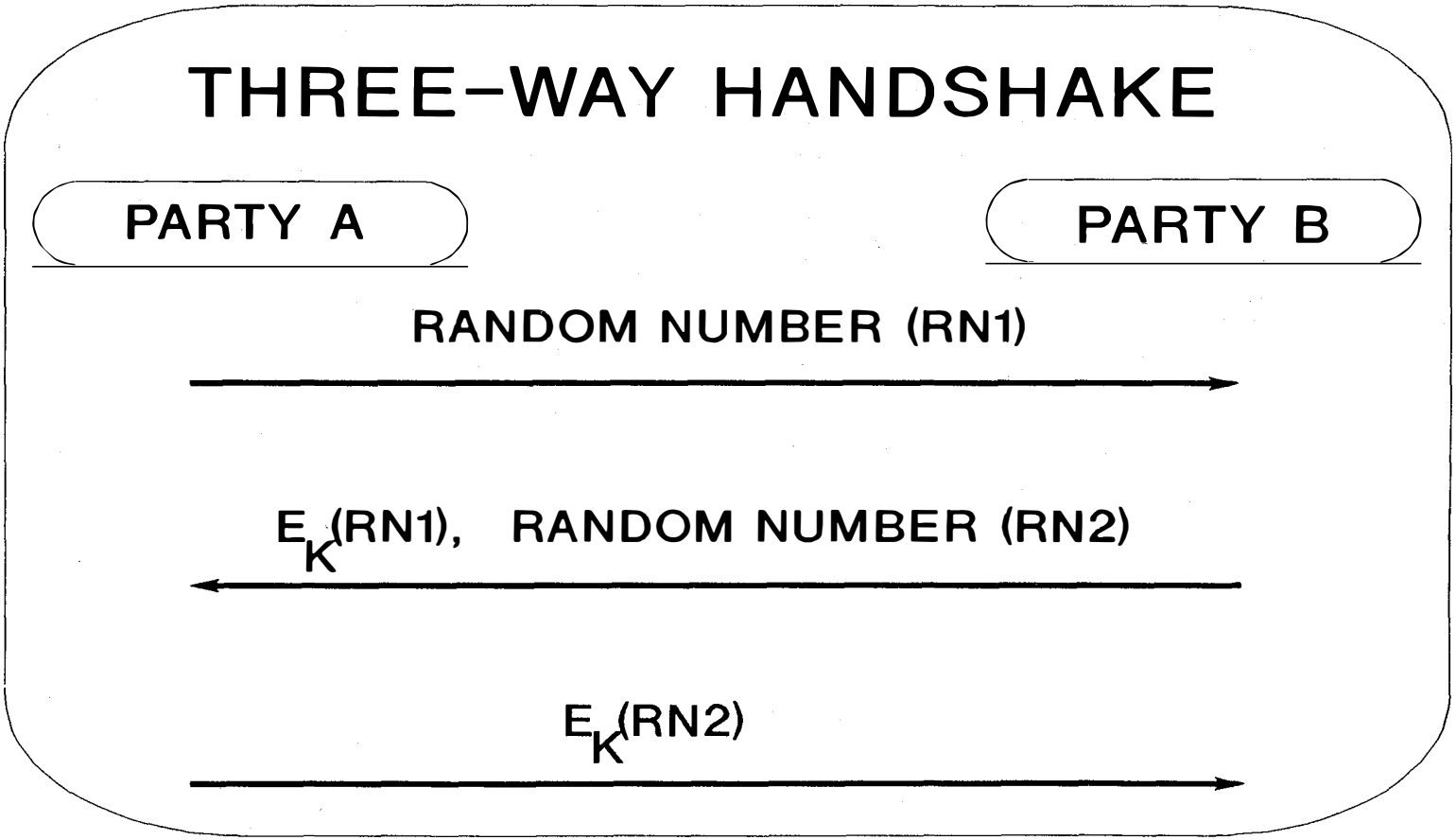
F. TOKEN DEACTIVATION

In addition to sequence control, the TBACS token is capable of deactivating itself when certain conditions are detected. Deactivation is accomplished by deleting the internal token identification number, after which none of the authentication steps required for user login will execute. A token is reactivated when a security officer installs a new token identification number. All prior user data is retained when a token is deactivated, avoiding the problem of rebuilding this information when the token is reactivated. The conditions which cause a token to deactivate itself are as follows:

1. Three failed login attempts. The token maintains a failure log, which is incremented each time a login fails.

2. Token expiration date is reached. The token contains an expiration date, which is compared to the current date at the beginning of each login session.

Figure 3.
Three-Way Handshake



V. KEY MANAGEMENT

In the TBACS system a user has a separate DES key for each computer on which the user is permitted access. When a user first wishes to enroll on a TBACS computer, the user must contact the computer's security officer. The security officer initializes a blank token by loading the security officer ID encrypted using a security officer PIN, the token expiration date, the user ID encrypted using an initial user PIN, and a token identification number. After receiving the token from the security officer, the token user may reset the PIN to a new value by supplying the current PIN value.

The security officer initiates a process which generates a DES key and stores the key on the token encrypted using the user's PIN and indexed by computer's identification. The DES key is also stored in the computer's key database indexed by the user's identity. This key database replaces the password database currently used on most computers.

The user may now enroll on another TBACS computer by contacting the appropriate computer security officer. As previously described, the security officer initiates a process which generates a DES key and stores the key in the token and in the computer's key database. The TBACS token is designed so that only the security officer who first initialized the token can delete token keys. Other security officers can only append keys to the token key table.

In some situations it may be desirable to eliminate the key database stored in the computer. One possible method for accomplishing this task is to assign a single master key to the computer. This master key can be easily stored in the host computer's encryption module for extra security. DES keys for user tokens are generated from the master key by encrypting the user ID using the master key. Whenever the user attempts to login the user DES key is regenerated by again encrypting the user ID using the master key. Thus, only a single secret master key needs to be maintained by the computer or its encryption module.

VI. OTHER CAPABILITIES

A. Random Key Generation

The primary purpose of the token is to generate random challenges and to perform the encryption of challenges as part of the three-way handshake used in the authentication process. However, the token can be used as a portable key generator. The token can be commanded to generate a 64-bit random number which may be used to derive a DES key by the workstation or host cryptographic module.

B. Encryption

The token can also be used for data encryption. Both the Electronic Codebook and the Cipher Block Chaining modes are supported [10]. The communications overhead required to pass the data between the reader/writer and the token along with the overhead of the algorithm may make encryption of large amounts of data impractical. Nevertheless, it may be feasible to encrypt human interactive terminal to host communications. The token can also be used as part of an automated key distribution system to decrypt new cryptographic keys sent from the host.

C. MAC Calculation

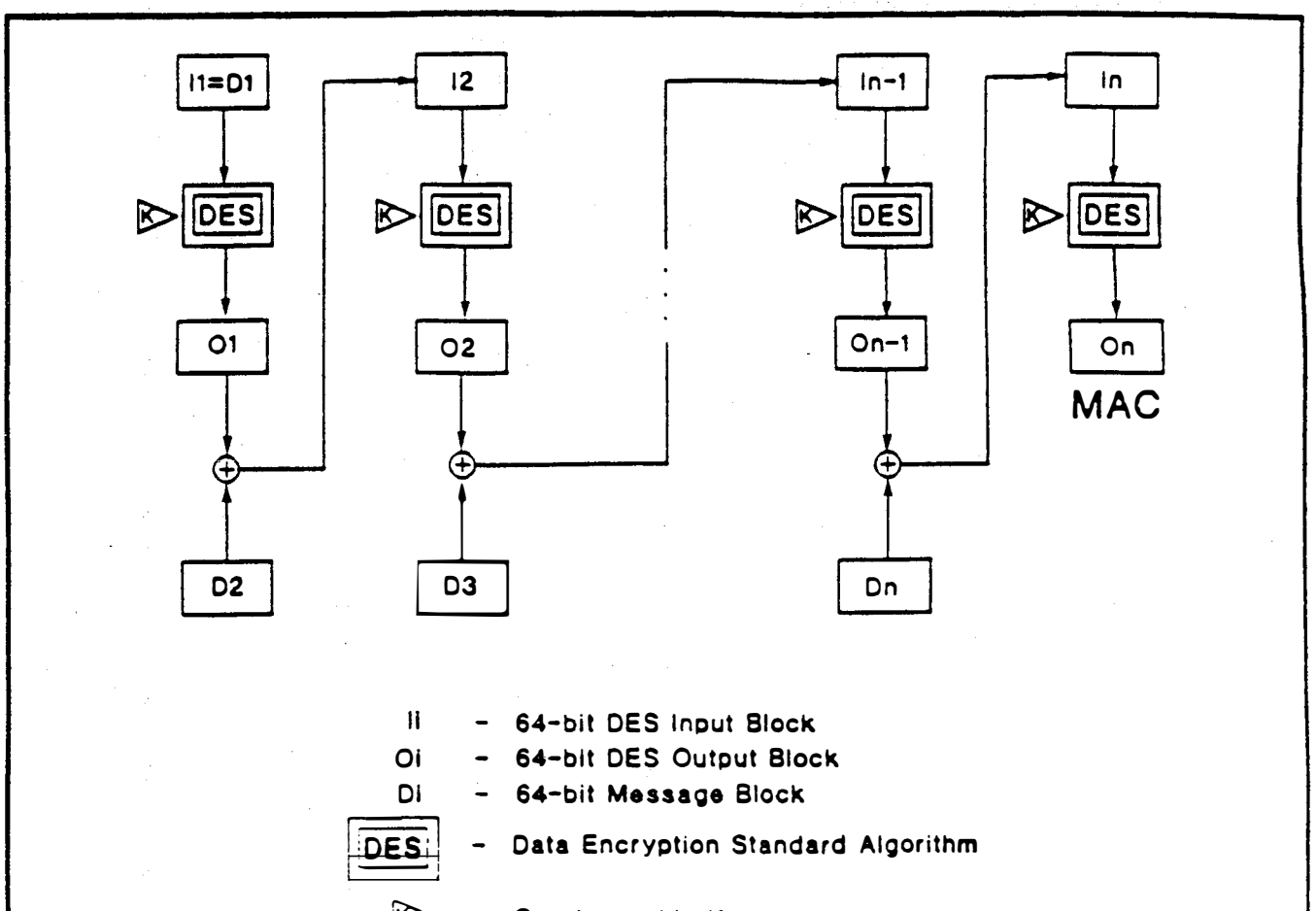
The token may be used to detect unauthorized modifications to messages by calculating a Message Authentication Code (MAC) as defined in ANSI X9.9 [6]. This algorithm is currently being used to authenticate Electronic Funds Transfer (EFT) messages worth trillions of dollars each day. The MAC computation is similar to Cipher Block Chaining encryption except that the MAC is selected from the last cipher block (Figure 4). The unencrypted data and the MAC are transmitted to the receiver. The receiver performs the MAC computation on the received message and compares the computed MAC to the received MAC. If the two values are equal then the message is accepted as unmodified. If the two values are not equal an unauthorized modification is assumed. As with data encryption, MAC computations on large messages may prove time consuming using the token. However, a message digest algorithm may be used to reduce a large message to a few 64-bit blocks which are then MACed by the token.

D. User Authorization Code Storage

The TBACS token can store user authorization codes which may control user access to information in the workstation or host computers. These codes can be passwords or read/write permissions for specific files or categories of files. A code may also indicate the security level of the user to help enforce mandatory access controls. The possible benefits of storing access control information in a token rather than in the target computer is a topic for future study.

Figure 4.

ANSI X9.9 DES Based MAC Calculation



VII. CONCLUSION

Smart tokens can play a major role in solving access control and other security problems. The computational capability of smart tokens can be used to perform cryptographic functions to authenticate users and protect data from disclosure and modification. Smart tokens permit cryptographic security mechanisms to be moved closer to the user where they may be protected by the user. Smart tokens can also provide conveniences for the user which make improved security requirements acceptable.

REFERENCES

1. Beardsley, Charles W., Is Your Computer Insecure? IEEE Spectrum, IEEE, Inc., New York, NY, January 1972, pp. 67-68.

2. Walker, Burce J., and Ian F. Blake, Computer Security Protection Structures, Dowden, Hutchinson and Ross, Inc., 1977.

3. Password Usage, National Institute of Standards and Technology (U.S.), Federal Information Processing Standards Publication 112, National Technical Information Service, Springfield, VA, May 1985.

4. Data Encryption Standard (DES), National Bureau of Standards (U.S.), Federal Information Processing Standards Publication 46, National Technical Information Service, Springfield, VA, April 1977.

5. Computer Data Authentication, National Institute of Standards and Technology (U.S.), Federal Information Processing Standards Publication 113, National Technical Information Service, Springfield, VA, May 1985.

6. American National Standard for Financial Institution Message Authentication (Wholesale), ANSI X9.9-1986, American Bankers Association, Washington, DC.

7. American National Standard for Financial Institution Message Authentication (Retail), ANSI X9.19-1985, American Bankers Association, Washington, DC.

8. Draft American National Standard for Financial Institution Sign-On Authentication for Wholesale Financial Systems, ANSI X9.26-198x, Draft 6.1, American Bankers Association, Washington, DC.

9. Smart Card Technology: New Methods for Computer Access Control, National Institute of Standards and Technology, Special Publication 500-157, National Technical Information Service, Springfield, VA, September 1988.

10. DES Modes of Operation, National Bureau of Standards (U.S.), Federal Information Processing Standards Publication 81, National Technical Information Service, Springfield, VA, December 1980.

APPENDIX A: TOKEN COMMAND SET

1) COMMAND: 00- RESET

   INPUTS: NONE

   PURPOSE: To allow for recovery from a critical error by resetting the token's temporary global variables to their initial state at power-on. The values stored in non-volatile memory are not affected.

2) COMMAND: 03- Enter SO PIN

   INPUTS: SO PIN, SO ID, Token expiration date

   PURPOSE: This command allows an SO to initialize a blank token by entering the required input parameters. After this command has been executed, only this SO will be able to enter the user PIN, null a value in the key table, or reactivate a token.

3) COMMAND: 04- Authenticate SO

   INPUTS: SO PIN, SO ID

   PURPOSE: To authenticate the SO by matching the input parameters against those stored on the token. Flag F2 is set upon successful completion.

4) COMMAND: 05- Enter User PIN

   INPUTS: Old User PIN, New User PIN, User ID

   PURPOSE: Allows SO to enter User PIN onto the token. The ID is encrypted under the PIN and then stored. This command can also be executed by the user in order to change the value previously stored on the token.

5) COMMAND: 06- Load Key

   INPUTS: Host ID, Key, User PIN

   PURPOSE: Allows an SO to load a host ID and corresponding key onto the token, granting the user access to that host. The token encrypts the key under the user PIN and stores the resulting value.

6) COMMAND: 07- Authenticate Token

INPUTS: Workstation ID, Random Number (RN1), date (YYYYMMDD)

OUTPUTS: Token PIN

PURPOSE: To verify the authenticity of the token to the user. The workstation displays the TIN to the user for verification.

7) COMMAND: 08- Generate Challenge

INPUTS: Workstation ID

OUTPUTS: Random Number (RN1)

PURPOSE: This command is the first step of the three-way handshake authentication. The workstation ID is stored for later use in key selection, and a random number is generated, stored and transmitted back to the workstation.

8) COMMAND: 09- Authenticate User

INPUTS: eK(user PIN XOR RN1), user ID

PURPOSE: Verifies the authenticity of the user based on the user PIN and ID. The user PIN is decrypted, extracted from RN1, and then used as the key to encrypt the user ID. The resulting value is then compared to the value stored on the token.

9) COMMAND: 10- Change Token PIN

INPUTS: (old token PIN), (new token PIN), workstation ID

PURPOSE: Allows the user or SO to change the current token PIN. If the old token PIN matches the value stored on the token, the new PIN is stored.

10) COMMAND: 11- Workstation Verify and Respond

INPUTS: eK(RN1), RN2

OUTPUTS: eK(RN2)

PURPOSE: To complete the final steps of the three-way handshake between the token and the workstation. The workstation encrypts the random number (RN1) received from the previous generate challenge command and generates a second random number (RN2). These values are sent to the token as input parameters for this command, which decrypts and verifies RN1. RN2 is encrypted and sent back to the workstation, which then decrypts and verifies it. This completes the three-way handshake.

11) COMMAND: 12- Output ID Table

INPUTS: none

OUTPUTS: Data block containing host IDs from key table

PURPOSE: Transfers the token's table of host IDs to the workstation, which uses this information to display a menu of available hosts to the user. Since the ID table may be larger than the capacity of the buffer, this command returns a NACK each time it is executed until the entire ID table has been transferred, at which time an ACK is returned. The workstation software checks this return value and repeatedly executes this command until an ACK is transmitted.

12) COMMAND: 13- Host Verify and Respond

INPUTS: eK(RN1), RN2

OUTPUTS: eK(RN2)

PURPOSE: Completes the three-way handshake process between the token and a remote host. This command is analogous to the workstation verify and respond.

13) COMMAND: 14- Read Zone

INPUTS: zone name

OUTPUTS: Contents of the specified zone

PURPOSE: To access the contents of a memory zone.

TABLE OF PERMISSIONS FOR ZONE COMMANDS

ACCESS TYPE:

ZONE	READ	WRITE	APPEND
0	all	user	user
1	user	none	SO

14) COMMAND: 15- Write Zone

INPUTS: zone name, data block

PURPOSE: To transfer data to a given memory zone on the token.

15) COMMAND: 16- Append Zone

INPUTS: zone name, data block

PURPOSE: To append data to a given memory zone on the token.

16) COMMAND: 17- CALLDES

INPUTS: 2-byte mode selector:

Bit 0 - set new key
Bit 1 - encrypt/decrypt
Bit 2 - load B from input buffer
Bit 3 - xor two input values (A ^ B)
Bit 4 - produce output

16-byte key or padding(required)
16-byte ASCII hex data string A
16-byte ASCII hex data string B (optional)

OUTPUTS: NACK or ACK and 16-byte result, unless output is suppressed (bit 4 of mode byte is 0).

17) COMMAND: 19- TEST

NOTE: The inputs consist of a 1-byte mode selector and additional parameters which are dependent on the mode selected, as follows:

MODE:		
	0	1
INPUTS:	data	none
OUTPUTS:	data	f_log) "NULL"

PURPOSE: **This command provides the following test modes:**

0- Echo data
1- Current token status