IMPLEMENTING AN ACCESS CONTROL SYSTEM WITH SMART TOKEN TECHNOLOGY

DRAFT COPY

April 12, 1989

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

National Institute of Standards and Technology

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

INTRODUCTION

The National Institute of Standards and Technology is currently developing a Token Based Access Control System (TBACS) which employs a single user password and a smart token with cryptographic capabilities. This system is designed to eliminate many of the problems associated with password-only systems. Later versions may employ biometrics for increased security as that technology becomes more cost-effective.

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 commercial standards. The token implements the Data Encryption Standard as specified in Federal Information Processing Standard (FIPS) 46 [1], and could be used to authenticate computer data as specified in FIPS 113 [2], ANSI X.9 [3], and ANSI X9.19 [4]. TBACS is consistent with Draft American National Standard for Financial Institution Sign-On Authentication for Wholesale Financial Systems (ANSI X9.26) [5].
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.

SYSTEM DESCRIPTION

The system configuration for TBACS consists of a number of workstations and host computers interconnected by a communications network (Figure 1). Each workstation on the network is connected to a reader/writer device, which provides the electrical interface between the TBACS token and the workstation. When the user inserts a token into the reader/writer, a program running on the workstation manages the authentication process by issuing a sequence of commands to the token and receiving the token's responses to these commands.

HARDWARE

The smart token consists of a plastic carrier containing a microprocessor and non-volatile memory (Figure 2). 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 (Figure 3). 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 Sun workstations under UNIX, which implement DES in hardware using the Advanced Micro Devices (AMD) cryptographic chip set. The use of IBM Personal Computers as workstations will also be supported.

SOFTWARE

The TBACS token responds to a set of 17 commands, which are implemented in firmware stored in the token's non-volatile memory (See Appendix A). The sequence in which these commands are executed is controlled by a set of flags which are checked at the first step of each command. If the flags are not set correctly, the given command will not be executed and the token will return an error code.

The commands are grouped into three general classes: security officer (SO) commands, user/workstation authentication commands, and user/remote host authentication commands. The SO commands provide for the initialization of new tokens by loading host IDs, cryptographic keys, and PINs. The token is ready to be issued to the user after the SO has completed this initialization process. The remaining commands implement the authentications required by TBACS to control the login process.

The token key table contains host IDs and corresponding cryptographic keys, with enough space for 100 ID/key entries. During an authentication sequence, the token uses the host ID as an index into the key table to select the appropriate cryptographic key for this host, and then uses this key to perform cryptographic processing during subsequent steps in the authentication process.

The workstation software interacts with the user through the workstation console, and with the user's token through the reader/writer. If the user wishes to login to a remote host, the workstation software initiates the necessary network communications protocols with the remote host, and issues commands to the token as required to perform the authentication process. Cryptographic keys for all valid system users are calculated by the workstation and host using a unique master key and the user's PIN, thus eliminating the need to store a separate key for each user.

A software simulation of the token command set has been written in the C language to provide a detailed functional specification for the TBACS token. A supervisory function is responsible for decoding the input command string and executing the C function which corresponds to a specific command in the token command set. This simulation consists of approximately 2500 lines of code, and is currently being used as a model for development of the token firmware.

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 authentication. The DES algorithm, operating firmware, and critical data are stored internally on the smart token, providing a higher level of security than systems which use tokens only as data storage devices.

USER/TOKEN AUTHENTICATIONS

When a user begins the login process on a workstation, he 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 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 by simply pressing a key. If the user chooses to continue, he must next prove his identity to the token. The login manager prompts the user for his PIN/password, which is then encrypted 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 ID. 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 (Figure 4). From this point on, TBACS uses the token to represent the user's identity for the remaining authentications.

THREE-WAY HANDSHAKE

Once the previous steps have been completed, the token and the workstation must authenticate to each other. This is accomplished through a three-way handshaking protocol which allows each party to prove that it posesses the same cryptographic key as the other party, without having to physically exchange keys [6](Figure 5). 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 secret key, generates a second random number, and transmits both values to party B.
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 posesses the same secret key.

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.

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 an rlogin 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 rlogin manager connects to an rlogin server on the remote host. At this point, the local rlogin manager acts primarily as a communications path between the token and the remote rlogin 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 repeated between the token and the rlogin server on the remote host, and finally the rlogin server terminates and the standard rlogin process connects the user to the remote host.

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, and 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.

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.

DURESS PASSWORD CAPABILITIES

If a user is forced to login under duress, the user can derive a duress password from the normal password using a simple algorithm. The token is able to recognize this as a duress password, and signal the login manager in an unobtrusive manner. The token will then continue to function as usual, and the login manager will take whatever additional steps are necessary to notify security personnel and other computers on the network of the duress condition.

KEY MANAGEMENT

When a user first enrolls on a TBACS computer system, the user must contact the appropriate security officer for that computer. The security officer initializes a blank token by loading the following data:

1. The security officer's ID, encrypted under the security officer's PIN.
2. The user's ID, encrypted under an initial user PIN.
3. A token identification number.
4. The token expiration date.

The security officer next generates a DES key which is loaded onto the token. The random number generation capability of the security officer's token can be used to generate these keys. The token encrypts this key using the user's PIN and stores it in the key table along with the computer's host identification number. The host computer can generate this key from the user's PIN and the host master key as required during future login processes. As an alternative, the DES key could be stored in the computer's key database indexed by the user's identity. After receiving the token from the security officer, the user may change the token identification number and the user PIN by entering the current values.

The user may now enroll on another TBACS computer by contacting the computer's security officer, who generates another DES key which is stored on the token and the host computer as previously described. 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 order to acitvate the token during a login, the user must supply the correct user PIN. Once activated, the token can be used to authenticate the user to the user's workstation and then to other host computers by means of the three-way handshake previously described.

CONCLUSION

The TBACS project demonstrates that smart token technology can provide an effective tool for the design of access control systems. By taking advantage of the processing abilities of smart tokens, such systems can provide a higher level of security than those which rely only on passwords while placing little additional burden on the user. In addition, smart tokens can store information on terminal configuration and other user preferences which can be used to personalize the system. This added convenience can help to increase user acceptance of more stringent security requirements.

REFERENCES

1. Data Encryption Standard (DES), National Bureau of Standards (U.S.), Federal Information Processing Standards Publication 46, National Technical Information Service, Springfield, VA, April 1977.
2. 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.
3. American National Standard for Financial Institution Message Authentication (Wholesale), ANSI X9.9-1986, American Bankers Association, Washington, DC.
4. American National Standard for Financial Institution Message Authentication (Retail), ANSI X9.19-1985, American Bankers Association, Washington, DC.
5. 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.
6. 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.

APPENDIX A: TOKEN COMMAND SET

1- RESET
2- ENTER SO PIN
3- AUTHENTICATE SO
4- ENTER USER PIN
5- LOAD KEY
6- AUTHENTICATE TOKEN
7- GENERATE CHALLENGE
8- AUTHENTICATE USER
9- CHANGE TOKEN PIN
10- WORKSTATION VERIFY AND RESPOND
11- OUTPUT HOST ID TABLE
12- HOST VERIFY AND RESPOND
13- READ ZONE
14- WRITE ZONE
15- APPEND ZONE
16- CALLDES
17- TEST