Technical Note
Operating Systems

Anita K. Jones
Editor

Password Authentication with Insecure Communication

Leslie Lamport
SRI International

A method of user password authentication is described which is secure even if an intruder can read the system's data, and can tamper with or eavesdrop on the communication between the user and the system. The method assumes a secure one-way encryption function and can be implemented with a microcomputer in the user's terminal.

Key Words and Phrases: security, authentication, passwords, one-way function
CR Categories: 4.35, 4.39

I. The Problem

In remotely accessed computer systems, a user identifies himself to the system by sending a secret password. There are three ways an intruder could learn the user's secret password and then impersonate him when interacting with the system:

(1) By gaining access to the information stored inside the system, e.g., reading the system's password file.
(2) By intercepting the user's communication with the system, e.g., eavesdropping on the line connecting the user's terminal with the system, or observing the execution of the password checking program.
(3) By the user's inadvertent disclosure of his password, e.g., choosing an easily guessed password.

The third possibility cannot be prevented by any password protocol, since two individuals presenting the same password information cannot be distinguished by the system. Eliminating this possibility requires some mechanism for physically identifying the user—for example, a voice print. Such a mechanism is beyond the scope of this paper, so we restrict ourselves to the problem of removing the first two weaknesses.

This work was funded in part by the National Science Foundation under Grant No. MCS-7816783.

Author's address: Leslie Lamport, SRI International, 333 Ravenswood Avenue, Menlo Park, CA 94025

II. The Solution

The first weakness can be eliminated by using a *one-way function* to encode the password. A one-way function is a mapping F from some set of words into itself such that:

(1) Given a word x, it is easy to compute $F(x)$.
(2) Given a word y, it is not feasible to compute a word x such that $y = F(x)$.

We will not bother to specify precisely what "easy" and "feasible" mean, so our reasoning will be informal. Note that given $F(x)$, it is always possible to find x by an exhaustive search. We require that such a computation be too costly to be practical. A one-way function F can be constructed from a secure encryption algorithm: one computes $F(x)$ by encrypting a standard word using x as a key [1].

Instead of storing the user's password x, the system stores only the value $y = F(x)$. The user identifies himself by sending x to the system; the system authenticates his identity by computing $F(x)$ and checking that it equals the stored value y. Authentication is easy, since our first assumption about F is that it is easy to compute $F(x)$ from x. Anyone examining the system's permanently stored information can discover only y, and by the second assumption about F it will be infeasible for him to compute a value x such that $y = F(x)$. This is a widely used scheme, and is described in [2] and [3].

While removing the first weakness, this method does not eliminate the second—an eavesdropper can discover the password x and subsequently impersonate the user. To prevent this, one must use a sequence of passwords $x_1, x_2, \ldots, x_{1000}$, where x_i is the password by which the user identifies himself for the ith time. (Of course, the value 1000 is quite arbitrary. The assumption we will tacitly make is that 1000 is small enough so that it is "feasible" to perform 1000 "easy" computations.) The system must know the sequence $y_1, \ldots, y_{1000}$, where $y_i = F(x_i)$, and the y_i must be distinct to prevent an intruder from reusing a prior password.

There are two obvious schemes for choosing the passwords x_i.

(1) All the x_i are chosen initially, and the system maintains the entire sequence of values $y_1, \ldots, y_{1000}$ in its storage.
(2) The user sends the value y_{i+1} to the system during the ith session—after logging on with x_i.

Neither scheme is completely satisfactory: the first because both the user and the system must store 1000 pieces of information, and the second because it is not robust—communication failure or interference from an

intruder could prevent the system from learning the correct value of y_{i+1}. We present here a method that combines the best features of both schemes without these drawbacks.

Our solution is to let the ith password x_i equal $F^{1000-i}(x)$ for some fixed word x, where F^n denotes n successive applications of F. Thus, the sequence of 1000 passwords is

$$F^{999}(x), \ldots, F(F(F(x))), F(F(x)), F(x), x.$$

The sequence of y_i needed by the system to authenticate these passwords is

$$F^{1000}(x), \ldots, F(F(F(x))), F(F(x)), F(x).$$

Since it is feasible to compute F^n for $n \leq 1000$, property 2 of the one-way function implies that these y_i are distinct. For example, if $F^{987}(x) = F^{123}(x)$, then given $y' = F^{123}(x)$, one can compute $x' = F^{986}(x)$ where $y' = F(x')$.

It follows from our definition of the x_i that $y_i = x_{i-1}$ for $i > 1$. In other words, each user password is the value needed by the system to authenticate the next password. Hence, the system must initially be given the value $y_1 = F^{1000}(x)$ and need subsequently remember only the last password sent by the user.

To see that the method is secure against eavesdropping and tampering with the communication, suppose that knowing the first 987 passwords $F^{999}(x), \ldots, F^{13}(x)$ enabled an intruder to find the next password $F^{12}(x)$. Then given $y' = F^{13}(x)$, it would be feasible to compute $F^{14}(x), \ldots, F^{999}(x)$ and then compute $x' = F^{12}(x)$ where $y' = F(x')$. This would contradict property 2 of the one-way function F. Since the password sequence is determined in advance, no amount of tampering with the communication will allow an intruder to impersonate or permanently lock out the user.

Our method has an important robustness property: If the system and the user have gotten out of synchrony—the user sending x_j and the system using y_k to authenticate it, with $j \neq k$—then this can be detected by repeatedly applying F to both the password and the system's authenticating value until a match is obtained. For example, if the user is sending x_j and the system is checking with y_{j+3}, then this can be discovered because $x_j = F^2(y_{j+3})$. The system can accept the value of x_j if $j > k$, and can request a later value if $j < k$.

This robustness can be used to prevent an intruder from taking advantage of system "crashes". Restarting the system after a crash usually requires "backing it up" to a prior point—a point at which it could be expecting a password already sent by the user. For example, suppose an intruder has been routinely recording all transmissions, and the system crashes after the user has transmitted x_{374}. If the system were backed up to a point where it was still expecting a password to be checked against y_{374}, the intruder could then obtain the value x_{374} from the recording he had made of the transmissions and impersonate the user.

With our method, backing the system up after a crash does not require backing up to a password that might already have been used. The system can be "jumped forward" instead. For example, suppose that the system as a whole is backed up to a point at which it was using $y_{374} = F^{626}(x)$ to authenticate the next password. Suppose further that users are warned not to perform more than one identification per hour. If the time between the back-up point and the system crash is less than two hours, then the user should not have transmitted any password beyond x_{375}. The system can then ask for x_{376} as the user's next password, since only the user should be able to generate it. The system can authenticate the value of x_{376} knowing only y_{374} because $y_{374} = F^3(x_{376})$. Thus, an eavesdropper could not use any of the passwords that he has discovered even in the event of a system restart.

III. Implementation

We envision that our method would be implemented with the aid of a microcomputer in the user's terminal. In the future, "intelligent" terminals will probably contain logic to perform data encryption quickly, so computation of the one-way function F presents no problems.

The user would first randomly choose x. He would then employ his terminal in a special local mode that accepts the value x and computes the values $F(x), F^2(x), \ldots, F^{1000}(x)$. This latter value would be displayed on the screen and the user would deliver it to the system by some tamper-proof method—perhaps copying it and physically carrying it to the computer center.

In the simplest implementation, the user would send the system his name, and the system would respond with a value and a request that the user send his ith password x_i. He would then enter x and i into his terminal and the terminal would compute $x_i = F^{1000-i}$. Computing $F(x')$ from x' should take only a couple of milliseconds, so the computation of x_i would take at most a couple of seconds.

It is possible to avoid this computation by saving the values $x, F(x), \ldots, F^{999}(x)$ obtained during the original computation of $F^{1000}(x)$. More generally, one can reduce computation at the expense of storage by saving the values $x, F^k(x), F^{2k}(x), \ldots$ for some k. They could be saved in some removable storage device (such as a cassette tape) that is inserted into the terminal. This type of removable device might be a standard feature of future terminals, so that different users can operate the same terminal at different times, each with his own private data (such as encryption keys).

If the user communicates with several different systems, then he must apply the same method independently for each system using different values of x. With removable storage devices for the terminal, he could use a separate device for each system.

Of course, after the user has identified himself to the system 1000 times, he must choose a new value for x and repeat the whole process. However, one should not

depend upon the secrecy of a single piece of data for too long, so the user should choose a new value of x at regular intervals anyway.

Received 12/79; revised 4/80; accepted 5/81

References

1. Diffie, W., and Hellman, M.E. New directions in cryptography. *IEEE Trans. Inform. Theory IT-22* (Nov. 1976), 644–654.
2. Evans, A., Kantrowitz, W., and Weiss, E. A user authentication scheme not requiring secrecy in the computer. *Comm. ACM 17*, 8 (Aug. 1974), 437–442.
3. Wilkes, M.V. *Time-Sharing Computer Systems.* American Elsevier, New York, 1972.

Corrigendum. Programming Techniques and Data Structures.

Paul Pritchard, A sublinear additive sieve for finding prime numbers. *Comm. ACM 24*,1 (Jan. 1981), 18–23.

Page 18: In the fifth line after "**1. Introduction**" delete "then".

Page 19, Column 2: In line (2) of the definition of I_2, replace "p_i" with "p_1".

Page 21, Column 1: The definition of x_j should begin "$x_j = {}^{df} p \cdot f_j$"

Page 22, Table II: The second column should be headed $\pi(N)$, and the last column should be deleted.

Page 23, Column 2, line 2: Insert "algorithm" after "practical".

Corrigendum: Systems Modeling and Performance Evaluation

Micha Hofri, Disk scheduling: FCFS vs. SSTF revisited. *Comm. ACM 23*, 11 (Nov. 1980) 645–653.

G.J. Arnaudo from IMAG, Grenoble, France has pointed out that the second line in Eq. (3) has been scrambled in the horizontal notation and should be:

$$-\alpha ME^2(S)]/[M(1-\alpha)-1]\}/2(1-\rho). \qquad (3)$$

The numbers in the paper were generated from the correct result.

Technical Note
Operating Systems

Anita K. Jones
Editor

Authentication of Signatures Using Public Key Encryption

Kellogg S. Booth
University of Waterloo, Canada

One of Needham and Schroeder's proposed signature authentication protocols is shown to fail when there is a possibility of compromised keys: this invalidates one of the applications of their technique. A more elaborate mechanism is proposed which does not require a network clock, but does require a third party to the transaction. The latter approach is shown to be reliable in a fairly strong sense.

Key Words and Phrases: authentication, digital signatures, notary, public key encryption

CR Categories: 3.81, 4.31, 4.35

I. Compromise in Public Key Systems

Needham and Schroeder [6] described a means for authenticating signatures using public key encryption. User A sends user B a message which has been doubly encrypted, first with A's secret key and then with B's public key. Using Needham and Schroeder's notation, this process is represented by

$$A \rightarrow B: \quad \{\{text\text{-}block\}^{SKA}\}^{PKB}.$$

The receiver B can read the message by applying his secret key first and then A's public key, thus decrypting the text. B can convince an arbiter of the authenticity of the message and of A's authorship simply by allowing the arbiter to apply A's public key to the message after it has been decrypted by B's secret key. In a world of permanent and uncompromised keys this technique provides a foolproof authentication mechanism.

This paper was originally submitted in January, 1979. It languished in the editorial process until a recent change in editors. The Afterword section was added to cite several papers that have a prior publication date, but which were written contemporaneously with or after this paper.

Author's present address: Kellogg S. Booth, Department of Computer Science, University of Waterloo, Waterloo, Ontario, Canada N2L 3G1.