Improved Private Information Download Protocol

Sung-Ming Yen[1)] Ray-Lin Oyan[1)] Yi-Yuan Lee[2)]

[1)] Dept of Electrical Engineering, Tamkang University
Tamsui, Taipei Hsien, Taiwan 25137, R.O.C.
E-mail: yensm@csie.ncu.edu.tw

[2)] Communication Network Lab.
Institute for Information Industry, Taiwan, R.O.C.

Abstract

In this paper, an improved private information download protocol will be proposed. The new protocol is more efficient than the well known NetWare version 4 protocol and the recently reported Perlman-Kaufman protocols. More importantly, in order to develop a generic protocol for private information download, no existence of secure trapdoor one-way function, i.e., public-key based encryption/decryption, is assumed for the protocol construction.

1 Introduction

Key management is one of the most important issues for computer and network security. How to keep and remember the private key and even the public key of a user are essential for many practical security applications. Often, it is difficult for the users to remember their lengthy private keys without any auxiliary device, e.g., IC card. However, use of the IC card requires additional cost, accessibility of the IC card readers, and system developments. Also, it requires the users to remember bringing their IC cards.

In 1994, Novel proposed its NetWare version 4 [1] network directory service, NDS, to provide a solution to the above key management requirement. The NetWare version 4 provides the users the ability to download their private keys from the remote key server. This protocol makes it be convenient for the users to access their private keys and even public keys without any additional hardware requirement. The users only have to remember their personal password, a *weak key*, in order to retrieve their private information from any *trusted* terminal supporting NetWare network protocol.

Although password-based protocols are often the most desirable solutions to provide user convenience [2], such kind of protocols should at least be strong enough to resist the off-line *dictionary attack* (or sometimes called the *password guessing attack*) [3, 4, 5].

Due to the importance of private key download, Perlman and Kaufman in 1999 proposed a set of protocols [6] trying to improve the NetWare version 4. However, it will be pointed out (this was also mentioned in [6]) that two of the most efficient protocols in [6] are vulnerable to the undetectable on-line password guessing attack [7]. The proposed protocols in this paper however can withstand the above attack.

In fact, in addition to retrieve the personal private key (the contrary of public key), the above protocols can in general be employed to securely download any secret keys (for applications in conventional symmetric key cryptosystems, e.g., DES [8]), any private personal information, or any system/network configuration data. Therefore, in order to develop a generic private information (it does not have to be a private key of a public-key encryption system, e.g., RSA [9]) download protocol, no existence of secure *trapdoor one-way function* [10], i.e., public-key based encryption/decryption, is assumed for the protocol construction. Only *one-way function*, e.g., discrete logarithm [10], will be required for protocol development.

2 Review of previous works

2.1 The NetWare version 4 protocol

In the following, the NetWare version 4 protocol [1] will be briefly reviewed. In the protocol, S is the key server which stores each user's private key and any public key or personal private information if required. Each user, say A, has his personal password P_a which is often a weak key and is much easy to remember. The server selects for user A a random 'salt_a' and stores the fol-

lowing information $\{\mathrm{ID}_a, x_a = h(P_a, \mathrm{salt}_a), \mathrm{salt}_a, z = \{SK_a\}_{P_a}\}$ in the secure system table where $h()$ is a cryptographic hash function, e.g., SHA [11], and $\{m\}_k$ means the conventional encryption of message m using k as the encryption key. The main purpose of the inclusion of salt is to reduce the possibility of having a same x for different users if unfortunately the same password is selected.

The purpose of the NetWare protocol is to enable the user A to download his protected private key $\{SK_a\}_{P_a}$ from the network key server. The protocol goes as follows.

(1)$A \rightarrow S$: ID_a.

(2)$S \rightarrow A$: salt_a, r_s, PK_s
where r_s is a random integer selected by the server and PK_s is the public key of the server used in a modern public-key based encryption system, e.g., the RSA [9] system. Note however that PK_s can already be accessible from the trusted terminal if the number of servers is limited and all the public keys are stored in advance.

(3)$A \rightarrow S$: $E_{PK_s}(Y, r_a)$
where r_a is a random integer selected by the user, $Y = h(x_a, r_s)$, and $E_{PK_s}(m)$ means public-key encryption of message m using PK_s as the encryption key.

(4)$S \rightarrow A$: $\{z \oplus r_a\}_Y$.

Under the assumption of employing a secure public-key cryptography, a passive attacker when trying to conduct an *off-line* dictionary attack should find both P_a and r_a at the same time. Often, r_a will be a very large random integer which makes the attack infeasible.

An active intruder trying to conduct an *on-line* password guessing attack can be identified by the server at the third step and the protocol will be terminated.

Note that in the above analysis of the NetWare protocol and all the other protocols in this paper, we assume that the server system table is securely protected and the value Y and r_a (together behave as the session key for secure communication) will be deleted after the protocol.

2.2 The Perlman-Kaufman protocols

In [6], a set of private key download protocols were proposed by Perlman and Kaufman based on the EKE [2] and the SPEKE [12] protocols. The basic 4-step EKE-based protocol was first developed, then the 4-step EKE-based protocol with precomputation (to reduce the server's computational load) and the 2-step protocol with precomputation were developed. Also, similar protocols based on SPEKE were reported. In the following, only the 4-step and the 2-step EKE-based protocols with precomputation will be briefly reviewed.

2.2.1 The 4-step Perlman-Kaufman protocol

The server selects for user A a random integer r_s and stores the following information $\{\mathrm{ID}_a, r_s, \{R_s = \alpha^{r_s} \bmod p\}_{h(P_a)}, z = \{SK_a\}_{P_a}\}$ in the secure system table. The protocol goes as follows with only some simplification.

(1)$A \rightarrow S$: ID_a.

(2)$S \rightarrow A$: $\{R_s\}_{h(P_a)}$.

(3)$A \rightarrow S$: $R_a = \alpha^{r_a} \bmod p$, $h(K)$
After receiving $\{R_s\}_{h(P_a)}$, the user A decrypts the ciphertext using $h(P_a)$ as the key to recover R_s. User A then computes and sends the server $R_a = \alpha^{r_a} \bmod p$ where r_a is a random integer. At the same time, $h(K)$ is computed for the purpose of data integrity check where $K = R_s^{r_a} \bmod p$.

(4)$S \rightarrow A$: $\{z\}_K$
Based on the received R_a, the server computes its copy of K as $K = R_a^{r_s} \bmod p$. Then, the hash value of this K is computed as $h(K)$ and is compared with the one received from user A. If these two values match, then $\{z\}_K$ will be sent to user A.

In the above protocol, the server will send $\{z\}_K$ (at the step 4) to the protocol initiator only if the hash value of $K = R_a^{r_s} \bmod p$ is equal to the one received at the step 3. This prevents the possible undetectable on-line password guessing attack [7] applicable to the next 2-step Perlman-Kaufman protocol.

2.2.2 The 2-step Perlman-Kaufman protocol

In addition to the previous information, the server also stores $h(P_a)$ in the system table. The protocol goes as follows.

(1)$A \rightarrow S$: ID_a, $\{R_a = \alpha^{r_a} \bmod p\}_{h(P_a)}$
where r_a is a random integer selected by the user.

(2)$S \rightarrow A$: $\{R_s\}_{h(P_a)}$, $\{z\}_K$
After receiving $\{R_a\}_{h(P_a)}$ from the user A, the server decrypts R_a using $h(P_a)$ as the key. Then, the server encrypts z using $K = R_a^{r_s} \bmod p$ as the key and sends $\{z\}_K$ to the user A.

An active attacker can guess a possible password P_a' and sends $\{R_a = \alpha^{r_a} \bmod p\}_{h(P_a')}$ to the server. When $\{R_s\}_{h(P_a)}$ is received, the attacker tries to decrypt R_s

and raises it to r_a to obtain K. If K and P'_a can successfully be used to retrieve the user's private key, then the on-line password guessing attack successes. Also reported in [6] that in the above scenario the server cannot distinguish a legitimate private key download from an on-line password guessing attack [7]. Even the attack can more or less be prevented, however this may complicate the server's load extensively.

3 The proposed private information download protocol

Although the protocol is generic for downloading any private information, in this paper we consider the case of downloading private key as before. In the following proposed private information download protocol, user A also chooses his password P_a but the salt is eliminated and is replaced by user identity. The server stores $\{\mathrm{ID}_a, x_a = h(P_a, \mathrm{ID}_a), z = \{SK_a\}_{P_a}\}$ in the secure system table. Since the server S will be set up before the protocol application and the terminal is trusted, it is reasonable to assume that the public key of the server can be accessed beforehand. Suppose that the server's public key is computed as $PK_s = \alpha^{SK_s} \bmod p$ where p is a prime, α is a primitive root modulo p, and SK_s is the server's private key.

The following proposed protocol aims to provide the same functionality as the NetWare version 4 protocol but with fewer communication interactions and without the need of public-key based encryption system (almost all such kind of encryption systems are protected by patents). The proposed protocol is also more efficient than the 4-step EKE-based and SPEKE-based Perlman-Kaufman protocols. The protocol consists of the following two transactions.

(1)$A \to S$: ID_a, $v = x_a + PK_s^{r_a} \bmod p$, $k = h(R_a)$
Since the user A knows P_a, so he can compute $x_a = h(P_a, \mathrm{ID}_a)$. Then based on this x_a and the public key PK_s of the server, user A computes $v = x_a + PK_s^{r_a} \bmod p$ where r_a is a random integer. The user A also computes k as the hash value of R_a where $R_a = \alpha^{r_a} \bmod p$. The purpose of k is used for data integrity check of R_a which will be recovered by the server in the step (2).

(2)$S \to A$: $\{z\}_{R_a}$
The server knows its private key SK_s and the secret value x_a of the user A. After receiving v, the server computes $R_a = (v - x_a)^{SK_s^{-1}} = \alpha^{r_a} \bmod p$ where SK_s^{-1} is the multiplicative inverse of SK_s modulo $(p-1)$. If the hash value of this R_a (i.e., $h(R_a)$) matches k received from the user A, then $\{z\}_{R_a}$ (i.e., encryption of z using R_a as the key) will be sent to user A.

3.1 Analysis of the protocol

Basically, the proposed new private information download protocol is as secure as the NetWare protocol. From the viewpoint of *on-line* password guessing attack, both the two protocols provide intrusion detection mechanism. In the NetWare protocol, with a guessed password P'_a, the attacker will compute a corresponding $Y' = h(x'_a, r_s)$ where $x'_a = h(P'_a, \mathrm{salt}_a)$. After receiving and decrypting $E_{PK_s}(Y', r_a)$ in the third step of the NetWare protocol, the server can identify an attempt of password guessing. Similarly in the proposed protocol, with a guessed password P'_a, the attacker will compute a corresponding $v = x'_a + PK_s^{r_a} \bmod p$ where $x'_a = h(P'_a, \mathrm{ID}_a)$. After receiving v and $k = h(R_a) = h(\alpha^{r_a} \bmod p)$, the server will compute $R'_a = (v - x_a)^{SK_s^{-1}} \neq \alpha^{r_a} \bmod p$. Evidently, the server can detect the active password guessing attack.

A minor difference between the above two on-line attacks is that in the NetWare protocol a replay of previous messages is detectable since it employs a challenge-response approach. So, the server will stop the fourth step. On the other hand, in the proposed protocol, a replay attempt cannot be detected and the second step of the protocol will be continued. However, under such situation what the attacker received is the replay of $\{z\}_{R_a}$ from the server. This brings no useful information to the attacker.

For the off-line passive attack, the attacker should find exactly both P_a and r_a and checks the correctness via the intercepted messages to break both the NetWare and the proposed protocol. The other possibility is to find exactly both P_a and SK_s to break both the systems. For the proposed protocol in this paper, to derive $SK_s = \log_\alpha PK_s \bmod p$ from PK_s, the attacker is enforced to solve the discrete logarithm problem which is well known as an one-way function [10] and is therefore infeasible.

All the above analyses justify the claim that the proposed private information download protocol is (at least) as secure as the NetWare version 4 protocol. Also, only one-way function but not trapdoor one-way function will be required. Note that, generally speaking, an one-way function is more generic and can almost all the time be more securely designed than a trapdoor one-way function.

3.2 Performance comparisons

Comparisons among numerous previous protocols and the proposed one on the communication interaction number, number of conventional encryption/decryption operations, and the number of modular exponentiation computations are summarized in the Table 1.

Table 1: Performance comparisons

	# of step	conventional encry/decry	exponentiation user/server
NetWare	4	1/2	1/1
4-step EKE	4	1/3	2/1
2-step EKE	2	2/4	2/1
4-step SPEKE	4	1/2	2/1
2-step SPEKE	2	1/2	2/1
proposed	2	1/2	2/1

Recall that the NetWare protocol needs special public-key based encryption and decryption and the 2-step EKE-based and 2-step SPEKE-based protocols are vulnerable to the undetectable attack.

4 A function enhanced private information download protocol

For some environments, even a protocol can detect the on-line password guessing attack and can employ some mechanisms to limit or to slow down guessing attempts. The users may also wish to be informed with some statistical data of possible attack and conduct suitable countermeasures if required.

The following modified protocol aims to provide the users the number of times of possible on-line password guessing attack recognized by the server.

(1)$A \rightarrow S$: ID_a, $v = x_a + PK_s^{r_a} \bmod p$, $k = h(R_a)$.

(2)$S \rightarrow A$: $\{z, T, N, r_s\}_{R_a}$
where T is the time of the last legitimate private information retrieval, N is the number of tries of on-line password guessing attack recognized by the server since the last legitimate retrieval, and r_s is a random integer selected by the server using as a challenge.

(3)$A \rightarrow S$: r_s.
The user A decrypts the received ciphertext and replies r_s as the response.

The above challenge-response approach is employed during the steps 2 and 3, using r_s as the random challenge. The number N will be incremented each time if the correct response r_s cannot be received by the server and N will be reset to zero if r_s can be received.

5 Conclusions

Computer and network security will become more and more important. Data stored and transmitted need protection. Resource access from a remote machine needs identity authentication. Most of the above security mechanisms need the involution of user's secret key, private key, system/network configuration data or many other security parameters which are often too lengthy for the user to remember. A simple solution is to store and download the required lengthy private information with guaranteed security. The NetWare version 4 protocol and the recent Perlman-Kaufman protocols are some possible solutions for this scenario. In this paper, an improved alternative is proposed which is more simpler with fewer cryptographic assumption for implementation and is more efficient for communication costs.

6 Acknowledgments

This work was partly supported by the National Science Council of the Republic of China under contract NSC89-2213-E-008-049 and was partly sponsored by MOEA and supported by Institute for Information Industry, R.O.C.

References

[1] R. Lee and J. Israel, "Understanding the role of identification and authentication in NetWare 4," Novel Application Notes, October 1994.

[2] S. Bellovin and M. Merritt, "Encrypted key exchange: Password-based protocols secure against dictionary attacks," *Proc. of the IEEE Symp. on Research in Security and Privacy*, pp.72–84, 1992.

[3] T. Lomas, L. Gong, J. Saltzer, and R. Needham, "Reducing risks from poorly chosen keys," *ACM Operating Systems Review*, Vol.23, No.5, pp.14–18, 1989.

[4] L. Gong, T. Lomas, R. Needham, and J. Saltzer, "Protecting poorly chosen secrets from guessing attacks," *IEEE Journal on Selected Areas in Communications*, Vol.11, No.5, pp.648–656, 1993.

[5] G. Tsudik and E. Van Herreweghen, "Some remarks on protecting weak keys and poorly-chosen secrets from guessing attacks," *1993 IEEE Symp. on Reliable Distributed Systems*, pp.136–142, 1993.

[6] R. Perlman and C. Kaufman, "Secure passwordbased protocol for downloading a private key," *Proc. of the 1999 Network and Distributed System Security Symp., the NDSS'99*, February 1999.

[7] Y. Ding and P. Horster, "Undetectable on-line password guessing attacks," Technical Report TR-95-13, Dept. of Computer Science, University of Technology, Chemnitz-Zwickau, Germany, 1995.

[8] NBS FIPS PUB 46, "Data Encryption Standard," National Bureau of Standard, U.S. Department of Commerce, Jan. 1977.

[9] R.L. Rivest, A. Shamir, and L.M. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Commun. ACM*, Vol.21, No.2, pp.120–126, 1978.

[10] A.J. Menezes, P.C. van Oorschot, and S.A. Vanstone, *Handbook of applied cryptography*, CRC Press, 1997.

[11] FIPS 180-1, "Secure Hash Standard," NIST, US Department of Commerce, Washington D.C., April 1995.

[12] D. Jablon, "Strong password-only authenticated key exchange," *Computer Communication Review, ACM SIGCOMM*, Vol.26, No.5, pp.5–26, October 1996.