An Improvement of VeriSign's Key Roaming Service Protocol

Jeeyeon Kim[1], Hyunjo Kwon[1], Haeryong Park[1], Seungjoo Kim[1], and Dongho Won[2]

[1] KISA (Korea Information Security Agency),
78, Garak-Dong, Songpa-Gu, Seoul 138-803, Korea
{jykim, hckwon, hrpark, skim}@kisa.or.kr
[2] Sungkyunkwan University,
300 Chunchun-Dong, Suwon, Kyunggi-Do, 440-746, Korea
dhwon@dosan.skku.ac.kr
http://dosan.skku.ac.kr

Abstract. In the past two or three years, most major Public Key Infrastructure(PKI) vendors have released products which allow users to roam from one machine to another without having to manually manage the export and import of their credentials such as private keys and corresponding certificates onto temporary media like diskettes. In this paper, we survey three popular key roaming products of Baltimore's, Entrust's and VeriSign's. We also propose key roaming system which improves VeriSign's roaming service and analyze its security.

1 Introduction

In PKI systems, user's credentials are often stored in the client system's hard disk. Credentials may consist of public/private key pairs, X.509 public key certificates and/or other private user data. These PKI systems are vulnerable to various attacks where the private key may be stolen or substituted, usually without user's even being aware of it. Furthermore the use of hard disk does not satisfy the needs of the roaming user, who accesses the Internet from different client terminals.

So far, there are two basic approaches to provide a secure roaming service.

– portable hardware key storage such as smartcards
– password-only mechanisms

While smartcards are commonly understood to be the best medium in which to store, carry and utilize the credentials, it is not currently practical because of the low level of penetration of smartcard readers into the general PC environment. Given the cost and availability problems of hardware storage devices today, more sophisticated approach is to use the password-only mechanisms. In this approach, a roaming users store their credentials at a central server and download temporary copies when needed to their local machine.

© Springer-Verlag Berlin Heidelberg 2003

In the past two or three years most major PKI vendors have used this password-only methods to release products which allow users to roam from one machine to another without having to manually manage the export and import of their credentials onto temporary media like diskettes.

Generally, key roaming systems consist of the following components.

- User : a roaming user or client.
- CA(Certification Authority) : a server to issue and manage certificates for encryption or authentication.
- RS(Roaming Server) : a server to provide a roaming user with an information needed to obtain user's credential.
- CS(Credential Server) : a server to store and manage a roaming user's credentials.

Note that RS and CS can be operated on the same server.

Generally, key roaming systems are vulnerable to exhaustive password guessing attack at server. Ford and Kaliski presented the key roaming protocol which overcomes this deficiency[8] and is used to VeriSign's roaming service. Ford and Kaliski's methods use multiple servers to further prevent guessing attacks by an enemy that compromises all but one server. However, their method which relies on techniques like server-authenticated Secure Sockets Layer(SSL) which is known to be vulnerable to web-server spoofing attacks[4],[7].

In this paper, we survey three popular key roaming products of Baltimore's, Entrust's and VeriSign's. We propose key roaming protocol which does not need a prior sever-authenticated channel such as SSL and analyze its security.

2 Existing Commercial Key Roaming Systems

In this section, we briefly survey the existing commercial key roaming systems of Baltimore's[1], Entrust's[6],[16] and VeriSign's[3], [8], [14], [15].

[Notations]
We use the following common notations throughout our paper.

- Alice : honest user.
- A : Alice's roaming client software.
- $ID(A)$: an identity of Alice.
- PWD : a password memorized by Alice.
- PRI and PUB : private and public key pair of Alice.
- E_X : a symmetric or public encryption with cryptographic key X.
- H : a cryptographic hash function.
- p : a prime, $p = 2q + 1$, where q is a large prime.

2.1 Baltimore's UniCert Roaming Service

[User Registration Process]

1. A $\rightarrow$ CA : A sends the request for registration to CA.
2. CA : CA generates A's password, PWD, key pair (PRI, PUB) and certificate, $Cert(A)$, and CA stores $Cert(A)$ in the repository.
3. CA $\rightarrow$ CS : CA forms A's $.p12$ file, such as $.p12 = (E_{PWD}(PRI), Cert(A))$ and sends it to CS, called "UniCERT Roaming Server Administrator and UniCERT Roaming Server" in [1].
4. CS : CS generates A's roaming authentication key, RAK, such as $RAK = H(PWD)$ and roaming encryption key, REK. CS computes $E_{REK}(.p12)$ and $E_{PUB(RS)}(RAK, REK)$, where $PUB(RS)$ is a public key of RS, called "UniCERT Roaming Protection Encryption Key Server" in [1]. And then CS stores $ID(A)$, $E_{REK}(.p12)$ and $E_{PUB(RS)}(RAK, REK)$ in the repository.
5. CA $\rightarrow$ A : CA securely sends PWD to A.

[Roaming Process]

1. A $\rightarrow$ CS : A prompts Alice to enter her ID and password. When Alice has done this, A sends a roaming request to the CS.
2. CS : After receiving and verifying the request, CS generates a protected access request number, ARN. Then CS retrieves A's $E_{PUB(RS)}(RAK, REK)$ and $E_{REK}(.p12)$ from the repository and gets RS's certificate, $Cert(RS)$ and Internet address.
3. CS $\rightarrow$ A : CS sends $E_{PUB(RS)}(RAK, REK)$, $E_{REK}(.p12)$ and ARN to A with RS's URL and certificate, $Cert(RS)$.
4. A $\rightarrow$ RS : A computes $E_{PUB(RS)}(OSK)$, where OSK is an one-time local session key, and sends it to RS with $E_{PUB(RS)}(RAK, REK)$, $E_{REK}(.p12)$, ARN and $H(PWD)$.
5. RS $\rightarrow$ A : RS checks ARN to its database of recent request number's received. If ARN exists within the database, the request is rejected. Otherwise RS decrypts $E_{PUB(RS)}(RAK, REK)$ using its private key corresponding to $PUB(RS)$, verifies $H(PWD)$ using RAK, and decrypts $E_{REK}(.p12)$ using REK to obtain $.p12$. Then RS computes $E_{OSK}(.p12)$ and sends it to A.
6. A : A decrypts $E_{OSK}(.p12)$ using OSK to get $.p12$ file.

[Security analysis]

If $H(PWD)$ is sent to CS without the security protocol such as SSL, Baltimore Unicert Roaming can be vulnerable to password guessing attack. Furthermore, CS's or RS's operator or attacker who compromises RS to obtain RS's private key can always mount an exhaustive search attack on a user's password.

2.2 Entrust's Authority Roaming Service

[Registration Process]

When creating/recovering a roaming user, Entrust/Session can be used to secure communications between RS/CS, called Entrust/Profile Server in [6], and A.

1. A $\rightarrow$ CA : A requests CA to register and issue its credential, $E_{PWD}(PRI)$ and $Cert(A)$.
2. CA $\rightarrow$ A : CA issues and sends the credential to A.
3. A $\rightarrow$ RS/CS : A calculates $H(PWD)$ and sends its credential and $H(PWD)$ to RS/CS.
4. RS/CS : RS/CS generates a symmetric key, K and encrypts A's credential with K. Then RS/CS computes $E_{RSK}(H(PWD), K)$, where RSK is a key of RS/CS, and stores it in the repository with $E_K(E_{PWD}(PRI), Cert(A))$.

[Roaming Process]

On roaming login, prior to decrypting the credential, SPEKE(Simple Password Exponential Key Exchange)[10] replaces Entrust/ Session as the mechanism used to secure communications between RS/CS and A.

1. Alice : Alice enter $ID(A)$ and PWD to login.
2. A : A retrieves $E_K(E_{PWD}(PRI), Cert(A))$ and $E_{RSK}(H(PWD), K)$ from the repository. Then A calculate $H(PWD)^x \bmod p$, where x is a random number.
3. A $\rightarrow$ RS/CS : A sends $E_{RSK}(H(PWD), K)$ to RS/CS with $ID(A)$ and $H(PWD)^x \bmod p$.
4. RS/CS : RS/CS decrypts $E_{RSK}(H(PWD), K)$ to get $H(PWD)$ and K, and then computes a session key $S1 = H(PWD)^{xy} \bmod p$, where y is a random number.
5. RS/CS $\rightarrow$ A : RS/CS computes $E_{S1}(K)$ and $H(PWD)^y \bmod p$, and sends them to A.
6. A : A also computes $S1 = H(PWD)^{xy} \bmod p$ and decrypts $E_{S1}(K)$ using $S1$ and $E_K(E_{PWD}(PRI), Cert(A))$ to obtain its credential.

[Security analysis]

Entrust's Authority Roaming system uses SPEKE[10] to provide strong password authentication for mobile users. Like Baltimore's Roaming, it also has the same problem that RS/CS's operator or attacker who compromises RS/CS to obtain a private key of RS/CS can always mount an exhaustive search attack on a user's password. Furthermore partition attack and subgroup confinement attack on SPEKE were discussed in [10] and [11].

2.3 VeriSign's Roaming Service

[Notation]

– f : a function that maps passwords to elements of multiplicative order q in Z_p^*
– KDF : a function that computes K by combining K_i
– OWF : an one way hash function

[User Registration Process]

VeriSign's Roaming Service should preferably be implemented in such a way that the integrity of the exchange is protected. For instance, the user might protect the exchange with SSL and trust the server to perform the computation correctly.

1. A : A, called "Personal Trust Agent(PTA)" in [14], generates Alice's key pair, (PRI, PUB).
2. A $\rightarrow$ CA : A sends the request of issuing a certificate and roaming registration to CA.
3. CA $\rightarrow$ A : CA issues and sends $Cert(A)$ to A.
4. Alice : Alice enters $ID(A)$ and PWD to A.
5. A $\rightarrow$ $\mathrm{RS_i}$: A generates a random number a for $1 \leq a \leq q-1$, and computes $w = f(PWD)$ and $r = w^a \bmod p$. Then A sends $ID(A)$ and r to $\mathrm{RS_i}$ for $i = 1, 2$.
6. $\mathrm{RS_i}$ $\rightarrow$ A : $\mathrm{RS_i}$ $(i = 1, 2)$ selects a secret exponent d_i between 1 and $q - 1$ for A, computes $s_i = r^{d_i} \bmod p$, and returns it to A. Then $\mathrm{RS_i}$ stores $ID(A)$ and d_i $(i = 1, 2)$ in its database.
7. A : A computes $K_i = s_i^{1/a} \bmod p$ $(i = 1, 2)$ and $K = KDF(K_1, K_2)$ [1], where $1/a$ is the inverse of a mod q. Then A computes $EPD = E_K(PRI, Cert(A))$.
8. A $\rightarrow$ $\mathrm{RS_i}$: A computes and sends $v_i = OWF(K, ID(\mathrm{RS_i}))$ for $i = 1, 2$ to $\mathrm{RS_i}$, where $ID(\mathrm{RS_i})$ is the identity of $\mathrm{RS_i}$. $\mathrm{RS_i}$ stores $(v_i,\ ID(A))$ in its database.
9. A $\rightarrow$ $\mathrm{RS_2}$: A securely sends EPD to $RS2$ through secure communications channel such as SSL, and destroys K. $\mathrm{RS_2}$ stores EPD in its DB.

[Roaming Process]

1. Alice : Alice enters $ID(A)$ and PWD to it's client software, A.
2. A $\rightarrow$ $\mathrm{RS_i}$: A computes $w = f(PWD)$ and $r = w^a \bmod p$, where a is a random number, and sends $ID(A)$ and r to $\mathrm{RS_i}$ for $i = 1, 2$.
3. $\mathrm{RS_i}$ $\rightarrow$ A : $\mathrm{RS_i}$ $(i = 1, 2)$ retrieves d_i corresponding to the received $ID(A)$, computes $s_i = r^{d_i} \bmod p$, and returns it to A.
4. A : A computes $K_i = s_i^{1/a}$, where $1/a$ is the inverse of a mod q, and regenerate $K = KDF(K_1, K_2)$.
5. A $\rightarrow$ $\mathrm{RS_i}$: A computes $v_i' = OWF(K, ID(\mathrm{RS_i}))$ for $i = 1, 2$ and sends it to $\mathrm{RS_i}$. To authenticate Alice, $\mathrm{RS_i}$ $(i = 1, 2)$ compares v_i' with v_i stored in its database.
6. A $\leftrightarrow$ $\mathrm{RS_2}$: If successfully authenticated, A requests EPD to $\mathrm{RS_2}$. $\mathrm{RS_2}$ retrieves EPD and securely returns through secure communication channel, such as SSL.
7. A : A decrypts EPD with K to gain its credential, PRI and $Cert(A)$. After using its credential, A destroys K and its credential.

[1] For combining secret components K_i, t-out-of-N threshold secret sharing methods may be applied [3].

[Security analysis]

VeriSign's roaming service with SSL can trick the user into using "valid" SSL connections to malicious RSs. In this case, malicious RSs use bogus secret data, d_i^* for $i = 1, 2$ to mount a password guessing attack on user's password. Each malicious RS, $\mathrm{RS_i^*}$, computes $s_i^* = r^{d_i^*} \bmod p$ for $i = 1, 2$ and returns it to A in the step 3 of the roaming process. Receiving s_i^*, A computes $K^* = KDF(K_1^*, K_2^*)$ and $v_i'^* = OWF(K^*, ID(\mathrm{RS_i}))$ and returns $v_i'^*$ in the step 5 of the roaming process. Malicious RSs checks whether $v_i'^*$ received is equal to $OWF(KDF(f(PWD')^{d_1^*}, f(PWD')^{d_2^*}), ID(\mathrm{RS_i}))$, where PWD' is a candidate of PWD. This match indicates a correct guess of the password. Therefore, VeriSign's roaming service is vulnerable to the password guessing attack in server spoofing attacks.

3 Our Proposed System

We propose a modified system to solve the problem mentioned in VeriSign's roaming service.

[Notations]

- g : a generator of G_q, where G_q is the unique subgroup of Z_p^* of order q
- x_i : a private key of $\mathrm{RS_i}$ for $i = 1, \cdots, n$.
- y : a group pubic key of RSs. It is generated as follows[5],[12],[13][2] : **(1)** Each $\mathrm{RS_i}$ $(i = 1, \cdots, n)$ chooses $r_i \in_R Z_q$ and makes $y_i = g^{r_i} \bmod p$ public. **(2)** Each $\mathrm{RS_i}$ selects a random polynomial $f_i \in_R Z_q[x]$ of degree $t-1$ such that $f_i(0) = r_i$. Let $f_i(x) = r_i + a_{i,1}x + a_{i,2}x^2 + \cdots + a_{i,t-1}x^{t-1} \bmod q$, where $a_{i,1}, a_{i,2}, \cdots, a_{i,t-1} \in_R Z_q$. $\mathrm{RS_i}$ computes $f_i(j) \bmod q$ $(\forall j \neq i, 1 \leq j \leq n)$ and sends it to $\mathrm{RS_j}$ securely. And each $\mathrm{RS_i}$ computes $g^{a_{i,1}} \bmod p, g^{a_{i,2}} \bmod p, \cdots, g^{a_{i,t-1}} \bmod p$ and makes them public. **(3)** Using received $f_j(i)$ $(\forall j \neq i, 1 \leq j \leq n)$, each $\mathrm{RS_i}$ verifies that $g^{f_j(i)} \overset{?}{=} y_j \cdot (g^{a_{j,1}})^{i^1} \cdot \cdots \cdot (g^{a_{j,t-1}})^{i^{t-1}} \bmod p$. **(4)** Define H as the set $\{\mathrm{RS_i} | \mathrm{RS_i}$ is an honest RS satisfying the step (3)$\}$. Each $\mathrm{RS_i}$ computes its private key $x_i = \sum_{j \in H} f_j(i)$ and keeps it secure. **(5)** RSs compute and publish their group public key $y = \prod_{j \in H} y_j$.

[User Registration Process]

- A : A generates PWD and random number, z. Then A computes w and S, such as $w = f(PWD)$ and $S = g^z w \bmod p$, where f is a function that maps passwords to elements of multiplicative order q in Z_p^*.
- $\mathrm{A} \rightarrow \mathrm{RS_i}$: A sends $(ID(A), S)$ to $\mathrm{RS_i}$ for $i = 1, \cdots, n$.
- $\mathrm{RS_i} \rightarrow \mathrm{A}$: $\mathrm{RS_i}$ $(i = 1, \cdots, n)$ computes $B_i = S^{x_i} \bmod p$ and sends it to A. Then $\mathrm{KS_i}$ stores $ID(A)$ in its repository.

[2] In [9], Gennaro et al. proved that one of the requirements in [12] is not guaranteed : more precisely, the property that the key is uniformly distributed in the key space. So far several solutions have been discovered, but for the practicality, we use Pedersen's scheme.

- A $\rightarrow$ CS : A computes $K = KDF((\prod_{1 \le i \le t} B_i^{\prod_{1 \le i \le t, j \ne i} j/(j-i)})/y^z \bmod p)$ and $K_1 = OWF(K, 1)$. Then A sends $(ID(A), K_1, E_{K_1}(PRI))$ to CS.
- CS : CS stores $(ID(A), K_1, E_{K_1}(PRI))$ in its database.

[Roaming Process]

- A : A generates a random number, z. Then A computes $w = f(PWD)$ and $S = g^z w \bmod p$ using PWD and z.
- A $\rightarrow$ RS$_\mathrm{i}$: A sends $(ID(A), S)$ to RS$_\mathrm{i}$ for $i = 1, \cdots, n$.
- RS$_\mathrm{i}$ $\rightarrow$ A : RS$_\mathrm{i}$ $(i = 1, \cdots, n)$ checks where A is the registered user. If $ID(A)$ exists within the repository, RS$_\mathrm{i}$ $(i = 1, \cdots, n)$ computes $B_i = S^{x_i} \bmod p$ and then sends it to A.
- A $\rightarrow$ CS : A computes $K = KDF((\prod_{1 \le i \le t} B_i^{\prod_{1 \le i \le t, j \ne i} j/(j-i)})/y^z \bmod p)$ and $K_1 = OWF(K, 1)$. Then A sends a roaming request to CS.
- CS $\rightarrow$ A: CS generates c $(1 \le c \le q - 1)$ and sends it to A.
- A $\rightarrow$ CS : A computes $r = OWF(K_1, c)$ and sends r to CS.
- CS $\rightarrow$ A : CS retrieves K_1 from its database, computes $OWF(K_1, c)$, and then checks whether it is equal to r. If it is equal to r, CS sends $E_{K_1}(PRI)$ to A.
- A : A gets PRI by decrypting $E_{K_1}(PRI)$ with K_1.

[Security analysis]

Since our system is designed to store only registered user's identity, attacker accessing to RSs' repository cannot obtain any information about user's password. Unlike VeriSign's roaming service, Alice's key, K, is generated using RSs' private keys and attacker without RSs' private keys cannot masquerade RSs in the roaming process. Therefore, our system is secure against server's spoofing attack without additional security protocol such as SSL. While VeriSign's roaming service requires (n+1) times of exponent calculation, our proposed system requires only two exponent calculations and $(n + 1)$ multiplications, where n is the number of RSs. Therefore it can efficiently compute K even if the number of RSs is increased.

4 Conclusion

We survey three popular key roaming products of Baltimore's, Entrust's and VeriSign's. We also propose key roaming system which retrieves user's credential from multiple related RS and CS, without exposing the password to off-line guessing unless all servers are compromised, and without relying on additional secure channels in roaming process.

References

1. Baltimore: Baltimore Roaming. Private Communications (2001)
2. Bellovin S., Merrit M.: Encrypted key exchange: password based protocols secure against dictionary attacks. In Proceedings of the Symposium on Security and Privacy (1992) 72–84
3. Burton S. , Kaliski JR.: Server-Assisted Regeneration of a Strong Secret From a Weak Secret. US Patent, US Patent Number 09/804,460 (2000)
4. Cohen F.: 50 Ways to Attack Your World Wide Web System. Computer Security Institute Annual Conference, Washington DC (1995)
5. Desmedt Y., Frankel Y.: Threshold cryptosystems. Advanced in Cryptology – Crypto'89, Springer-Verlag, LNCS 435 (1990) 307–315
6. Entrust : The Entrust Roaming Solution. Private Communications (2000)
7. Felton E., Balfanz D., Dean D., Wallach D.: Web Spoofing : An Internet Con Game. 20th National Information Systems Security Conference, Balimore Maryland (1997) available at http://www.cs.princeton.edu/sip/pub/spoofing.html
8. Ford W., Burton S., Kaliski JR.: Server-Assisted Generation of a Strong Secret from a Password. Proceedings of the IEEE 9th International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, NIST, Gaithersburg MD (2000)
9. Gennaro R., Jarecki S., Krawczyk H., Rabin T.: Secure distributed key generation for discrete-log cryptosystems. Advanced in Cryptology – Eurocrypt'99, Springer-Verlag, LNCS 1592 (1999) 295–310
10. Jablon D.: Strong password-only authenticated exchange. ACM Computer Communications Review, vol 26, No.5 (1996)
11. Oorschot P., Wiener M.J.: On Diffie-Hellman Key Agreement with Short Exponents. EUROCRYPT'96, Spriger-Verlag, LNCS 1070 (1996)
12. Pedersen T.P. : A threshold cryptosystem without a trusted party. Advanced in Cryptology – Eurocrypt'91, Springer-Verlag, LNCS 547 (1991) 522–526
13. Pedersen T.P. : Distributed provers with applications to undeniable signatures. Advanced in Cryptology – Eurocrypt'91, Springer-Verlag, LNCS 547 (1991) 221–238
14. VeriSign Inc.: Roaming Service Administrator's Guide. (2002) available at http://www.verisign.com
15. VeriSign Inc.: Roaming Service. (2002) available at http://www.verisign.com/products/roaming/
16. Wiener M. J.: Secure Roaming with Software Tokens. PKI TWG Meeting (2000) available at http://csrc.nist.gov/pki/twg/y2000/presentations/twg-00-32.pdf