Scalable Storage Scheme from Forward Key Rotation

Chunbo Ma[1,2], Jun Ao[3], and Jianhua Li[1]
[1] School of Information Security Engineering
Shanghai Jiao Tong University, Shanghai, 200030, P. R. China
machunbo@sjtu.edu.cn
[2] The State Key Laboratory of Information Security,
Beijing, 100049, P. R. China
[3] State Key Laboratory for Radar Signal Processing,
Xidian University, Xi'an, Shanxi, 710071, P. R. China
Junjunao1@263.net

Abstract. Kallahalla et al. presented a RSA-based Forward Key Rotation mechanism in secure storage scheme PLUTUS to ensure that the key used for encrypting updated files is related to the keys for all files in the file group. The encryption scheme based on Forward Key Rotation is such a scheme that only the authorized person is allowed access to the designated files and the previous versions. In this paper, we present a Forward Key Rotation storage scheme based on discrete logarithm and discuss its security. Moreover, we propose another improved Forward Key storage scheme from pairing on elliptic curves. Compared to the scheme presented by Kallahalla et al., our scheme uses relatively short keys to provide equivalent security. In addition, the re-generated keys can be verified to ensure that the keys are valid in the improved scheme.

Keywords. Secure Storage, Public Key, Backward Key Rotation, Files, Scalability

1. Introduction

As more information becomes available in digital format, enormous quantities of data are created. With the increasing requirement to both temporarily and permanently retain information, the secure storage technique is attracting much attention. In a data storage system, the storage medium for preserving the information is an important target to a malicious attacker. Once the attacker breaks the system, he can gain unauthorized information, disclose some valuable secrets or prevent the access of legitimate users. In order to avoid these risks, the researchers have proposed many potential systems for securing stored information, such as NASD [2][3], PASIS [4][5], CFS [6], SNAD [7][8] and PLUTUS [1], and so on. Moreover, all these researches come into being a field of computer research. Currently, the major target of a secure storage system is ensuring information confidentiality, integrity and availability without substantially degrading performance.

Kallahalla et al. [1] present a secure storage scheme, PLUTUS, in 2003. The primary goal of the scheme is to provide file owners with direct control over authorizing access to their files as well as scalable key management. Key revocation is a major problem in secure storage system. Due to the proliferation of keys and the use of file groups, the key revocation is very complicated. Since several files in the file system are encrypted with the same key, key revocation will result in mass re-encryption. However, Kallahalla et al designed a key rotation scheme to alleviate the negative effects. The key rotation scheme can make the updated secret key relate to the previous versions. For the user of PLUTUS, when he is allowed access to a certain re-encrypted files using given file-group key, then he can generate previous versions from the given key. In other words, any valid user can re-generate the matching key for a given file if he has the latest file-group key. Therefore, we call the key rotate scheme designed by Kallahalla Forward Key Rotate (FKR) scheme.

The security of the FKR presented by Kallahalla et al. is based on the RSA. However, limited by the prime generation technique, producing a pair of suitable keys used in RSA scheme is not an easy thing. Moreover, until now there is no way to show the strength of the RSA is equivalent to that of decomposing a large number. Currently, RSA based schemes use relatively long keys compared to the security they provide. In this aspect, a scheme based on elliptic curves provides much shorter keys. As the requirement of practice, it is necessary to design other KFR storage scheme related to different intractable problems.

In this paper, we first present a FKR storage scheme based on discrete logarithm and discuss its security. Subsequently, we present an improved FKR storage scheme from pairings on elliptic curves. Compared to the mechanism presented by Kallahalla et al., the improved scheme provides relatively short keys to perform the same function. Moreover, the user can verify the validity of the re-generated keys via bilinear pairings.

The paper is organized as follows. In section2, we will review Kallahalla et al.'s KFR scheme. Some complexity assumptions are presented in section 3. We propose our FKR storage scheme based on discrete logarithm in section 4. In section 5, we discuss the security of the proposed FKR storage scheme. Subsequently, we present an improved KFR storage scheme in section 6. Finally, we draw the conclusions in section 7.

2. Kallahalla et al.'s Forward key rotation scheme

Kallahalla et al. [1] proposed a Forward Key Rotation scheme used in PLUTUS system in 2003. In this section, we will briefly review their FKR scheme. Suppose that there exists a secure RSA encrypt scheme as defined in [13]. Let U_0 be a user who will establish a FKR scheme, e_0 be the public key of U_0 and d_0 be the matching private key.

— User U_0 chooses $k_0 \in Z_q^*$ uniformly at random, and computes

$$k_1 = k_0^{d_0}, k_2 = k_1^{d_0}, \cdots, k_l = k_{l-1}^{d_0}.$$

— When user $U_{i\neq 0}$ wants to access the FKR scheme, the user U_0 gives $U_{i\neq 0}$ the latest key $k_l = k_{l-1}^{d_0}$. Hence, $U_{i\neq 0}$ has ability to compute $k_{l-1} = k_l^{e_0} = k_{l-1}^{d_0 e_0}$ and get any $k_{z\leq l}$.

The security of the scheme is based on RSA. Kallahalla et al. simplify the key management of PLUTUS using this FKR scheme.

3. Background

3.1 Complexity Assumptions

We assume that a prime p is chosen at random such that $p-1$ has a large prime factor q. Let g be an element of order q. Define the following problems.

— **Computation Diffie-Hellman [CDH]**: Given g^a, g^b for unknowns $a, b \in Z_q^*$, compute $g^{ab} \in Z_p^*$.

— **Decision Diffie-Hellman [DDH]**: Given g^a, g^b, g^c for unknowns $a, b, c \in Z_q^*$, decide whether $g^{ab} \overset{?}{=} g^c$.

— **K-Exponent Assumption [k-E]** [9]: Given $\{g, g^x, g^{x^2}, \cdots, g^{x^k}\}$ for unknown x, compute $g^{x^{k+1}} \in Z_p^*$.

3.2 Bilinear Maps

Let G_1 be a cyclic multiplicative group generated by g, whose order is a prime q and G_2 be a cyclic multiplicative group of the same order q. Assume that the discrete logarithm in both G_1 and G_2 is intractable. A bilinear pairing is a map $e: G_1 \times G_1 \rightarrow G_2$ and satisfies the following properties:

1. Bilinear: $e(g^a, p^b) = e(g, p)^{ab}$. For all g, $p \in G_1$ and $a, b \in Z_q$, the equation holds.
2. Non-degenerate: There exists $p \in G_1$, if $e(g, p) = 1$, then $g = O$.
3. Computable: For $g, p \in G_1$, there is an efficient algorithm to compute $e(g, p)$.

Typically, the map e will be derived either from the modified Weil pairing [10][11] or the Tate pairing [12] on an elliptic curve over a finite field. Pairings and other parameters should be selected in proactive for efficiency and security. In group G_1, the **CDH** and **K-E** assumptions defined in 3.1 are intractable. However, the **DDH** assumption is tractable.

3.3 General Scheme

A FKR storage scheme consists of four algorithms.

1. **Initialize.** Given the security parameter l, the algorithm outputs the system parameters.

2. **Key Generation (i)**. Input the number i, output i-th secret key K_i used to encrypt i-th files.
3. **Encrypt (K_{i+1}, K_i, $File_i$)**. Input two secret keys (K_{i+1}, K_i) and a file $File_i$. The algorithm encrypts $File_i$ using (K_{i+1}, K_i), and outputs the corresponding ciphertext EF_i.
4. **Decrypt (EF_i, K_{i+1}, K_i)**. Input the ciphertext EF_i and the secret keys (K_{i+1}, K_i). The algorithm decrypts the ciphertext using the secret key (K_{i+1}, K_i) and outputs the corresponding plaintext $File_i$.

3.4 Security Notions

We define adaptively chosen ciphertext security of a FKR storage scheme. Security is defined using the following game between an *Attacker* and *Challenger*.

1. **Setup.** The *Challenger* initializes the system. The *Challenger* gives the *Attacker* the resulting system parameters.
2. **Query phase 1.** The *Attacker* adaptively issues encrypt queries $q_1.q_2,\cdots,q_m$ and decryption queries $q_1.q_2,\cdots,q_n$, respectively. The *Challenger* simulates Encrypt and Decrypt respectively, and responds with matching answers.
3. **Challenge.** Once the *Attacker* decides that **Query phase 1** is over it outputs two equal length files $(File_0, File_1)$ to the *Challenger*. The *Challenger* picks a random bit $\lambda \in \{0,1\}$, and encrypts the message $File_\lambda$. It gives ciphertext C to the *Attacker.*
4. **Query phase 2.** The *Attacker* continues to adaptively issue encryption and decryption queries. The *Challenger* responds as in the phase 1. These queries may be asked adaptively as in **Query phase 1**. Note that the decrypt query $q_j = C$ is not permitted, where $0 \le j \le n$.
5. **Guess.** Finally, the *Attacker* outputs a guess $\lambda' \in \{0,1\}$ for λ and wins the game if $\lambda' = \lambda$.

The storage scheme is secure against chosen ciphertext attack, if the *Attacker* has a negligible advantage $\varepsilon = \left|\Pr(\lambda = \lambda') - \frac{1}{2}\right|$ to win the game.

4. Forward Key Rotation Storage Scheme

In this section, we will design a FKR storage scheme related to discrete logarithm. We suppose that Server who has a series of files $\{File_1, File_2, \cdots, File_n\}$ will establish the FKR storage scheme and a user Alice asks to access it. Let p be a large prime and $\langle g \rangle$ be a cyclic multiplicative group generated by g, whose order is a prime q such that $q \mid (p-1)$. There exists a pair of security algorithms (E, D), where the algorithm E is a secure encryption algorithm based on discrete logarithm, and D is the matching decryption algorithm.

Step1. Server chooses a random number $r \in Z_q^*$ and computes $K_i = g^{r^i} \bmod p$ as a secret key.

Step2. Server generates the encrypted files $EF_i = E_{K_{i+1}}(K_i \| File_i)$, where $a \| b$ denotes the concatenate of a and b. Thereafter, Server publishes all encrypted files.

When Alice asks to access to the FKR storage scheme at the point j, Server sends the secret key K_{j+1} to Alice. Since Alice can obtain any encrypted files, she can compute

$$K_j \| File_j = D_{K_{j+1}}(EF_j)$$

and get K_j and $File_j$. Similarly, Alice can obtain K_{j-1} and $File_{j-1}$ via K_j and EF_{j-1}.

However, by the K-E assumption defined in section 3.1, Alice can't produce the secret key K_{j+2} using K_j and K_{j+1}. In other words, Alice doesn't have ability to decrypt EF_{j+1} even though she has secret key $K_{i \le j+1}$.

5. Security

The security of the scheme is partly based on the algorithms (E, D). Thereby, we assume that these two algorithms are secure. Given $\{g^r, g^{r^2}, \cdots, g^{r^i}\}$, one can't deduce $g^{r^{i+1}}$ by the **K-E** assumption mentioned in section 3.1. It means that if Server gives an access point at j, one can't access to the files $File_k$, where $k > j$.

Then we will give the following theorem to show the security of the proposed storage scheme.

Theorem. We assume that an attacker Eve who can, with success probability ε, break the FKR storage scheme within a time τ by asking **Encrypt** and **Decrypt** oracles at most q_E and q_D queries respectively, then there exists a challenger Alice who running in a time $\tau^{'}$ can solve the **DDH** problem with success probability $\varepsilon^{'}$, where

$$\varepsilon^{'} = \varepsilon, \qquad \tau^{'} = \tau + (q_E + q_D + 1)t_{ED},$$

where t_{ED} is the time for performing an encryption or decryption algorithm.

Proof. We assume that Eve is an attacker who has the ability to break the storage scheme. Then there exists a challenger Alice who can solve the **DDH** problem by running Eve as a subroutine. The system chooses a random bit $b \in \{0,1\}$ and a random number $r \in Z_q^*$. The challenger Alice is given $\{g^r, g^{r^2}, \cdots g^{r^{i-1}}, T, g^{r^{i+1}}, \cdots, g^{r^k}\}$. If $b = 1$, the system sets $T = g^{r^i}$, otherwise, chooses a random number $T \in Z_p^*$.

The attacker Eve is allowed to issue **Encrypt**, **Decrypt** and **Challenge** queries. The challenger Alice will simulate the corresponding oracles to output the answers.

Query Phase 1. The attacker Eve is allowed to issue following queries.

Encrypt queries. Eve chooses a random number $j \in [1, k]$, and takes any file $File_j$, and then issues query on $(j, File_j)$.

— If $j \neq i$, Alice computes $EF_j = E_{K_{j+1}}(K_j \| File_j)$, and outputs EF_j as the answer, where $K_{j+1} = g^{r^{j+1}}$ and $K_j = g^{r^j}$.

— If $j = i$, Alice outputs error messages and halts.

Decrypt queries. Eve chooses a random number $j \in [1, k]$, and takes any ciphertext EF_j, and then issues query on (j, EF_j).

— If $j \neq i$, Alice computes $K_j \| File_j = D_{K_{j+1}}(EF_j)$, and outputs K_j and $File_j$ as the answer, where $K_{j+1} = g^{r^{j+1}}$ and $K_j = g^{r^j}$.

— If $j = i$, Alice outputs error message and halts.

Since above simulation is perfect, the attacker Eve can't distinguish the simulated result from the actual results. The above queries can be asked several times. When Eve decides this phase is over, he issues challenge query.

Challenge query. Eve outputs two equal length files $File_0$ and $File_1$ to Alice. Upon receiving the two files, Alice chooses a random bit $\lambda \in \{0,1\}$, and computes

$$EF_i = E_{K_{i+1}}(T \| File_\lambda)$$

where $K_{i+1} = g^{r^{i+1}}$. Thereafter, Alice sends EF_i to Eve as the answer.

Note that the **Challenge query** is allowed only once.

Query Phase 2. The attacker Eve continues to adaptively issue encryption and decryption queries. The challenger Alice will respond as in the phase 1. However, decryption query $q_j = EF_i$ is not permitted. We assume that Eve issues at most q_E encrypt queries and q_D decrypt queries.

Guess. After receiving the answer from Alice, Eve outputs his guess $\lambda^{'}$. If $\lambda^{'} = \lambda$, Alice decides $b = 1$, otherwise $b = 0$.

Since Eve has ability to break the scheme with non-negligible probability ε, i.e. outputs $\lambda^{'} = \lambda$

with probability ε, then the challenger Alice can solve **DDH** with the same probability.

□

6. Improved scalable FKR storage scheme

In section 4, we have presented a FKR storage scheme based on discrete logarithm. However, in practice some devices only have limited capability, so we should design a scheme which can provide shorter keys. Thereby, an improved FKR storage scheme from pairing on elliptic curve is presented in this section. In this scheme the re-generated keys can be verified via bilinear pairings. Thus also provide a measure for a user to verify the validity of the plaintext.

Let G_1 and G_2 be two groups that support a bilinear map as defined in section 3.2. There exists a pair of security algorithms (E, D), where the algorithm E is a secure encryption algorithm based on elliptic curves, and D is the matching decryption algorithm.

Step1. Server chooses a random number $r \in Z_q^*$ and computes $K_i = g^{r^i}$ as a secret key, where $g^{r^i} \in G_1$.

Step2. Server generates the encrypted files $EF_i = E_{K_{i+1}}(K_i \| File_i)$. Thereafter, Server publishes all encrypted files.

When Alice asks to access to the FKR storage scheme at the point j, Server sends the secret key K_{j+1} to Alice. Since Alice can obtain any encrypted files, she can compute

$$K_j \| File_j = D_{K_{j+1}}(EF_j)$$

and get K_j and $File_j$. Similarly, Alice can obtain K_{j-1} and $File_{j-1}$ via K_j and EF_{j-1}.

Step3. After computing K_j and K_{j-1}, Alice performs the following step to verify the validity of the re-generated keys.

$$e(K_{j+1}, K_{j-1}) \overset{?}{=} e(K_j, K_j).$$

If above equation is true, the re-generated keys are valid. Otherwise, Alice outputs error messages.

7. Conclusions

Secure storage is a crucial problem in the Internet. Motivated by Kallahalla's forward key rotation and the requirement of the short key schemes, we present two FKR storage scheme. One is based on discrete logarithm, and another is from bilinear pairing on elliptic curve. The latter storage scheme is suitable for implementation in many scenarios, especially those where the storage capability of the users is limited.

References

1. Mahesh Kallahalla, Erik Riedel, Ram Swaminathan, Qian Wang, and Kevin Fu. PLUTUS: SCALAble secure file sharing on untrusted storage. In Conference on File and Storage Technology (FAST'03) pp. 29-42.
2. Howard Gobioff, David Nagle, Garth Gibson. Embedded Security for Network-Attached Storage. CMU SCS technical report CMU-CS-99-154, June 1999.
3. Howard Gobioff, Garth Gibson, Doug Tygar. Security for Network Attached Storage Devices. CMU SCS technical report CMU-CS-97-185 1997.
4. Gregory R. Ganger, Pradeep K. Khosla, Mehmet Bakkaloglu, Michael W. Bigrigg, Garth R. Goodson, Semih Oguz, Vijay Pandurangan, Craig A. N. Soules, John D. Strunk, Jay J. Wylie. Survivable Storage Systems. DARPA Information Survivability Conference and Exposition, pp. 184-195 Vol 2. IEEE 2001.
5. Jay J. Wylie, Michael W. Bigrigg, John D. Strunk, Gregory R. Ganger, Han Kiliccote, and Pradeep K. Khosla. Survivable Information Storage Systems. IEEE Computer, 33(8): 61-68, August 2000.
6. Matt Blaze, A Cryptographic File System for Unix. First ACM Conference on Communications and Computing Security, Fairfax, VA November, 1993.
7. William Freeman and Ethan Miller. Design for a Decentralized Security System for Network-attached Storage. In Proceedings of the 17th IEEE Symposium on Mass Storage Systems and Technologies, pp. 361-373, College Park, MD, March 2000.

8. Ethan L. Miller, Darrell D. E. Long, William Freeman, and Benjamin Reed. Strong Security for Distributed File Systems. In Proceedings of the 20th IEEE International Performance, Computing and Communications Conference, pp. 34-40, Phoenix, April 2001. IEEE.
9. F. Zhang, R. Safavi-Naini, W. Susilo. An Efficient Signature Scheme from Bilinear Pairings and Its Applications. Practice and Theory in Public Key Cryptography-PKC 2004, Berlin: Springer-Verlag, 2004: 277-290.
10. Boneh D, Lynn B, and Shacham H. Short signatures from the Weil pairing. Advances in Cryptology -- Asiacrypt'2001, Gold Coast, Australia, Lecture Notes in Computer Science, 2248, Springer-Verlag (2001) 514-532.
11. Dan Boneh, and Matthew K. Franklin. Identity-based encryption from the Weil pairing. SIAM J. Comput., 32(3): 586-615, 2003.
12. Paulo S. L. M. Barreto, Hae Yong Kim, Ben Lynn, and Michael Scott. Efficient algorithms for pairing-based cryptosystems. In Crypto'02, pages 354-368, London, UK, 2002. Springer-Verlag.
13. R. L. Rivest, A. Shamir and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. CACM, Vol. 21, N.2, Feb. 1978, pp. 120-126.