Dynamic Multi-Secret Sharing Scheme

Han-Yu Lin* and Yi-Shiung Yeh
Department of Computer Science
National Chiao Tung University
Hsinchu, 300, Taiwan, Republic of China
* e-mail: hanyu.cs94g@nctu.edu.tw

Abstract

Secret sharing schemes are very important techniques for the key management. To provide more efficient and flexible alternatives for the applications of secret sharing, this paper presents a dynamic multi-secret sharing scheme. A significant characteristic of the proposed scheme is that each participant has to keep only one master secret share which can be used to reconstruct different group secrets according to the number of threshold values. By applying successive one-way hash functions and the exclusive OR (XOR) operation, the proposed scheme is secure against the notorious conspiracy attack even though the pseudo secret shares are compromised. Further, when one of the group secrets is updated with a new one, each participant's master secret share is still unchanged, i.e., these master secret shares are truly multi-use instead of one-time-use.

Keywords: dynamic, secret sharing, key management, Lagrange Interpolation, threshold

1. Introduction

Since Diffie and Hellman [2] introduced the first public key system based on the intractability of solving the discrete logarithm problem (DLP) [2, 9] in 1976, public key cryptosystems [3, 12] had been widely used in various applications such as e-cash [1], e-voting [11] and e-market [8], etc. One important issue of the public key system is the key management. Consider the operations of enterprise organizations in reality that lots of confidential documents are usually kept in a safe deposit. The management of the secret key to the safe deposit thus becomes crucial. Someone who is responsible for keeping the secret key has to be a trusted one. Yet, the case that the secret key is lost still might occur. A better alternative would be reducing the risk of key loss by distributing the responsibility of key management among many persons, i.e., the secret key is divided into several pieces which are then sent to different individuals. If sufficient participants are willing to cooperate with each other, they can reconstruct the original secret key by offering their key shares.

In 1979, Shamir [13] put the idea into practice and proposed a $(t, n)$ threshold
secret sharing scheme in which the master secret was divided into \( n \) secret shares and were delivered to different users. Any \( t \) or more of the \( n \) users could cooperatively reconstruct the master secret while less than or equal to \( t - 1 \) could not. However, Shamir’s scheme could not share multiple secrets and once the master secret was updated with a new one, the system had to reissue renewed secret shares to each user, which was considered to be system resources consuming and impracticable. To eliminate the weaknesses, in 1994, He and Dawson [5] proposed a multistage secret sharing scheme based on the one-way function. By applying successive one-way hash functions, the He-Dawson scheme realized the notion of multi-secret sharing. Yet, in 2007, Geng et al. [4] pointed out that the He-Dawson scheme was actually the one-time-use scheme [6] and further proposed a new multi-secret sharing scheme with multi-policy. Preserving the merit of Geng et al.’s scheme that different group secrets are reconstructed according to the number of threshold values, this paper presents a dynamic multi-secret sharing scheme which also outperforms Geng et al.’s scheme in terms of the computation complexity.

2. Dynamic Multi-Secret Sharing Scheme

This section introduces the dynamic multi-secret sharing scheme. The proposed scheme can be divided into three stages: the system initialization, the pseudo secret share generation, and the group secret reconstruction stages.

The system initialization stage:

Let SA be the system authority (SA) who is responsible for initializing the following public parameters:

\[ p \quad : \text{a large prime}; \]
\[ g \quad : \text{a primitive element over GF}(p); \]
\[ h(\cdot) \quad : \text{a secure one-way hash function which accepts input of any length and generates a fixed length output}; \]
\[ ID_j \quad : \text{the identifier with respect to the user } U_j, \text{ for } j = 1, 2, \ldots, n; \]

The pseudo secret share generation stage:

Assume the SA wants to share \( k \) group secrets \( s_i \), for \( i = 1, 2, \ldots, k \), among \( n \) users. The SA can perform the following steps to generate pseudo secret shares and distribute master secret shares.

Step 1 Choose distinct \( x_j \in R Z_p^* \), for \( j = 1, 2, \ldots, n \), as the master secret shares;

Step 2 Construct a polynomial \( f_i(x) \) of degree \( (i - 1) \), for \( i = 1, 2, \ldots, k \), as
\[
f_i(x) = s_i + d_1x + \ldots + d_{i-1}x^{i-1} \quad \text{where } f_i(0) = s_i; \tag{1}
\]

Step 3 For \( i = 1, 2, \ldots, k \) and \( j = 1, 2, \ldots, n \), compute
\[
V_{ij} = f_i(ID_j), \tag{2}
\]
Dynamic multi-secret sharing scheme

\[ c_{ij} = h'(x_j) \oplus x_j, \quad (3) \]
\[ R_{ij} = V_{ij} - c_{ij} \mod p; \quad (4) \]

Here, \( c_{ij} \)'s are the pseudo secret shares, the symbol ‘\( \oplus \)’ denotes the exclusive OR (XOR) operation and \( h'(x_j) \) denotes \( i \) successive applications of \( h \) to \( x_j \).

**Step 4** Deliver the master secret share \( x_j \), for \( j = 1, 2, \ldots, n \), to each user \( U_j \) via a secure channel and publish all \( R_{ij} \)'s;

**The group secret reconstruction stage:**

To reconstruct the \( l \)th group secret, say \( s_l \), at least \( l \) participants or more of \( n \) users must cooperatively perform the following steps with the group secret combiner:

**Step 1** Each \( U_j \), for \( j = 1, 2, \ldots, l \), computes his pseudo secret share as
\[ c_{lj} = h'(x_j) \oplus x_j, \quad (5) \]
and then sends it to the group secret combiner;

**Step 2** Upon receiving all \( c_{lj} \)'s, for \( j = 1, 2, \ldots, l \), the group secret combiner reconstructs the \( l \)th group secret as
\[ s_l = \sum_{j=1}^{l} (c_{lj} + R_{lj}) \prod_{r=1, r \neq j}^{l} \frac{-ID_r}{ID_j - ID_r} \mod p. \quad (6) \]

The correctness of Eq. (6) can be assured as the proof of Theorem 1.

**Theorem 1.** After receiving at least \( l \) pseudo secret shares, the group secret combiner can reconstruct the \( l \)th group secret, \( s_l \), by Eq. (6).

**Proof:** From the right-hand side of Eq. (6), we have
\[ \sum_{j=1}^{l} (c_{lj} + R_{lj}) \prod_{r=1, r \neq j}^{l} \frac{-ID_r}{ID_j - ID_r} \]
\[ = \sum_{j=1}^{l} (c_{lj} + V_{lj} - (h'(x_j) \oplus x_j)) \prod_{r=1, r \neq j}^{l} \frac{-ID_r}{ID_j - ID_r} \quad \text{(by Eqs. (3) and (4))} \]
\[ = \sum_{j=1}^{l} ((h'(x_j) \oplus x_j) + V_{lj} - (h'(x_j) \oplus x_j)) \prod_{r=1, r \neq j}^{l} \frac{-ID_r}{ID_j - ID_r} \quad \text{(by Eq. (3))} \]
\[ = \sum_{j=1}^{l} (V_{lj}) \prod_{r=1, r \neq j}^{l} \frac{-ID_r}{ID_j - ID_r} \]
\[ = \sum_{j=1}^{l} f_l(ID_j) \prod_{r=1, r \neq j}^{l} \frac{-ID_r}{ID_j - ID_r} \quad \text{(by Eq. (2))} \]
\( s_i \equiv (\text{mod } p) \) (by Lagrange Interpolation [14])

which equals to the left-hand side of Eq. (6).

Q.E.D.


In this section, we will discuss some security considerations of the proposed scheme followed by the performance evaluation.

3.1 Security Considerations

The security assumptions of our proposed scheme are the one-way hash function (OHF) [2, 9] and the XOR operation. The definitions of OHF are briefly restated below: Let \( h \) be an OHF. It is computationally infeasible to derive \( m \) from \( h(m) \). In addition, finding a pair \((m, m')\) which satisfies \( h(m) = h(m') \) is also infeasible. In the following, we analyze some security considerations of the proposed scheme.

(1). The confidentiality of pseudo secret shares: To derive a user’s pseudo secret share \( c_{ij} \) from Eq. (4), an attacker may first obtain the corresponding public information \( R_{ij} \). However, the pseudo secret share \( c_{ij} \) is protected by the integer \( V_{ij} \) which can only be computed from the secret polynomial \( f_i(ID_j) \). Hence, the attacker will fail to make it. On the contrary, if the attacker attempts to directly derive \( c_{ij} \) from Eq. (3), he has to know the master secret share \( x_j \) first, which is also computationally infeasible.

(2). The confidentiality of master secret shares: The master secret share \( x_j \) is randomly chosen by the SA and then delivered to each user \( U_j \) via a secure channel. Even though the pseudo secret share \( c_{ij} \) is compromised, any malicious adversary can not successfully derive \( x_j \) from Eq. (3) under the protection of the OHF and the XOR operation.

(3). The confidentiality of group secrets: Consider the notorious conspiracy attack that \( l - 1 \) malicious insiders cooperatively attempt to reconstruct the \( l \)th group secret \( s_l \). Unfortunately, according to the analysis of the confidentiality of pseudo secret shares, they can not obtain sufficient valid \( c_{lj} \)’s to reconstruct the group secret \( s_l \) by Eq. (6). It is also computationally infeasible to compute \( c_{lj} \) from \( c_{l-1,j} \) under the protection of the XOR operation.

From the above discussions, it can be seen that our proposed scheme is secure based on the hardness of XOR and OHF assumptions.

3.2 Performance Evaluation

In the subsection, we compare the proposed scheme (LY for short) with
Geng et al.’s scheme (GFH for short) [4] in terms of the computation complexity. For facilitating the comparisons, we first define the following notations:

\[ T_h: \text{ the time for performing a one-way hash function} \]
\[ T_m: \text{ the time for performing a modular multiplication computation} \]
\[ T_e: \text{ the time for performing a modular exponentiation computation} \]

The time for performing the modular addition and the exclusive OR (XOR) operation is ignored because they are negligible as compared to the others. Mitchell et al. [10] also stated that a hash function would not take longer time than that of a modular multiplication computation. Consequently, we can use one \( T_m \) to approximate one \( T_h \) without affecting the correctness in the evaluation. For ease of comparisons, the above various computations can be utilized into the same unit of modular multiplication computation [7, 10]. The detailed comparisons are listed as Table 1. In parentheses, the rough estimation in terms of \( T_m \) is given. As the result shown in Table 1, it can be seen that the proposed scheme outperforms Geng et al.’s scheme for the entire protocol.

<table>
<thead>
<tr>
<th>Stages</th>
<th>Scheme</th>
<th>Time complexity</th>
<th>Rough Estimation</th>
</tr>
</thead>
<tbody>
<tr>
<td>System initialization</td>
<td>GFH</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>LY</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pseudo secret share</td>
<td>GFH</td>
<td>( k(2n + 1)T_e + kT_h )</td>
<td>( k(481n + 240)T_m )</td>
</tr>
<tr>
<td>generation</td>
<td>LY</td>
<td>( kT_h )</td>
<td>( kT_m )</td>
</tr>
<tr>
<td>Group secret</td>
<td>GFH</td>
<td>( 2kT_e + (k^2 - k)T_m + k^2T_h )</td>
<td>( (2k^2 + 479k)T_m )</td>
</tr>
<tr>
<td>reconstruction</td>
<td>LY</td>
<td>( (k^2 - 2k)T_m + k^2T_h )</td>
<td>( (2k^2 - 2k)T_m )</td>
</tr>
</tbody>
</table>

Remark: \( T_h \approx T_m, T_i \approx 3T_m, \) and \( T_e \approx 240T_m. \)

4. Conclusions

In this paper, we have proposed a dynamic multi-secret sharing scheme based on the one-way hash function. The major characteristics of its design are multi-use of the master secret shares and that different group secrets can be reconstructed according to the number of threshold values, which provides more flexibility. By applying successive one-way hash functions and the XOR operation, the proposed scheme is secure against notorious conspiracy attacks even though the pseudo secret shares are compromised. Besides, our scheme also outperforms recently proposed Geng et al.’s scheme in terms of the computation complexity.
References


Received: July 29, 2007