Practical Construction for Multicast Re-keying Schemes using Reed-Solomon Codes

Chun-yan Bai, Roberta Houston and Gui-liang Feng *

Abstract

Multicast Re-keying means the establishment of a new session key for the new subgroup in the multicast system. Two practical construction methods for multicast re-keying scheme using Reed-Solomon codes are presented in this paper with examples to show the detailed constructions. The constructions require no computational assumptions and the storage complexity for GC and each user and the transmission complexity for the schemes have been reduced to $O(\log n)$ at the same time.

Index Terms: Multicast, Re-keying, Reed-Solomon code, Hash function, KDP.

1 Introduction

With the rapid development of networks, the need for high bandwidth, very dynamic and secure group(multicast) communications is increasingly evident in a wide variety of commercial, government, and Internet communities such as video-on-demand, multi-party teleconferencing, stock quote distribution and updating software. Specifically, the security in the multicast communication is the necessity for multiple users who share the same security attributes and communication requirements to securely communicate with each other using a common group session key.

The general goal of secure group communication is to dynamically transmit a message encrypted by the new session key over a broadcast channel shared by an exponential number $n = 2^m$ of users so that all but some specified small coalition of $k$ excluded users can decipher the message, even if these excluded users collude with each other in an arbitrary manner. This is what we call the broadcast exclusion problem(also known as the blacklisting problem). The establishment of a new session key for the new subgroup is called the Re-keying of the system.

In the multicast communication system, the group is dynamic, which means that at different time, different subgroups of the initial group is authorized to receive the multicast message because of those dynamically joining and leaving group members. So the secure communication in multicast environment is much more challenging than traditional point-to-point communication an raises numerous new security problems. Examples are

*chun-yan Bai, Roberta Houston and Gui-liang Feng are with the Center for Advanced Computer Studies, University of Louisiana at Lafayette, Lafayette, LA 70504, USA. Email: cxb7146@cacs.louisiana.edu, rah1231@cacs.louisiana.edu and glf@cacs.louisiana.edu.
the forward secrecy and backward secrecy guarantee. A protocol provides \textit{perfect backward secrecy} if a member joining the group at time \( t \) does not gain any information about the content of messages communicated at times \( t' < t \). A protocol provides \textit{perfect forward secrecy} if a member leaving the group at time \( t \) does not gain any information about the content of messages communicated at time \( t' > t \).

Member-joining is easy to handle by just encrypting the new session key with the old session key which is decryptable by all old members and sending the new session key individually to each new member encrypted by their own secret keys. So we just focus on the member-leaving case and assume that there is a group controller (GC) who knows all the system keys in this paper.

The initial study on the secure multicast communication can be traced back to the early 90’s [1]. And a lot of works had followed [2,3,4,5,6,7,8,9]. All in all, the work can be divided into two major groups, one of which [2,3,4,5] uses the concept of key tree structure to set up the new session key based on the Diffie-Hellman key agreement. In [2], Wallner proposed a scheme which requires only \( O(n) = O(n + (n - 1)) \) keys for GC, \( O(\log^2 n) = O(d + 1) \) keys for each user and have at most \( O(\log^3 n) = O(kd - 1) \) transmissions overhead per single eviction. The requirement is further improved in [3,4,5] which greatly reduces the transmission and storage complexity of re-keying schemes. Another stronger property of the tree structured scheme is that it allows the number of excluded users \( k \) to be arbitrary, rather than fixed in advance. But some balanced tree structure based schemes have the disadvantage of not providing collusion prevention.

The other group makes use of the broadcast encryption idea proposed by Fiat and Naor[6]. The broadcast encryption scheme enables the GC to communicate data secretly to dynamically changing authorized users while preventing any coalition of users to learn anything about the data. Other studies on broadcast encryption schemes can be found in [7,8] and [9]. In [7], the concept of the Perfect Hash Family (PHF) is reviewed and proved to be useful for the secure new session key distribution. The possibility of using the error correcting code to construct such a scheme is given there without providing any practical and detailed construction. Hartono et.al. [8] borrows the idea of Key Distribution Pattern (KDP), based on which the broadcast encryption scheme that can remove up to \( t \) users from a group of \( n \) users and is secure against collusion of \( t \) malicious users can be set up. How to use the error correcting codes to construct such a KDP is not discussed.

In [9], Poovendran and Baras show that by assigning probabilities to member revocations, the optimality, correctness and the system requirements of some of the schemes in [6,7,8] can be systematically studied using information theoretic concepts and also show that the optimal average number of keys per member in a secure multicast scheme is related to the entropy of the member revocation event, thus provides a way for us to inspect each scheme from the theory point of view.

2 Related Work review

Assume that there is a set of users \( U \), a group controller GC and a set \( K \) of Key Encrypting Keys (KEK) that is generated and stored by the GC. \textit{Session keys} are used for group member communication. A user \( u_i \) will have a subset of Key Encrypting Keys, \( K(u_i) \subseteq K \). KEKs are used to update the SK in the event of membership change due to any of the
following reasons: (a) a new member admission, (b) expiration of the SK, (c) member compromise, (d) voluntary leave, and (e) member revocation. We only consider the last case, member revocation in this paper.

The secure group communication requires KEKS to securely distribute the updated SK. If every member has an individual public key, for a group consisting of $n$ members, the SK update will involve $O(n)$ encryptions by the GC. The linear increase of the required number of encryptions in group size is not suitable for very large scale applications common in Internet, due to the amount of computational burden on the GC.

Next, we will review two scalable re-keying schemes which can reduce the number of encryptions.

2.1 Re-keying scheme based on PHF

A re-keying scheme called OR scheme in [7] specifies an algorithm by which the GC produces a common session key $k^{U \setminus W}$ for the group $U \setminus W$ without letting those users in $W$ to know the new session key, where $W \subseteq U$. The scheme we use here is as follows:

1. Key initialization: The GC generates and stores a set $K$ of KEKS and securely gives $u_i$ the set of his KEKS $K(u_i) \subseteq K$.

2. Broadcast: To remove a set of users $W$ from $U$, the GC randomly chooses a session key $k^{U \setminus W}$ and encrypts it with those keys not belonging to $W$, then broadcasts the encrypted messages to all the users. That is, the GC broadcasts

$$\{E_k(k^{U \setminus W})| k \in K, k \notin K(W), K(W) = \cup_{j \in W} K(u_j)\}.$$ 

3. Decryption: Each user $u_i \in U \setminus W$ uses one of his own EKES $k \in K(u_i)$ to decrypt $E_k(k^{U \setminus W})$ and obtain the new session key $k^{U \setminus W}$.

We review the concept of PHF here for the completeness of this paper. Let $n$ and $m$ be integers such that $2 \leq m \leq n$, $A = \{1, 2, \ldots, n\}$ and $B = \{1, 2, \ldots, m\}$ be two sets. A hash function is a function $h$ from $A$ to $B$ $h : A \rightarrow B$. We say a hash function $h : A \rightarrow B$ is perfect on a subset $X \subseteq A$ if $h$ is injective when restricted on $X$. Let $w$ be an integer such that $2 \leq w \leq m$, and let $H \subseteq \{h : A \rightarrow B\}$. $H$ is called an $(n,m,w)$ perfect hash family (PHF) if for any $X \subseteq A$ with $|X| = w$ there exists at least one element $h \in H$ such that $h$ is perfect on $X$.

It is proven in [7] that if there exists a PHF $(N,n,m,w)$, then there exists a re-keying scheme in which the number of KEKS for each user and the GC are $N$ and $Nm$ respectively and the number of broadcast transmissions to remove up to $w$ users is less than $(m-1)N$.

It is also proven in [7] that an $(N,n,d,m)$ code gives rise to a PHF $(N,n,m,w)$ as long as $N > \binom{w}{2}(N-d)$, thus can be used for the construction of the above re-keying scheme. Such a scheme can prevent $w$ users from colluding. The performance of the re-keying scheme based on PHF is determined by the parameter $N$ when $w$ and $m$ are fixed, which should be minimized to reduce the storage and transmission complexity. In But the author didn’t mention any details on which kind of error correcting code should be used and how it is used for the construction.
2.2 Re-keying scheme based on KDP

In [8], H. Kurnio reviewed the concept of Key distribution Patterns (KDP).

Let $X = \{x_1, x_2, ..., x_n\}$ and $B = \{B_1, B_2, ..., B_N\}$ be a family of subsets of $X$. The pair $(X, B)$ is called an $(n, N, t)$-key distribution pattern if

$$|(B_i \cap B_j) \cup_{k=1}^{t} B_{i_k}| \geq 1$$

for any $(t + 2)$-subset $\{i, j, s_1, ..., s_t\}$ of $\{1, 2, ..., N\}$.

With the idea of KDP, the author presented a theorem to show the existence of a multicast re-keying scheme with dynamic controller based on KDP. But how to effectively construct KDP is still an open problem.

Inspired by the work from [7] and [8], we look at the problem of multicast re-keying from the error-correcting codes point of view in this paper. In order to achieve constructions with feasible storage that do not require computational assumptions, we make an improvement on the constraints that must be satisfied to construct the broadcast encryption scheme in [7,8] by avoiding the requirement of being PHF and KDP. Based on the OR model mentioned above and assumed a system with GC, we give two practical construction of schemes based on Reed-Solomon codes and avoid any computational assumptions. Conditions underlining the constructions are also given together with examples to show the detail constructions.

Kumar et.al. [10] also consider the blacklisting problem through error-correcting codes, but their method is quite different from ours.

3 Multicast Re-keying Scheme based on R-S code

3.1 Background on code

Let $GF(q)$ be a finite field.

**Definition 3.1** (Linear Code) An $[n, k, d]$ linear code is a $k$-dimensional subspace $V_{n,k}$ of $n$-dimensional linear space $V_n$ over $GF_q$, where the minimum Hamming distance between any pair of elements is $d$.

Reed-Solomon code is an important kind of linear block BCH codes which had been widely used in such areas as space communication systems, spread-spectrum communication systems and computer storage systems.

**Definition 3.2** (Reed – Solomon Code) Let $x_1, ..., x_n \in GF(q)$ be distinct and $k > 0$. The $(n, k)$, $q$ Reed-Solomon code is given by the subspace $\{(f(x_1), ..., f(x_n))|f \in GF_{q,k}\}$, where $GF_{q,k}$ denote the set of polynomials on $GF(q)$ of degree less than $k$.

R-S code is a Maximum Distance Separable (MDS) code, which means that the error-correcting capability of the R-S code can reach the Singleton bound. The R-S code has the property that the $(n, k)_q$ R-S code is an $[n, k, n – k + 1]_q$ linear code and it requires that $n \leq q$. 

4
3.2 R-S code construction

3.2.1 First construction

Theorem 3.1 Let \((N, k, d)\) be a Reed-Solomon code over \(GF(q)\), where \(N\) is the length of the codewords, \(k\) is the length of the information bits and \(d\) is the minimum distance of the code. The number of the codewords \(n = q^k\). Let \(W\) be a subset of \(\{1, 2, ..., n\}\) with \(|W| = w\). Then such an error-correcting code can be used to construct a multicast encryption scheme as long as it satisfies that

\[ N > w \ast (N - d). \]

Proof: Let \(T\) be the set of codewords of a \((N, k, d)\) code, \(|T| = n\). We write each element of \(T\) as \((c_{i1}, c_{i2}, ..., c_{iN})\) with \(c_{ij} \in \{1, 2, ..., q\}\), where \(1 \leq i \leq n, 1 \leq j \leq N\) and \(n\) is the number of codewords. For each \(j\) we define a function \(h_j\) from \(A = \{1, ..., n\}\) to \(B = \{1, ..., q\}\) by \(h_j(i) = c_{ij}\) and let \(H = \{h_j \mid j = 1, ..., N\}\).

In the key initialization phase, the GC generates and stores a set of \(Nq\) keys defined as \(K = \{k_{h, b} \mid h \in H, b \in B\}\). For a user \(u_i, 1 \leq i \leq n\), GC secretly gives \(u_i\) the set of \(N\) Key Encryption Keys \(K(u_i) = \{k_{h, h(i)} \mid h \in H\}\).

In the broadcast stage of removing a set of users \(W\) from \(U, |W| \leq w\), the GC randomly select a new session key and encrypt it with those KEKs that do not belong to \(W\), then broadcast the encrypted messages to all the users. So those users that have been removed can not use their own KEKs to decrypt and obtain the new session key.

As to the decryption phase, we need to prove that any user \(u_i\) that does not belong to \(W\) has at least one key to decrypt and obtain the new session key.

Let \(W = \{u_{i1}, ..., u_{iw}\}\). Since the minimum distance of the code is \(d\), for any given pair of elements \(x_1, x_2 \in U\), there are at most \(N - d\) functions from \(H\) such that the values of these \(N - d\) functions evaluated on \(x_1\) and \(x_2\) are the same. For any user \(u_i \not\in W\), it has at most \(N - d\) functions that is the same as \(u_{i1}\), at most \(N - d\) same functions as \(u_{i2}\), ... and at most \(N - d\) same functions as \(u_{iw}\). The worst case is that the same \(N - d\) functions that \(u_i\) has with \(u_{i1}\) is different from those \(N - d\) functions that \(u_i\) has with \(u_{i2}\), which is different from those \(N - d\) functions that \(u_i\) has with \(u_{i3}\), ... That is, all the \(w\) \((N-d)\) functions are different. So we conclude that if \(N > w \ast (N - d)\), then \(u_i\) has at least one function that is different from all those functions belong to \(W\). That is, there exists a function \(h_{\alpha} \in H\) such that \(\{h_{\alpha}(j) \mid j = i_1, i_2, ..., i_w, i\}\) are all distinct. It follows that \(k_{[h_{\alpha}(j)]}\) is in \(K(u_i) \subseteq K(U \setminus W)\), so \(u_i\) can decrypt the encrypted message and obtain the new session key \(k_{U \setminus W}\).

The theorem holds for any set \(L\) of members who wants to leave the original group as long as \(|L| \leq w\).

Example 3.1 Take the \((N, k, d) = (4, 2, 3)\) RS code over finite field \(GF(4) = GF(2^2) = \{0, 1, \alpha, \alpha^2\}\). The primitive element \(\alpha\) is the root of \(x^2 + x + 1 = 0\). From theorem 3.2.1 we know that, if

\[ N - w(N - d) > 0, \]

then there exists a broadcast encryption scheme based on such a RS code, which means that \(w < \frac{N}{N - d} = \frac{4}{1 - 3} = 4\).
Since \( k = 2 \), the information sequence is

\[
\overline{m} = (m_1, m_2).
\]

The codewords, that is KEKs for all users corresponding to all possible information sequence is shown in Table 1.

<table>
<thead>
<tr>
<th>( m_2 )</th>
<th>( m_1 )</th>
<th>( h(0) )</th>
<th>( h(1) )</th>
<th>( h(\alpha) )</th>
<th>( h(\alpha^2) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( u_1 )</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>( u_2 )</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>( u_3 )</td>
<td>0</td>
<td>( \alpha )</td>
<td>( \alpha )</td>
<td>( \alpha )</td>
<td>( \alpha )</td>
</tr>
<tr>
<td>( u_4 )</td>
<td>0</td>
<td>( \alpha^2 )</td>
<td>( \alpha^2 )</td>
<td>( \alpha^2 )</td>
<td>( \alpha^2 )</td>
</tr>
<tr>
<td>( u_5 )</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>( \alpha )</td>
</tr>
<tr>
<td>( u_6 )</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>( \alpha^2 )</td>
</tr>
<tr>
<td>( u_7 )</td>
<td>1</td>
<td>( \alpha )</td>
<td>( \alpha )</td>
<td>( \alpha^2 )</td>
<td>0</td>
</tr>
<tr>
<td>( u_8 )</td>
<td>1</td>
<td>( \alpha^2 )</td>
<td>( \alpha^2 )</td>
<td>( \alpha )</td>
<td>1</td>
</tr>
<tr>
<td>( u_9 )</td>
<td>( \alpha )</td>
<td>0</td>
<td>0</td>
<td>( \alpha )</td>
<td>( \alpha^2 )</td>
</tr>
<tr>
<td>( u_{10} )</td>
<td>( \alpha )</td>
<td>1</td>
<td>1</td>
<td>( \alpha^2 )</td>
<td>( \alpha )</td>
</tr>
<tr>
<td>( u_{11} )</td>
<td>( \alpha )</td>
<td>( \alpha )</td>
<td>( \alpha )</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>( u_{12} )</td>
<td>( \alpha )</td>
<td>( \alpha^2 )</td>
<td>( \alpha^2 )</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>( u_{13} )</td>
<td>( \alpha^2 )</td>
<td>0</td>
<td>0</td>
<td>( \alpha^2 )</td>
<td>1</td>
</tr>
<tr>
<td>( u_{14} )</td>
<td>( \alpha^2 )</td>
<td>1</td>
<td>1</td>
<td>( \alpha )</td>
<td>0</td>
</tr>
<tr>
<td>( u_{15} )</td>
<td>( \alpha^2 )</td>
<td>( \alpha )</td>
<td>( \alpha )</td>
<td>1</td>
<td>( \alpha^2 )</td>
</tr>
<tr>
<td>( u_{16} )</td>
<td>( \alpha^2 )</td>
<td>( \alpha^2 )</td>
<td>( \alpha^2 )</td>
<td>0</td>
<td>( \alpha )</td>
</tr>
</tbody>
</table>

Table 1. User KEKs constructed from \((4,2,3)\) R-S code.

3.2.2 Discussion

1. From [7], it is also known that any \((N,k,d)\) error correcting code gives rise to a PHF\((N,n,m,w)\) which is proven to be effective to set up the multicast encryption scheme. There, it requires that the minimum distance of the error-correcting code \(d'\) should satisfies that

\[
d' > \left( \left( \frac{w}{2} \right) - 1 \right) N \left( \frac{w}{2} \right).
\]

That is,

\[
d' = \left[ \left( \frac{w}{2} \right) - 1 \right] N + 1.
\]

While here, from the above theorem, since

\[
N > w \times (N - d),
\]

6
the minimum distance \( d \) has to satisfy that
\[
d > \left(1 - \frac{1}{w}\right)N.
\]
That is,
\[
d = \left\lfloor \frac{w-1}{w}N \right\rfloor + 1.
\]
Because \( d \leq d' \), the requirement for constructing the broadcast encryption scheme using R-S code had been reduced, which allows us to increase the length of the information bit \( k \) when the code length is fixed and further more to reduce the requirement for the bandwidth.

2. For any \([n, k, d]_q\) R-S code over finite field \( F_q \), when \( N = q \), \( k = \log_q n \), where \( n \) is the number of codewords,
\[
d' = N - k + 1 = q - \log_q n + 1,
\]
then from
\[
N < w \ast (N - d),
\]
we get
\[
w < \frac{N}{N-d} = \frac{q}{\log_q n - 1}.
\]

**Example 3.2** Take an \([8, 3, 6]_8\) R-S code over finite field \( GF(8) = \{0, 1, \alpha, \alpha^2, \alpha^3, \alpha^4, \alpha^5, \alpha^6\} \) as an example. Since \( q = 8, N = q = 8, k = 3, d = N - k + 1 = 8 - 3 + 1 = 6 \), then the number of codewords \( n = 8^3 = 512 \) and
\[
w < \frac{N}{N-d} = \frac{q}{\log_q n - 1} = \frac{8}{3 - 1} = 4.
\]

So the \([8, 3, 6]_8\) R-S code over finite field \( GF(8) \) can be used to construct the secure broadcast scheme as long as the number of members who want to quit is less than or equal to 3 where \( N > w(N - d) \) \( \iff \) \( 8 > 3(8 - 6) \).

### 3.2.3 Extension of the scheme

In order to improve the communication efficiency, the OR scheme we discussed can be slightly modified with erasure code such that the bandwidth used by the GC for broadcasting information can be reduced.

An \([n, k, m]\) erasure code is a special class of error-correcting codes that allow recovery of a message if part of its messages \((\leq (n-m))\) are damaged or erased during the transmission. An erasure code can be constructed using Reed-Solomon code over finite field \( GF(q) \). The decoding procedure uses \( k \) pairs of \((e_i, p(x))\) to recover the original \( k \) information messages, where \( e_i \) is one of the field element over \( GF(q) \) and \( p(x) = v_0 + v_1(x) + \cdots + v_{k-1}(x^{k-1}) \) is the polynomial for the R-S encoding.

The broadcast encryption scheme described in Theorem 3.1 can be modified as follows.

In the broadcast phase, before broadcasting the new session key \( k^{U\setminus W} \), an encoding procedure is first applied to the new session key. The new session key \( k^{U\setminus W} \) was divided into \( t \) pieces \( k^{U\setminus W} = (k_1^{U\setminus W}, k_2^{U\setminus W}, \ldots, k_t^{U\setminus W}) \), then encodes them using \([Nm, t, \alpha]\) erasure code.
to obtain the codeword $C(k^U \setminus W) = (c_1, c_2, ..., c_{Nm})$. The GC uses all the KEKs that do not belong to the users of $W$ to encrypt the corresponding components of $C(k^U \setminus W)$ and broadcasts the encrypted messages to all the users. That is, the GC broadcasts

$$\{E_{k_i}(c_i) \mid k_i \in K, k_i \notin K(W)\}.$$

As long as each non-excluded user has at least $\alpha$ keys that can decrypt $\alpha$ messages of $\{E_{k_i}(c_i)\}$, he can then apply the erasure code to obtain the new session key. While for those users in $W$, same as before, they can not find the session keys.

**Theorem 3.2** The above scheme works as long as the following equality holds:

$$N - w(N - d) \geq \alpha.$$

For an $[n, k, m]$ erasure code over $GF(q)$, we expect $k$ to be as large as possible in order to minimize the extra bandwidth $n/k$ for the transmission. Actually, the basic scheme we discussed in Theorem 3.1 is a special case of using $[n, 1, 1]$ erasure code for the construction.

**Example 3.3** Consider the same example as in Example 3.1: the RS code $(N, k, d) = (4, 2, 3)$ over finite field $GF(4) = GF(2^2) = \{0, 1, \alpha, \alpha^2\}$. The primitive element $\alpha$ is the root of $x^2 + x + 1 = 0$. The KEKs for all users corresponding to all possible information sequence is shown in Table 1.

For the above scheme to work, it needs that

$$N - w(N - d) \geq \alpha,$$

that is,

$$4 - w(4 - 3) \geq \alpha,$$

that is,

$$4 - w \geq \alpha.$$

For $\alpha = 1$, $w$ can be 1, 2 or 3. For $\alpha = 2$, $w$ can be 1 or 2. And for $\alpha = 3$, $w$ can only be 1. We take $w = 2$ and $\alpha = 2$ as an example.

We divide the new session key $k^U \setminus W$ into two parts $k^U \setminus W = (k_0^U \setminus W, k_1^U \setminus W)$, then encodes $k^U \setminus W$ using a $[16, 2, 2]$ erasure code to obtain a codeword

$$C(k^U \setminus W) = (c_1(0), c_2(1), c_3(\alpha), ..., c_{16}(\alpha^{14})),$$

where

$$c_i(e_i) = p(e_i), e_i \in GF(16)$$

and

$$p(x) = k_0 + k_1x.$$

Suppose any two users $W = \{u_{s_1}, u_{s_2}\}, |W| = 2$ want to leave the group, the GC uses all the KEKs that do not belong to this two users to encrypt all these 16 pieces of encoded keys and broadcasts the encrypted messages to all the users. Then each user that is not in $W$ has at least 2 keys to decrypt 2 messages, thus can recover the original new session key.
3.3 Second R-S code construction

In [8], Hartono proposed a broadcast encryption scheme based on the KDP (Key Distribution Pattern) which can be used for dynamic GC. If the GC is fixed in the system and is trustable, then the condition for the scheme can be improved to make it work for general case.

3.3.1 Scheme description

**Theorem 3.3** Let $X = \{x_1, x_2, \ldots, x_n\}$ be a set of KEKs, $U = \{U_1, U_2, \ldots, U_n\}$ be the set of users' KEKs, which is a family of subset of $X$, that is, for $\forall i$, $U_i \subseteq X$. Let $W = \{U_1, U_2, \ldots, U_n\}$ be a subset of $U$ with $|W| = w$. If for $\forall i$, it satisfies that:

$$|U_i \setminus \bigcup_{k=1}^{w} U_{i_k}| \geq 1,$$

then an broadcast encryption scheme can be constructed which can remove up to $w$ users from a group of $n$ users and the system is secure against collusion of $t$ malicious users.

In this scheme, The GC generates and stores a set $X$ of KEKs in the key initialization phase, and sends each user $u_i$ a subset $U_i \subseteq U$ of $X$ as the user’s KEKs. When a set of users $W$ want to quit from the group, the GC selects a new session key $k^{U \setminus W}$ and encrypts the session key with all KEKs except those belong to users in $W$, that is, GC broadcasts $\{E_{k_r}(k^{U \setminus W}) \mid k_r \in X \setminus (U_1 \cup U_2 \ldots \cup U_n)\}$. So, those users in $W$ can not decrypt the encrypted message. While, since for $\forall i$, that $U_i \in X \setminus W$,

$$|U_i \setminus \bigcup_{k=1}^{w} U_{i_k}| \geq 1,$$

it has at least one key that does not belong to $W$, so it can decrypt $E_{k_r}(k^{U \setminus W})$ and obtain the new session key $k^{U \setminus W}$.

From the theorem we know that for any given $w$ and $n$, we should make $n^*$ as small as possible. Same, for any given $w$ and $n^*$, we hope $n$ to be as large as possible.

Next, we will show how to use Reed-Solomon code to construct the KEK set $X$ and $U$ and how the scheme works.

3.3.2 R-S code construction

We take the R-S code $(N, k, d)$ over the finite field $GF(q) = \{0, 1, \alpha, \ldots, \alpha^{q-2}\}$. The number of users $n = q^k$. The RS codeword $\mathbf{s}$ of length $N$ is generated from $k$ information symbols taken from the finite field $GF(q)$ through polynomial

$$h(x) = m_0 + m_1 x + m_2 x^2 + \ldots + m_{k-2} x^{k-2} + m_{k-1} x^{k-1},$$

where

$$\mathbf{m} = (m_0, m_2, \ldots, m_{k-1})$$

and

$$\mathbf{s} = (c_0, c_1, \ldots, c_{q-1}) = (h(0), h(\alpha), h(\alpha^2), \ldots, h(\alpha^{q-1})).$$

For each user $u_i$, the KEK set that corresponds to the $k$ information symbols

$$\mathbf{m}_i = (m_{i1}, m_{i2}, \ldots, m_{ik})$$
is

\[ U_i = \{(h_i(0), h_i(\alpha), ..., h_i(\alpha^{q-1})\}) \],

where \(|U_i| = q = N\). So, the KEK set for all users is

\[ U = \cup_{i=1}^{n} U_i. \]

The total KEK set \(X\) for GC is \(X = \{X_i, i = 1, 2, ..., n^*\}\), where \(n^* = N \times q = q^2\), and

\[ X_i = \{(h, \beta) \mid h \in \{h(0), h(\alpha), ..., h(\alpha^{q-1})\}, \beta \in GF(q)\} \]

Next, we will use an example to show the exact procedure on the RS-code construction of the scheme.

**Example 3.4** Take the same example as in Example 3.1: that is the \((N, k, d) = (4, 2, 3)\) RS code over finite field \(GF(4) = GF(2^2) = \{0, 1, \alpha, \alpha^2\}\). The primitive element \(\alpha\) is the root of \(x^2 + x + 1 = 0\). The KEKs for all users corresponding to all possible information sequence is shown in Table 1. After extending the users’ KEKs by use of the way shown in section 3.3.2, we obtain the KEKs set \(X = \{X_i, i = 1, 2, ..., 16\}\) as shown in Table 2.

<table>
<thead>
<tr>
<th></th>
<th>(h_1)</th>
<th>(h_1)</th>
<th>(h_1)</th>
<th>(h_2)</th>
<th>(h_2)</th>
<th>(h_2)</th>
<th>(h_2)</th>
<th>(h_3)</th>
<th>(h_3)</th>
<th>(h_3)</th>
<th>(h_3)</th>
<th>(h_4)</th>
<th>(h_4)</th>
<th>(h_4)</th>
<th>(h_4)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(x_1)</td>
<td>(x_2)</td>
<td>(x_3)</td>
<td>(x_4)</td>
<td>(x_5)</td>
<td>(x_6)</td>
<td>(x_7)</td>
<td>(x_8)</td>
<td>(x_9)</td>
<td>(x_{10})</td>
<td>(x_{11})</td>
<td>(x_{12})</td>
<td>(x_{13})</td>
<td>(x_{14})</td>
<td>(x_{15})</td>
</tr>
<tr>
<td>(u_1)</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>(u_2)</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>(u_3)</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>(u_4)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>(u_5)</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>(u_6)</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>(u_7)</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>(u_8)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>(u_9)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>(u_{10})</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>(u_{11})</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>(u_{12})</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>(u_{13})</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>(u_{14})</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>(u_{15})</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>(u_{16})</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

**Table 2. Construction of Re-keying scheme with \((4, 2, 3)\) R-S code.**

From Table 2 we can see that,

\[ U_1 = \{x_1, x_5, x_9, x_{13}\} = \{(h_1, 0), (h_2, 0), (h_3, 0), (h_4, 0)\} \]

\[ U_2 = \{x_2, x_6, x_{10}, x_{14}\} = \{(h_1, 1), (h_2, 1), (h_3, 1), (h_4, 1)\} \]
\[ U_3 = \{ x_3, x_7, x_{11}, x_{15} \} = \{(h_1, \alpha), (h_2, \alpha), (h_3, \alpha), (h_4, \alpha)\} \]
\[ U_4 = \{ x_4, x_8, x_{12}, x_{16} \} = \{(h_1, \alpha^2), (h_2, \alpha^2), (h_3, \alpha^2), (h_4, \alpha^2)\} \]
\[ U_5 = \{ x_1, x_6, x_{11}, x_{16} \} = \{(h_1,0), (h_2,1), (h_3, \alpha), (h_4, \alpha^2)\} \]
...
\[ U_{16} = \{ x_4, x_5, x_{11}, x_{14} \} = \{(h_1, \alpha^2), (h_2,0), (h_3, \alpha), (h_4, 1)\} \]

All the KEKs hold by the GC is given by:
\[ U = \bigcup_{i=1}^{16} U_i. \]

Suppose there are \( w = 2 \) users who want to quit from the group \( \{ u_1, u_2, ..., u_{16} \} \), say users \( W = \{ u_7, u_8 \} \), we can check that for each user \( u_i \notin W \),
\[ |U_i \setminus \bigcup_{k=1}^{w} U_{s_k}| \geq 1. \]

For example,
\[ |U_1 \setminus \bigcup_{k=1}^{2} U_{s_k}| = |\{ x_1, x_5 \}| = 2 \geq 1. \]

So such a set of KEKs can be used to implement the broadcast encryption scheme.

4 Construction of the Scheme using A-G code

Since the R-S code over \( GF(q) \) requires the length of the codewords \( N \leq q \), we can not make the codeword longer than \( q \). Using Algebraic-geometric code (A-G code), the scheme can be extended to the case when codeword length \( N > q \). Next we will show an example on how to use A-G code to construct the OR model for the multicast re-keying scheme.

4.1 A-G code

For those who are interested in more details about A-G code, please refer to the paper [11] and [12].

4.2 Example of S-G code based multicast re-keying scheme

Let us consider the Hermitian code over \( GF(4) = GF(2^2) \) with \( k = 2 \). The Hermitian curve over \( GF(2^2) \) is \( x^3 + y^2 + y = 0 \). The curve has rational points:
\[ \{(0,0), (0,1), (1, \alpha), (1, \alpha^2), (\alpha, \alpha), (\alpha, \alpha^2), (\alpha^2, \alpha), (\alpha^2, \alpha^2)\} \]
\[ =^\Delta \{(x_1, y_1), (x_2, y_2), (x_3, y_3), (x_4, y_4), (x_5, y_5), (x_6, y_6), (x_7, y_7), (x_8, y_8)\}. \]

Let code polynomial be \( c(x) = m_0 + m_1x \), it has 16 codewords:
\[ \sigma = (c(x_1, y_1), c(x_2, y_2), c(x_3, y_3), c(x_4, y_4), c(x_5, y_5), c(x_6, y_6), c(x_7, y_7), c(x_8, y_8)) \]

All the codewords, that is, the KEKs set are shown in Table 3.
<table>
<thead>
<tr>
<th>$m_2$</th>
<th>$m_1$</th>
<th>$h(0)$</th>
<th>$h(1)$</th>
<th>$h(\alpha)$</th>
<th>$h(\alpha^2)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$u_1$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$u_2$</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$u_3$</td>
<td>0</td>
<td>$\alpha$</td>
<td>0</td>
<td>$\alpha$</td>
<td>$\alpha^2$</td>
</tr>
<tr>
<td>$u_4$</td>
<td>0</td>
<td>$\alpha^2$</td>
<td>0</td>
<td>$\alpha^2$</td>
<td>$\alpha^2$</td>
</tr>
<tr>
<td>$u_5$</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$u_6$</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>$\alpha^2$</td>
</tr>
<tr>
<td>$u_7$</td>
<td>1</td>
<td>$\alpha$</td>
<td>1</td>
<td>$\alpha^2$</td>
<td>$\alpha$</td>
</tr>
<tr>
<td>$u_8$</td>
<td>1</td>
<td>$\alpha^2$</td>
<td>1</td>
<td>$\alpha^2$</td>
<td>$\alpha$</td>
</tr>
<tr>
<td>$u_9$</td>
<td>$\alpha$</td>
<td>0</td>
<td>$\alpha$</td>
<td>$\alpha$</td>
<td>$\alpha$</td>
</tr>
<tr>
<td>$u_{10}$</td>
<td>$\alpha$</td>
<td>1</td>
<td>$\alpha$</td>
<td>$\alpha^2$</td>
<td>0</td>
</tr>
<tr>
<td>$u_{11}$</td>
<td>$\alpha$</td>
<td>$\alpha$</td>
<td>$\alpha$</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>$u_{12}$</td>
<td>$\alpha$</td>
<td>$\alpha^2$</td>
<td>$\alpha$</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$u_{13}$</td>
<td>$\alpha^2$</td>
<td>0</td>
<td>$\alpha^2$</td>
<td>$\alpha^2$</td>
<td>$\alpha$</td>
</tr>
<tr>
<td>$u_{14}$</td>
<td>$\alpha^2$</td>
<td>1</td>
<td>$\alpha^2$</td>
<td>$\alpha$</td>
<td>1</td>
</tr>
<tr>
<td>$u_{15}$</td>
<td>$\alpha^2$</td>
<td>$\alpha$</td>
<td>$\alpha^2$</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$u_{16}$</td>
<td>$\alpha^2$</td>
<td>$\alpha^2$</td>
<td>$\alpha^2$</td>
<td>0</td>
<td>$\alpha$</td>
</tr>
</tbody>
</table>

Table 3. User KEKs constructed from (8,2,6) A-G code.

In this example, $e(x)$ has at most 2 zero points, $d = 8 - 2 = 6$. Since $q = 4, N = 8, d = 6$, the number of users $n = q^2 = 16$, the number of keys is $N * q = 8 * 4 = 32$.

For the OR multicast re-keying scheme to work, it requires that

$$N > w(N - d),$$

that is,

$$8 > w(8 - 6),$$

so $w$ can be 2 or 3. Since $w$ can be 3, from $N - w(N - d) = \alpha$, we know that $\alpha$ can be 2.

5 Conclusions

In this paper, two practical constructions for Multicast Re-keying Schemes using Reed-Solomon Codes are given with examples to show the detailed construction procedure. Because it has many properties we expected, RS code provides us a practical way to construct the multicast re-keying scheme efficiently. The storage complexity and transmission complexity have been reduced at the same time, which is another advantage of the method proposed in this paper.

Because this paper is only an initial work for using RS-code in constructing the re-keying scheme, a lot of work is being done and will be done in the future such as how to improve the communication transmission efficiency by encoding the new session key with error correcting code first, how to deal with multiuser and multistage leaving from the group, how to handle when a new user is joining, but the members in the group has reached the maximum, how to apply AG code instead of RS code in the construction to improve the performance, how to make the GC be a group member also, how to extend these two schemes to apply for the distributed environment and so on.
References


[11]

[12]