

SEC: A Practical Secure Erasure Coding Scheme for Peer-to-Peer Storage System¹

Jing Tian, Zhi Yang, Yafei Dai

CNDS Lab, Peking University
{tianjing, yangzhi, dyf}@net.pku.edu.cn

Abstract. Though conventional block cipher is widely used in Peer-to-Peer storage systems, it is insufficient to ensure the confidentiality of long-term archive data because of the inherent Peer-to-Peer storage vulnerability that data is stored on other untrusted peers. There are two potential security weak spots: (1) sensitive data may be exposed from the encrypted data stored on an adversary peer when the weakness of block cipher is discovered or there is a leakage of secret key; (2) even when a user realizes the security threats, he/she cannot destroy the encrypted data stored on an adversary peer to minimizing the loss.

This paper proposes a novel secure erasure code (SEC) scheme to solve the above security problems. The SEC scheme encodes the sensitive data into several fragments, and then stores the fragments onto different peers. In theory, SEC ensures unconditional confidentiality at the fragment level, which means that an adversary cannot obtain any portion of sensitive data from local stored encrypted fragment even if he/she has the secret key and infinite computation power. By leveraging the large scale property of Peer-to-Peer systems, SEC further makes it infeasible for the adversary to collect enough fragments for decrypting. SEC can be used alone or together with block cipher to solve these potential security problems. The performance results show that the SEC scheme is practical for the real-world applications.

1 Introduction

In recent years, the Peer-to-Peer technology has been shown to be promising in constructing large scale persistent storage. Many Peer-to-Peer storage systems are proposed, such as [1] [2] [3] [4] [5] [6]. Extensive research efforts have focused on providing stable overlay network and long-term data durability. However, there has been little research reported on the confidentiality of the data.

In Peer-to-Peer storage systems, the user's data is stored on disks of other untrusted peers, and any peer can leave the system at any time. Consequently, there is a great challenge for the confidentiality and durability of user's data. The conventional approach used in several proposed systems [1] [2] [6] is to encrypt the user's data to

¹ This work is supported by National Grand Fundamental Research 973 program of China under Grant No.2004CB318204, National Natural Science Foundation of China under Grant No.90412008

ensure its confidentiality, and then store the encrypted data using either a replication or an erasure code [7] strategy. However, we believe that only the block cipher encryption is insufficient because of the inherent Peer-to-Peer storage system vulnerability that data is stored on other untrusted peers. The potential security weak spots are: (1) sensitive data may be exposed from the encrypted data stored on an adversary peer when the weakness of block cipher is discovered or there is a leakage of secret key; (2) even when a user realizes the security threats, the user cannot destroy the encrypted data stored on an adversary peer to minimizing the loss. In other words, once the encrypted data is stored, the owner is put in a passive position in the face of various security threats. These problems set great barrier for the applications of the Peer-to-Peer storage system.

To solve these problems, this paper proposes a novel secure erasure code (SEC) scheme for the Peer-to-Peer storage systems. The SEC scheme is in fact a customized Reed-Solomon erasure code. By customizing, SEC provides a strong strength data encryption as well as the data redundancy at the same time. The SEC first encrypts data into several fragments, and then distributes the fragments out as independent objects to different peers by the underlying Peer-to-Peer storage protocol. The SEC scheme guarantees that less than a threshold number of fragments do not give any information about any portion of the encrypted data theoretically. This property prevents adversary from decrypting the sensitive data only from the local stored fragment even if the weakness of the cipher is discovered or the adversary obtains the secret key. By the system assumption detailed in section 4, the SEC scheme further makes it infeasible that the adversary can collect enough fragments of one object for decrypting throughout the Peer-to-Peer system. Even though the adversary may collect enough fragments, the strong strength encryption of SEC protects the confidentiality of data. Thus, SEC provides a three-layer security. The SEC alone can be used as an encryption scheme, while it can also be used with conventional block cipher to solve security problems in Peer-to-Peer storage systems.

The body of this paper is organized as follows. After introducing some preliminary knowledge in section 2, we describe the whole SEC scheme in section 3. In section 4, the system model and a three-layer security analysis framework are presented, and the optimal parameter of the SEC is suggested according to a detailed analysis of the security. The performance results are given in section 5 which proves the practicality of the SEC scheme. In section 6, we show the two application modes of the SEC. After introducing the related works in section 7, we conclude this paper in section 8.

2 Reed-Solomon Codes Preliminaries

As opposed to replication scheme, erasure codes provide a more flexible redundancy scheme to tolerate storage component failures. Erasure codes divide a data object into m fragments, and encode them into n fragments, where all the fragments have same size and $n \geq m$, so the redundancy rate r is n/m . When decoding, erasure codes only need any t ($t \geq m$) fragments out of all the n encoded fragments to reconstruct the data. The original data fragments can be viewed as a vector $D = (D_1, D_2, \dots, D_m)$, and the

encoding function Enc maps the original data fragments onto the encoded fragments $Enc(D) = E$, $E = (E_1, E_2, \dots, E_n)$. The decoding function Dec maps a subset of E back to the original data, $Dec(E_{i_1}, E_{i_2}, \dots, E_{i_m}) = D$.

Reed-Solomon codes are an important subclass of erasure codes. Reed-Solomon codes are *linear codes*, and that means the encoding function is linear. Consequently, the encoding function of a Reed-Solomon code can be described by a vector-matrix multiplication $Enc(D) = DG$ of a generator matrix G and the original data fragments vector D , where G is an $m \times n$ matrix.

Reed-Solomon codes are also *maximal distance separable* (MDS) codes, where MDS means that any m encoded data fragments are sufficient to recover the data. It is easy to see that the MDS property requires that any $m \times m$ square submatrix of G is non-singular. Consequently, the decoding function can also be described by vector-matrix multiplication $Dec(E') = E'G^{-1}$, where E' is an m elements sub vector of E and G' is the submatrix with m corresponding rows of G .

To achieve a low computation cost, designers usually employ the *systematic* Reed-Solomon codes. A code is called systematic if and only if its generator matrix has the form $(I_m R)$, where I_m is an $m \times m$ identity matrix. Hence, the m original data fragments also appear in the encoded fragments.

3 The SEC Scheme

The SEC scheme is composed of two parts, the encryption part and the fragment naming part. The encryption part is in fact a customized version of Reed-Solomon codes, which encodes the original data into several encrypted fragments. Then the fragment naming part assigns the encrypted fragments with some meaningless names and distributes them out. Recall that the Reed-Solomon code need the G^{-1} to reconstruct the original data, then if we keep the generator matrix G as the secret key, the adversary cannot get the original data even when he/she has collected enough encoded fragments. Based on this fact, SEC encrypts the data by creating a secret generator matrix G_{SEC} from the user specified key. The fragment naming part further makes it impractical for the adversary to collect enough fragments. In this section, we first show how to construct the required G_{SEC} , and then describe the fragment naming algorithm.

3.1 The Secure Generator Matrix

The Reed-Solomon codes do work in an infinite Z field. However, the real domain and range of the computation in our computer are binary words of a fixed length L . Hence, we must perform the Reed-Solomon codes over a *Galois Field* with 2^L elements, denoted by $GF(2^L)$ [8]. The elements of $GF(2^L)$ are the integers from zero to $2^L - 1$, and each element is represented by a L bits word. To ensure the correctness of our computation, the $GF(2^L)$ should at least contain more than $n+m$ elements[9], namely $2^L > m+n$. In the following discussion, all the arithmetic is over the $GF(2^L)$.

However, the detailed computation over Galois Field is beyond the scope of this paper. Over the $GF(2^L)$, we define our G_{SEC} over $GF(2^L)$ as follows

Definition 1 An $m \times n$ matrix G over $GF(2^L)$ is called a secure generator matrix of the SEC scheme if and only if G is non-singular and it does not contain any $(m-1) \times (m-1)$ singular submatrix.

One important subclass of non-singular matrices over $GF(2^L)$ is Cauchy matrices, and we construct our G_{SEC} by customizing the Cauchy matrix. Here, we give the definition of Cauchy matrix.

Definition 2 Let $\{x_1, \dots, x_m\}$ and $\{y_1, \dots, y_n\}$ be two sets of elements in $GF(2^L)$. If the two sets satisfy the following constraint

- (1) $x_i \neq x_j \quad \forall i, j \in \{1, \dots, m\}, i \neq j$
- (2) $y_i \neq y_j \quad \forall i, j \in \{1, \dots, n\}, i \neq j$
- (3) $x_i \neq y_j \quad \forall i \in \{1, \dots, m\}, \forall j \in \{1, \dots, n\}$

then, the matrix

$$\begin{bmatrix} \frac{1}{x_1 + y_1} & \frac{1}{x_1 + y_2} & \dots & \frac{1}{x_1 + y_n} \\ \frac{1}{x_2 + y_1} & \frac{1}{x_2 + y_2} & \dots & \frac{1}{x_2 + y_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{1}{x_m + y_1} & \frac{1}{x_m + y_2} & \dots & \frac{1}{x_m + y_n} \end{bmatrix}$$

is called a Cauchy matrix over $GF(2^L)$.

A Cauchy matrix has the following property, and the proof of it can be found in [10].

Theorem 1 Every square submatrix of a Cauchy matrix is non-singular.

3.2 The Construction of G_{SEC}

G_{SEC} requires any $(m-1) \times (m-1)$ square submatrix to be non-singular, and a Cauchy matrix satisfies this condition. However, it is impractical to keep such a big matrix G_{SEC} as secret key. Contrarily, we generate the G_{SEC} from a user specified secret key. In this subsection, we introduce how to construct the G_{SEC} by customizing a Cauchy matrix according to a user specified l_k bit secret key k_G .

Since the G_{SEC} is an $m \times n$ matrix over $GF(2^L)$, we also need m and n to be algorithm parameters besides k_G . Here, we require l_k to be a multiple of $m+n$, and $L=l_k/(m+n)$. Subsequently, the k_G can be segmented into a vector consisting of $m+n$ elements, where each element is a bit string of size L . By using the pseudo random function, we get a new vector of $m+n$ distinct elements from the prior vector. Finally, these $m+n$ elements can be used to generate the G_{SEC} . The pseudocode is in Fig1.

In the key generation procedure, there are two constraints of the parameters as follows,

$$L > \log_2(m+n) = \log_2(m \times (1+r)) \quad (1)$$

$$l_k = L \times (m+n) = L \times m \times (1+r) \quad (2)$$

To encode the data, we should segment the original data into vectors, each of which has m bit strings of size L . Hence, for each vector, the vector-matrix multiplication yields an encoded vector of size n . Thus, all the i th elements in all encoded vec-

```

generateMatrix(string key, int m, int n)
{
    /*check if Matrix can be constructed*/
     $l_k \leftarrow \text{key.length}$ ;
    if ( $l_k \% (m+n) \neq 0$ )
        return false;
     $L \leftarrow l_k / (m+n)$ ;
    if ( $L \leq \log_2(m+n)$ )
        return false;

    /*get n+m different elements from key*/
    get  $m+n$  elements by partitioning key, where each element contains  $L$  bits of key;
    vector  $E \leftarrow m+n$  elements;
    for  $i \leftarrow 1$  to ( $m+n$ )
        if ((there exists prior element  $E[j]$ ) && ( $E[i] == E[j]$ ))
            pseudo_random.setSeed ( $E[i]$ );
        do
             $E[i] \leftarrow \text{pseudo\_random.nextInt}(2^L - 1)$ ;
        until no prior element equals  $E[i]$ ;

    /*construct the secret matrix */
    vector  $x \leftarrow$  first  $m$  elements of vector  $E$ ;
    vector  $y \leftarrow$  last  $n$  elements of vector  $E$ ;
    for  $i \leftarrow 1$  to  $n$ 
        for  $j \leftarrow 1$  to  $m$ 
             $G_{SEC}[i, j] \leftarrow 1 / (x_i + y_j)$ ;
    return true;
}

```

Fig1: The pseudocode to construct the G_{SEC}

tors form the i th encoded fragment, and then we get n encoded fragments.

3.3 Fragments Naming and Placement

To keep the whole data secure, we store each encoded fragment as a new object individually in the Peer-to-Peer storage system to make it impractical for the adversary to collect enough fragments. Here, we employ a strong one-way hash function *Hash*, e.g. MD5 or SHA1, to hide the relationship between any two fragments of the same original data. We use SHA1 in this paper. Besides, we also need a bit vector V for each

original data and a separate key k_N . The filename of the i th fragment f_i is generated as follows for $i=1, 2, \dots, n$ (\parallel is the bit concatenation):

$$\begin{cases} f_i = \text{Hash}(f_{i-1} \parallel k_N \parallel V) \\ f_0 = "" \end{cases} \quad (3)$$

The V can be a concatenation of some properties of the original data, e.g. the path-name and the last modified time of a file. Though the k_N can be the same with the key k_G for G_{SEC} , it is not suggested to do that.

After assigned meaningless names, all the fragments should be distributed out randomly or by their names, and be stored on different peers throughout the whole Peer-to-Peer system using the underlying protocol.

4 The Security of SEC

Unlike the block cipher, the SEC scheme provides data confidentiality at three layers. In this section, after giving the system model we present a three-layer security analysis framework for the SEC scheme, and then give a detailed analysis of the confidentiality based on the analysis framework, and finally we suggest the optimal parameter selection according to the analysis.

4.1 System Model

In the following analysis, we distinguish the two security levels, the information-theoretic security and computational security. The information-theoretic security, also called “unconditional security”, means that the ciphertext gives no information about the original data even if the adversary has infinite computational power. Compared with information-theoretic security, the computational security is a little weaker, and it means that no polynomial-time algorithm can reconstruct the original data from the ciphertext.

By classifying the adversary’s intent, there are two adversary models for Peer-to-Peer storage systems, the *passive* adversary and the *active* adversary. A passive adversary only tries to peek at the data stored on the disk of local computer, and the adversary does not make an effort to compromise more computers for more information. In a Peer-to-Peer storage system, every user can be a passive adversary since the data are all stored on peers. Therefore, the passive adversaries are very dangerous though they seem somewhat naïve. It is clear that the conventional block cipher scheme is only computational secure in the face of passive adversary. An active adversary has more powerful ability. Besides accessing the local stored data, the active adversary may request other files in the system, or compromise other peers to learn the desired files. In particular, an active adversary tries to collect all the fragments for one data object in a system using the SEC scheme. SEC provides very strong computational security in the face of the active adversary.

In this paper, we make the following system assumptions of the underlying Peer-to-Peer system: (1) it provides anonymous routing [12] [13]; (2) it is Byzantine-fault-

tolerant [14] [15]; (3) it is Sybil attack resistant [16]. These system assumptions are hot issues of Peer-to-Peer security, which have been extensively studied.

4.2 Three-layer Security Framework

The SEC scheme provides the following three-layer securities,

Fragment Security: The fragment security is the base layer security, which provides an information-theoretic secure at fragment level. This ensures that less than m fragments reveal nothing about any fragment of the original data theoretically. It has been proved in [17] that less than m shares definitely give some information about the secret when the size of the share is smaller than the secret. However, less than m fragments can give no information about any single fragment of original data because the encoded fragments and the original fragments are in the same size.

Distribution Security: The SEC scheme assigns the fragments with different meaningless names, which makes it impractical for the adversary to find out all the fragments of one data object throughout the large scale Peer-to-Peer system.

Encryption Security: Though an adversary may collect enough m fragments, he/she has to find out the secret matrix G_{SEC} to decrypt encoded fragments.

The fragment security is important since a single fragment is theoretically meaningless for the adversary. Therefore, only with the local stored fragment the adversary cannot get any portion of the sensitive data even if the weakness of the cipher is discovered or the adversary gets the secret key. Thus, the passive adversary is no longer a threat for the sensitive data in SEC scheme, while the data stored on passive adversary is the core reason of the potential security problems in conventional scheme. Subsequently, fragment security enforces active adversary to collect enough fragments, which results in distribution security. Based on the system assumptions, an adversary can only try all the candidate combinations of fragments in the whole system for decrypting, which is shown to be impractical in next subsection. Because the adversary cannot get all the fragments of the sensitive data easily, there is a chance for the user to remove other fragments when there is security threat. Eventually the strong encryption security of SEC protects the sensitive data even if the adversary collects enough fragments.

Without anonymous routing, the owner information of the fragments will be learned by adversaries, which will significantly weaken the distribution security. If there is Byzantine fault or Sybil attack, by attacking the routing tables or using multiple identifiers the adversary can lead more upload traffic onto themselves, and then the adversary can reduce the number of trials of candidate fragment combinations by the timing information of the concurrent uploads. This also weakens the distribution security.

Though we can store the encrypted fragments of conventional block cipher by the simple segmenting, this simple scheme does not provide fragment security and the distribution security. This is because the adversary can obtain the portion of the sensitive data by decrypting a single fragment, and it is not necessary for the adversary to collect all the fragments. Therefore, the fragment security and the distribution security are particular properties of the SEC scheme.

4.3 Analysis of Three-layer Security

With respect to the three-layer security framework, we prove the information-theoretic property of fragment security layer, and give a quantitative analysis on strength of the distribution security layer and the encryption security layer.

Fragment Security

To prove the information-theoretic property of fragment security, we first give the following lemma.

Lemma 2 *Let G be an $m \times m$ matrix. There is no zero element in the inverse matrix G^{-1} if G does not contain any $(m-1) \times (m-1)$ singular submatrix.*

Proof: According to the computation of inverse matrix,

$$G^{-1} = \frac{G^*}{|G|}$$

if there is no $(m-1) \times (m-1)$ singular submatrix in G , there is no zero element in G^* , so there is no zero element in G^{-1} . Q.E.D of Lemma 2.

Theorem 3 *Let the $m \times n$ matrix G be the secure generator matrix of SEC. The SEC scheme provides information-theoretic security at fragment level i.e. the adversary cannot obtain any information about any fragment of the original data from less than m fragments even when the adversary has got the matrix G .*

Proof: Let (D_1, D_2, \dots, D_m) be the original data vector, and $(E_{i_1}, E_{i_2}, \dots, E_{i_m})$ be any encoded fragments vector with m element. Let G' be the $m \times m$ submatrix of G_{SEC} with the columns i_1, i_2, \dots, i_m . Therefore, we have $(D_1, D_2, \dots, D_m) = (E_{i_1}, E_{i_2}, \dots, E_{i_m})G'^{-1}$. According to lemma 2, the G'^{-1} does not have any zero element. As a result, any fragment of the original data D_j must be computed from enough m encoded fragments, and less than m encoded fragments give no information about any fragment of the original data in information theoretic sense. Q.E.D of Theorem 3.

Distribution Security

Since the SEC provides information-theoretic security at the fragment level, the active adversary will try to collect enough fragments throughout the whole system. However, meaningless fragment naming and random placement make it infeasible to collect sufficient m fragments. For a (m, n) SEC scheme, if there are N possible fragments in the system, there are $C(N, m)$ possible combinations of m fragments while only $C(n, m)$ are valid ones. Thus, we get the number of trials for the adversary as following,

$$\frac{C(N, m)}{C(n, m)} = \frac{C(N, m)}{C(m \times r, m)} \quad (4)$$

If the adversary knows the V in (3), then the number of trails is as following,

$$\min \left\{ \frac{C(N, m)}{C(n, m)}, 2^{l_{k_N}} \right\} \quad (5)$$

where l_{k_N} is the length of k_N .

The SEC's distribution security leverages the huge possible candidate fragments in Peer-to-Peer system to keep the high security of the data object. From (4), we find that the distribution security will strengthen dramatically along with the number of candidates N and that the larger redundancy rate will result in worse distribution security. In a real Peer-to-Peer storage system, we can improve the distribution security by increasing m while N is a relative stable property of the system.

Encryption Security

If the worst case happens, that the adversary collects enough m fragments, the SEC would falls back on its third line of defense, the encryption security. Here, we give the analysis on the number of trials for the brute force attack.

Lemma 4 Let $\{x_1, x_2, \dots, x_m, y_1, y_2, \dots, y_n\}$, and $\{x'_1, x'_2, \dots, x'_m, y'_1, y'_2, \dots, y'_n\}$ be two sequences over $GF(2^L)$, where $m > 1$ and $n > 1$. If

$$x_i + y_j = x'_i + y'_j \quad \forall i \in \{1, \dots, m\}, \forall j \in \{1, \dots, n\}$$

then

$$x_i = x'_i, \forall i \in \{1, \dots, m\} \text{ and } y_j = y'_j, \forall j \in \{1, \dots, n\}$$

Proof: Because of that $m > 1$ and $n > 1$, the following linear equation system always exists (a, b, c and d are constants)

$$\begin{cases} x'_1 + y'_1 = a = x_1 + y_1 \\ x'_1 + y'_2 = b = x_1 + y_2 \\ x'_2 + y'_1 = c = x_2 + y_1 \\ x'_2 + y'_2 = d = x_2 + y_2 \end{cases} \quad (6)$$

It is clear that there is only one unique solution for both four variables linear equation systems in (6). Hence, we get $x'_1 = x_1$ and $y'_1 = y_1$. Consequently, we get $y_j = y'_j, \forall j \in \{1, \dots, n\}$ since $x_1 + y_j = x'_1 + y'_j$ and $x'_1 = x_1$. For the same reason, we have $x_i = x'_i, \forall i \in \{1, \dots, m\}$. Q.E.D of Lemma 4.

From Lemma 4, we yield the following theorem,

Theorem 5 A Cauchy matrix is determined by one and only one unique sequence $\{x_1, \dots, x_m, y_1, \dots, y_n\}$

Suppose that the adversary has collected the m encoded fragments $(E_{i_1}, E_{i_2}, \dots, E_{i_m})$, then we have

$$(D_1, \dots, D_m) \begin{pmatrix} 1 & \dots & 1 \\ x_1 + y_{i_1} & \dots & x_1 + y_{i_m} \\ \vdots & \ddots & \vdots \\ 1 & \dots & 1 \\ x_m + y_{i_1} & \dots & x_m + y_{i_m} \end{pmatrix} = (E_{i_1}, \dots, E_{i_m}) \quad (7)$$

where (D_1, D_2, \dots, D_m) is the original data vector. Theorem 5 implies that there is only one sequence $\{x_1, \dots, x_m, y_{i_1}, \dots, y_{i_m}\}$ which can generate the $m \times m$ matrix used in (7). As a result, the adversary can only guess that very sequence over $GF(2^L)$ for the decryption. Since there are $2 \times m$ different elements in the sequence, the adversary must try $P(2^L, 2m)$ times in the worst case. From (2) we yield,

$$P(2^L, 2m) = P(2^{\frac{l_k}{(1+r)m}}, 2m) \quad (8)$$

From (8), it is obvious that for the given l_k and r , the strength is determined by m , and the number of trials will degrade by increasing m .

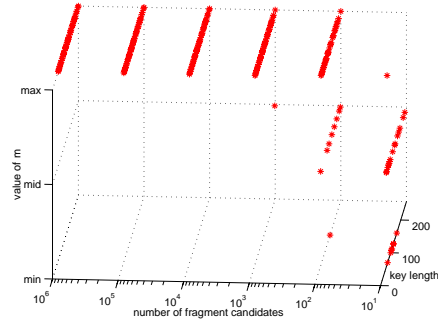


Fig 2: Optimal m for different key length l_k and number of candidate fragments N with $r=1$

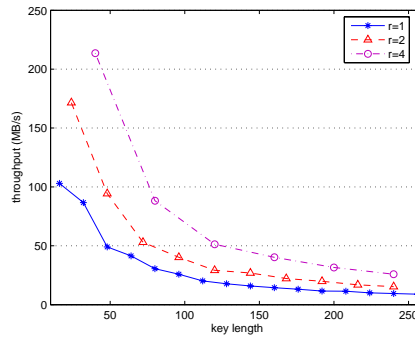


Fig 3: Encoding throughput versus key length, where L is 8 for all key lengths.

5 Performance Evaluation

Although the analysis has shown that the SEC scheme can provide strong strength confidentiality for Peer-to-Peer storage systems, it is important for a real applied scheme to consume less of the end user's CPU cycles in a Peer-to-Peer environment. In this section, we report some performance results of our implementation in Java language. The experiment results are collected on a desktop computer, with a Celeron(R) CPU running at 2.4GHz, and 512MB of main memory, which are the typical settings of the desktop computer.

Computation Overheads

There are some computation overheads before the main vector-matrix multiplication in the encoding and decoding phases of the SEC scheme. First of all, the SEC generates the *log* and *inverse log* tables for every elements in $GF(2^L)$. Then the SEC generates the encoding matrix from the specified key for the encoding, or the inverse matrix for the decoding. We present the performance for a 512-bit key in table 1. It is clear that the computation overheads are negligible.

Table 1. The computation overheads, $l_k=512$, $r=1$, $m=16$, $L=16$

log/inverse log tables in $GF(2^L)$	encoding matrix from key	inverse matrix for decoding
31ms	0.157ms	0.047ms

Encoding and Decoding Throughput

Now we focus on the main computation part of the SEC, the vector-matrix multiplication over $GF(2^L)$. We collect the throughput information for encoding a 100M data object using different l_k and different r , and plot them in Fig 3. From the experiment results, we get

$$\text{encoding throughput} \propto \frac{r+1}{l_k} \quad (10)$$

which can be also deduced from [11]. The throughput for the decoding procedure is the same as the encoding procedure since the vector-matrix computations are the same.

From (10) we get the following implications. First, the encoding and decoding throughputs are both independent with the parameter m when the key length is fixed. Therefore, we can make use of the optimal parameter m discussed in the prior section without influencing the throughput. Secondly, the larger r will result in better throughputs though it decreases the security strength. Thirdly, we can use a smaller key length to achieve better performance, when the SEC is used together with a block cipher and the block cipher provides strong enough encryption security.

Fig 3 shows that the performance of the SEC is good enough. When using a 128-bit key with $r=1$, the SEC achieves a throughput of $20MB/s$. Alternatively speaking, the SEC scheme only costs 7% of CPU cycles for a typical wide area peer with a $10Mb/s$ bandwidth. If short key length and large redundancy rate are used, the SEC scheme can even achieve a throughput over $200MB/s$.

6 Application of SEC Scheme

Stand-alone Encryption

The SEC scheme provides information-theoretic security against the passive adversary and computational security against the active adversary. Therefore, the SEC alone can be used as an encryption scheme while providing the data redundancy as well. The encryption security of the SEC scheme can be very strong with $r=1$, e.g. the

number of trials for a 128-bit key can be $2^{127.99}$. Though the encryption security drops when increasing r , the distribution security of SEC still keeps the overall security very strong. For instance, if the designer uses a SEC scheme with $r=2$ and $l_k \leq 128$ in a system with $N=10^4$, the upper bound of the encryption security is only 2^{88} . However, when we use a 120-bit key with the parameter $m=8$, the overall number of trials for the active adversary can reach $2^{150.8}$.

The parameters of the SEC scheme can be adaptively configured to satisfy the given requirements of the performance, the security and the redundancy. First, according to (10) SEC determines the upper bound of l_k for the given r and the performance criteria. Second, according to the estimated parameter N of the system, SEC looks for the optimal m and l_k that satisfy both (1) and (2), where the overall security of (9) meets the designer's requirement. If the optimal m is found, the performance, the redundancy and the security are all satisfied by the selected parameters.

When used in stand-alone mode, the vector-matrix multiplication may be attacked by the known-plaintext. To solve this problem, we can simply use the hash value of the secret key and the object identifier as the encryption key for every data object.

Working with Block Cipher

The SEC scheme can also be used together with the conventional block cipher, since the block cipher is known to have a strong encryption security and high performance. For example, the AES with 128-bit key has a throughput of 61MB/s [25]. When using together, the block cipher first encrypts the data object, and then SEC encrypts the encrypted data and distributes the fragments out. As the SEC is mainly responsible for the fragment security and the distribution security, it is not necessary to use a long encryption key length l_k , which will result in a better throughput of the SEC. As a result, the overall performance is still good enough.

7 Related Works

The confidentiality is the essential issue for every Peer-to-Peer storage system. Most of the proposed systems, such as [1] [2] [3] [4] [5] and [6], simply employ the standard block cipher algorithm.

Shamir develops a secret sharing algorithm in [19], where less than a threshold fragments give no information of the data. Unfortunately, each share of the secret is the same size as the secret, so it is not space efficient for the storage system. The IDA [20] algorithm is also a secret sharing algorithm, where each share is smaller than the secret. The IDA can be applied in routing for parallel computers and reliable storage. [21] proposes a combined scheme of XOR secret sharing and replication to eliminate the key management problem in the collaborative environments. Pasis [18] uses the threshold scheme for both confidentiality and durability, where the threshold scheme includes the secret sharing scheme by definition. Unlike the threshold scheme, SEC can work alone as an encryption function because it encrypts data using the user specified secret key. Moreover, with the fragment naming sub-algorithm SEC is de-

signed particularly for the Peer-to-Peer systems because it leverages the large scale distribution feature of Peer-to-Peer system to protect sensitive data.

Rao proposes the idea of joint encryption and error correction schemes in [22] [23]. The proposed scheme keeps the whole generator matrix as the secret key, which makes it infeasible to be employed in a storage system. Xu develops an encryption scheme based on MDS code in [24]. This scheme keeps the indices of the encoded fragments as the secret key, and returns the secret key to user, which results in a large number of keys. In contrast, our SEC generates the generator matrix from a user specified secret key, which eliminates the key management problem.

8 Conclusions

Confidentiality and durability are the essential issues in Peer-to-Peer storage systems. Though the conventional block cipher encryption is widely used to keep the confidentiality, we emphasized that the inherent vulnerability of the Peer-to-Peer storage systems makes it dangerous to only use block cipher. The proposed SEC scheme encrypts data into fragments, and then distributes them out onto different peers. Besides data redundancy, SEC scheme provides a three-layer security, including fragment security, distribution security and encryption security. By a detailed security analysis, we have shown how SEC eliminates the potential security problems of Peer-to-Peer storage systems. Experiments have shown that SEC has a practical throughput. Finally, the SEC can work either in a stand-alone mode or together with conventional block cipher to keep the confidentiality of data.

9 Acknowledgement

The authors wish to thank Yong Liu for the helpful discussions.

References

1. J. Kubiatowicz, D. Bindel, P. Eaton, Y. Chen, D. Geels, R. Gummadi, S. Rhea, W. Weimer, C. Wells, H. Weatherspoon, and B. Zhao. *OceanStore: An architecture for globalscale persistent storage*. ACM SIGPLAN Notices, 35(11):190--201, November 2000.
2. F. Dabek, M. Kaashoek, D. Karger, R. Morris, and I. Stoica. *Wide-area cooperative storage with CFS*. In Proc. of the 18th ACM Symposium on Operating Systems Principles (SOSP), Chateau Lake Louise, Banff, Canada, October 2001.
3. P. Druschel and A. Rowstron, *PAST: A large-scale persistent peer-to-peer storage utility*. In Proc. HOTOS Conf., 2001.
4. I. Clarke, O. Sandberg, B. Wiley and T.W. Hong. *Freenet: A distributed anonymous information storage and retrieval system*. In Proc. of the ICSI Workshop on Design Issues in Anonymity and Unobservability (Berkeley, California, June 2000).

5. A. Muthitacharoen, R. Morris, T. M. Gil, and B. Chen. *Ivy: A read/write peer-to-peer file system*. In Proc. of the 5th Symposium on Operating Systems Design and Implementation (OSDI), Boston, MA, Dec. 2002
6. A. Adya, W. J. Bolosky, M. Castro, G. Cermak, R. Chaiken, J. R. Douceur, J. Howell, J. R. Lorch, M. Theimer, and R. P. Wattenhofer, *FARSITE: Federated, available, and reliable storage for an incompletely trusted environment*. In Proc. 5th Symposium on Operating System Design and Implementation (OSDI), Usenix, 2002.
7. M. Mitzenmacher, *Digital fountains: A survey and look forward*. In Proc. of the IEEE Information Theory Workshop 2004, San Antonio, TX, USA., pp. 271–276, Oct. 2004.
8. J. S. Plank. *A tutorial on Reed-Solomon coding for fault tolerance in RAID-like systems*. Software---Practice and Experience (SPE), 27(9):995--1012, Sept. 1997.
9. W. Peterson, *Error Correcting Codes*, M.I.T. Press and Wiley & Sons, New York, 1961.
10. L. Mirsky, *An Introduction to Linear Algebra*. Dover: Mineola, New York, 1990.
11. J. Bloemer, M. Kalfane, M. Karpinski, R. Karp, M. Luby, and D. Zuckerman. *An XOR-based erasure-resilient coding scheme*. Technical Report TR-95-048, ICSI, Berkeley, California, Aug. 1995
12. L. Zhuang, F. Zhou, B. Y. Zhao, and A. Rowstron. *Cashmere: Resilient anonymous routing*. In Proc. of NSDI 2005.
13. M. J. Freedman and R. Morris. *Tarzan: A Peer-to-Peer Anonymizing Network Layer*. In Proc. of the 9th ACM Conference on Computer and Communications Security (CCS 2002), Washington, D.C., November 2002.
14. F. Schneider. *Implementing Fault-Tolerant Services Using the State Machine Approach: A Tutorial*. ACM Computing Surveys, 22(4), 1990
15. M. Castro and B. Liskov. *Practical Byzantine Fault Tolerance*. In Proc. of the 3rd Symposium on Operating Systems Design and Implementation (OSDI), Feb. 1999.
16. J. Douceur, *The Sybil Attack*. In Proc of the 1st International Workshop on Peer-to-Peer Systems (IPTPS02).
17. E. Karnin, J. Green and M. Hellman, *On secret sharing systems*, IEEE Trans. Information Theory, vol. IT-29, no. 1, pp. 35--41, 1983
18. G. R. Ranger, P. K. Khosla, M. Bakaloglu, M. W. Bigrigg, G. R. Goodson, S. Oguz, V. Pandurangan, C. A. N. Soules, J. D. Strunk, and J. J. Wylie. *Survivable storage systems*. In DARPA Information Survivability Conference and Exposition II, pp 184--195. IEEE Computer Society, June 2001
19. A. Shamir, *How to share a secret*, Communications of the ACM, vol. 22, n.11, pp. 612--613, Nov. 1979
20. M. O. Rabin, *Efficient dispersal of information for security, load balancing, and fault tolerance*, J. Assoc. Comput. Mach., vol. 36, no. 2, pp. 335--348, Apr. 1989
21. A. Subbiah and D. M. Blough. *An approach for fault tolerant and secure data storage in collaborative work environments*. In Proc. of the 2005 ACM Workshop on Storage Security and Survivability, pp. 84--93, Nov. 2005.
22. T.R.N. Rao, *Joint Encryption and Error Correction Schemes*, Proc. 11th Intl. Strnp. on Comp. Arch., Ann Arbor, Mich., May 1984.
23. T.R.N. Rao and K.H. Nam, *Private--key algebraic--code encryptions*, IEEE Transactions on Information Theory, Vol.35, No.4, pp.445--457, 1989.
24. L. Xu, *A General Encryption Scheme based on MDS Codes*, in Proc. of the IEEE International Symposium on Information Theory, June 2003
25. <http://www.eskimo.com/~weidai/benchmarks.html>