A Secure Decentralized Erasure Code for Distributed Networked Storage

Hsiao-Ying Lin, *Student Member*, *IEEE*, and Wen-Guey Tzeng, *Member*, *IEEE*

Abstract—Distributed networked storage systems provide the storage service on the Internet. We address the privacy issue of the distributed networked storage system. It is desired that data stored in the system remain private even if all storage servers in the system are compromised. The major challenge of designing these distributed networked storage systems is to provide a better privacy guarantee while maintaining the distributed structure. To achieve this goal, we introduce secure decentralized erasure code, which combines a threshold public key encryption scheme and a variant of the decentralized erasure code. Our secure distributed networked storage system constructed by the secure decentralized erasure code is decentralized, robust, private, and with low storage cost.

Index Terms—Networked storage system, distributed storage system, decentralized erasure code, threshold encryption, security.

1 Introduction

Storing personal data, such as e-mails and photos, on the Internet has become a common practice. Distributed networked storage systems aim to provide the storage service on the Internet. Current research on distributed networked storage systems focuses on efficiency and robustness of the storage systems, i.e., the methods for accelerating the storing and retrieval processes with minimal cost and maximal robustness. Since the Internet is a public environment that anyone can freely access, it is also important to consider the privacy issue of the stored information of the users.

The purpose of distributed networked storage systems [1], [2], [3] is to store data reliably over a very long period of time by using a distributed accumulation of storage servers. Long-term reliability requires some sort of redundancy. A straightforward solution is simple replication; however, the storage cost for the system is high. Erasure codes are proposed in several designs for reducing the storage overhead in each storage server [4], [5] after linear network codes [6], [7] are proposed. A decentralized erasure code [8] is an erasure code with a fully decentralized encoding process. Assume that there are n storage servers in the networked storage system, and k messages are stored into the storage servers such that one can retrieve the k messages by only querying any k storage servers. The method of erasure codes provides some level of privacy guarantee since the stored data in less than k storage servers are not enough to reveal all information about the k messages. However, it is hard to assure that only less than k storage servers are compromised in an open network. We need a more sophisticated method to protect the data in the storage servers, while the owner of the messages can retrieve them even if only some storage servers respond to the retrieval request.

In this paper, we propose a *secure decentralized erasure code*, which combines the concepts of data encryption and decentralized erasure codes. In this code, the messages are stored in an encrypted form. Even if the attacker compromises all storage servers, he cannot compute information about the content of the k messages. In our storage system, the owner shares his decryption key to a set of key servers in order to mitigate the risk of key leakage. As long as less than t key servers are compromised by the attacker, the decryption key is safe. Furthermore, as long as t key servers get ciphertexts from some storage servers to decrypt, the owner can compute the messages back.

At first glance, we make contrary assumptions on storage servers and key servers. We assume that the key servers are more secure than the storage servers so that it is hard for the attacker to compromise more than t key servers. In cryptography, the security of a system lies on protection of the secret key. Thus, the key servers that hold the secret key are set up or carefully chosen by the owner. Due to their importance, they are highly protected by various system and cryptographic mechanisms, such as proactive cryptographic methods [9], [10]. We can treat storage servers and key servers as two different types of servers. The keys servers are much more secure and their number is much less. The storage servers provide large capacity of storage, while they are prone to attacks.

To maintain the decentralized architecture while applying the data encryption, we present a new threshold public key encryption scheme such that each key server can independently perform the decryption. In traditional threshold public key encryption schemes [11], [12], decrypting a set of ciphertexts requires that each of the key servers decrypts all of the ciphertexts. On the other hand, in our threshold public key encryption scheme, decrypting a set of ciphertexts only requires that each of the key servers decrypts one of the ciphertexts. As a result, the distributed networked storage system constructed by our secure decentralized erasure code is secure and fully decentralized: each encrypted message is distributed independently; each storage server

• *The authors are with the Department of Computer Science, National Chiao Tung University, No. 1001, University Road, Hsinchu 30050, Taiwan. E-mail: hsiaoying.lin@gmail.com, wgtzeng@cs.nctu.edu.tw.*

Manuscript received 27 May 2009; revised 29 Sept. 2009; accepted 6 Jan. 2010; published online 26 Jan. 2010.
Recommended for acceptance by H. Jiang.
For information on obtaining reprints of this article, please send e-mail to: tpds@computer.org, and reference IEEECS Log Number TPDS-2009-05-0239.
Digital Object Identifier no. 10.1109/TPDS.2010.27.

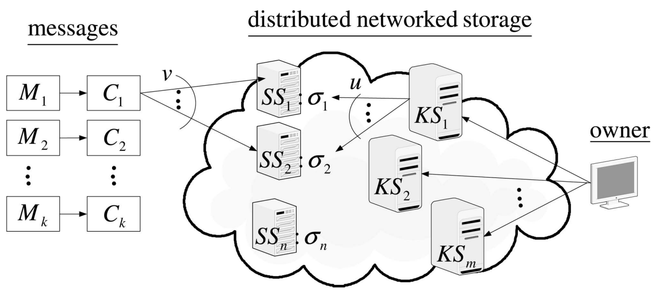

Fig. 1. The model of our distributed networked storage system.

performs the encoding process independently; and each key server executes decryption independently.

The model. Fig. 1 provides an overview of our system. There are k messages M_i, $1 \le i \le k$, to be stored into n storage servers SS_i, $1 \le i \le n$. We could think that these messages are the segments of a file. For those k messages, we assign a message identifier. Each message M_i is encrypted under the owner's public key pk as $C_i = E(pk, M_i)$. Then, each ciphertext is sent to v storage servers, where the storage servers are randomly chosen. Each storage server SS_i combines the received ciphertexts by using the decentralized erasure code to form the stored data σ_i. The owner's secret key sk is shared among m key servers KS_i, $1 \le i \le m$, by a threshold secret sharing scheme so that the key server KS_i holds a secret key share sk_i. To retrieve the k messages, the owner instructs the m key servers such that each key server retrieves stored data from u storage servers and does partial decryption for the retrieved data. Then, the owner collects the partial decryption results, called decryption shares, from the key servers and combines them to recover the k messages.

Our main result shows that for fixed k and $m \ge k > 1$, when $n = ak^{3/2}, v = bk^{1/2} \ln k$, $b > 5a$, $a > \sqrt{2}$, and $u = 2$, the probability that the owner retrieves the k messages is at least $1 - k/p - o(1)$, where p is the size of the group used in our system. The length of p is about 1,000 bit.

Road map. We propose our threshold public key encryption and briefly describe the decentralized erasure code in Section 2. Our construction of secure decentralized erasure code is given in Section 3. In Section 4, we provide a detailed analysis for the probability of successful retrieval of our system in various parameter settings. Conclusion can be found in Section 5.

2 Preliminaries

In this section, we briefly describe bilinear maps and propose our threshold public key encryption using bilinear maps. We also give a short description for the decentralized erasure codes.

2.1 Bilinear Map and Assumptions

Bilinear map. Let $\mathbb{G}_1, \mathbb{G}_2$ be the cyclic multiplicative groups[1] with prime order p and $g \in \mathbb{G}_1$ be a generator. A map $\tilde{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ is a bilinear map if it has the bilinearity and nondegeneracy: for any $x, y \in \mathbb{Z}_p, \tilde{e}(g^x, g^y) = \tilde{e}(g, g)^{xy}$ and $\tilde{e}(g, g)$ is not the identity element in $\mathbb{G}_2$. In fact, $\tilde{e}(g, g)$ is a generator of $\mathbb{G}_2$. Let $\mathrm{Gen}(1^\lambda)$ be an algorithm generating $(p, \mathbb{G}_1, \mathbb{G}_2, \tilde{e}, g)$, where λ is the length of p.

Let $x \in_R X$ denote that x is randomly chosen from the set X.

Bilinear Diffie-Hellman assumption. Following the above parameters, given g, g^x, g^y, g^z, where x, y, and z are randomly chosen from $\mathbb{Z}_p$, the bilinear Diffie-Hellman problem is to find $\tilde{e}(g, g)^{xyz}$. The assumption is that it is hard to solve the problem with a significant probability in polynomial time. Formally, for any probabilistic polynomial-time algorithm $\mathcal{A}$, the following probability is negligible (in λ):

$$\Pr[\mathcal{A}(g, g^x, g^y, g^z) = \tilde{e}(g, g)^{xyz} : x, y, z \in_R \mathbb{Z}_p].$$

Decisional Bilinear Diffie-Hellman assumption. This assumption is that given g, g^x, g^y, g^z, it is hard to distinguish $\tilde{e}(g, g)^{xyz}$ from a random element from $\mathbb{G}_2$. Formally, for any probabilistic polynomial time algorithm $\mathcal{A}$, the following is negligible (in λ):

$$|\Pr[\mathcal{A}(g, g^x, g^y, g^z, \mathbb{Q}_b) = b : x, y, z, r \in_R \mathbb{Z}_p;$$
$$\mathbb{Q}_0 = \tilde{e}(g, g)^{xyz}; \mathbb{Q}_1 = \tilde{e}(g, g)^r; b \in_R \{0, 1\}] - 1/2|.$$

2.2 Threshold Public Key Encryption

A threshold public key encryption consists of six algorithms: SetUp, KeyGen, ShareKeyGen, Enc, ShareDec, and Combine. SetUp generates the public parameters of the whole system, and KeyGen generates a key pair, consisting of a public key pk and a secret key sk, for each user. Each user uses ShareKeyGen to share his secret key into n secret key shares such that any t of them can recover the secret key. Enc encrypts a given message by a public key pk and outputs a ciphertext. ShareDec partially decrypts a given ciphertext by a secret key share and outputs a decryption share. Combine takes a set of decryption shares as input and outputs the message if and only if there are at least t decryption shares.

We propose a threshold public key encryption scheme Π using bilinear maps as follows:

- SetUp(1^λ). To generate μ, run $\mathrm{Gen}(1^\lambda)$ and set $\mu = (p, \mathbb{G}_1, \mathbb{G}_2, \tilde{e}, g)$.
- KeyGen(μ). To generate a key pair for a user, select $x \in_R \mathbb{Z}_p$ and set $pk = g^x$, $sk = x$.
- ShareKeyGen(sk, t, n). The secret key shares $sk_i = f(i)$ are derived by the polynomial $f(\mathrm{z})$, where

  $$f(\mathrm{z}) = sk + a_1\mathrm{z} + a_2\mathrm{z}^2 + \cdots + a_{t-1}\mathrm{z}^{t-1} \pmod p,$$

  and $a_1, a_2, \ldots, a_{t-1} \in_R \mathbb{Z}_p$.
- Enc(pk, M). To generate a ciphertext C of the message $M \in \mathbb{G}_2$, compute

  $$C = (\alpha, \beta, \gamma) = (g^r, h, M\tilde{e}(g^x, h^r)),$$

  where $r \in_R \mathbb{Z}_p$ and $h \in_R \mathbb{G}_1$.
- ShareDec(sk_i, C). Let $C = (\alpha, \beta, \gamma)$. By using the secret key share sk_i, a decryption share ζ_i of C is generated as follows:

  $$\zeta_i = (\alpha_i, \beta_i, \beta'_i, \gamma_i) = (\alpha, \beta, \beta^{sk_i}, \gamma).$$

1. It can also be described as additive groups over points on an elliptic curve.

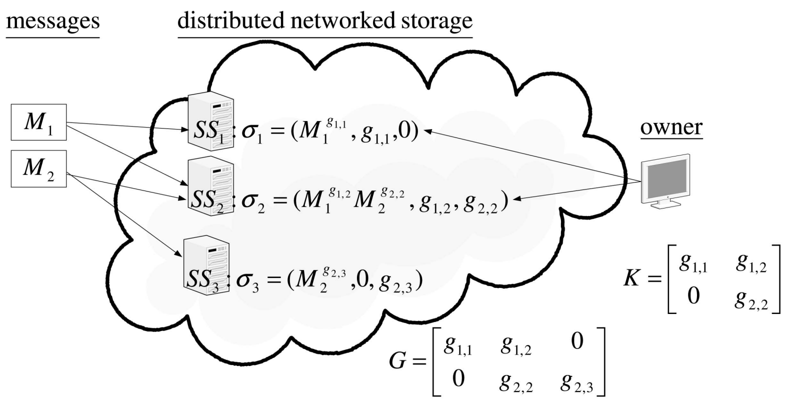
messages
distributed networked storage
owner
M_1
M_2
SS_1 : $\sigma_1 = (M_1^{g_{1,1}}, g_{1,1}, 0)$
SS_2 : $\sigma_2 = (M_1^{g_{1,2}} M_2^{g_{2,2}}, g_{1,2}, g_{2,2})$
SS_3 : $\sigma_3 = (M_2^{g_{2,3}}, 0, g_{2,3})$
$K = \begin{bmatrix} g_{1,1} & g_{1,2} \\ 0 & g_{2,2} \end{bmatrix}$
$G = \begin{bmatrix} g_{1,1} & g_{1,2} & 0 \\ 0 & g_{2,2} & g_{2,3} \end{bmatrix}$

Fig. 2. A storage system using a variant of the decentralized erasure code.

- Combine($\zeta_{i_1}, \zeta_{i_2}, \ldots, \zeta_{i_t}$). It combines the t values $(\beta'_{i_1}, \beta'_{i_2}, \ldots, \beta'_{i_t})$ to obtain $\beta^{sk} = \beta^{f(0)}$ via Lagrange interpolation over exponents:

$$\beta^{sk} = \prod_{i \in \mathrm{S}} \left((\beta'_i)^{\prod_{r \in \mathrm{S}, r \neq i} \frac{-i}{r-i}} \right),$$

where $\mathrm{S} = \{i_1, i_2, \ldots, i_t\}$ and $\zeta_{i_j} = (\alpha_{i_j}, \beta, (\beta)'_{i_j}, \gamma_{i_j})$ for all $1 \leq j \leq t$. The output message is $M = \gamma / \tilde{e}(\alpha, \beta^{f(0)})$.

When a fixed h is used for a set of ciphertexts, the set of those ciphertexts is multiplicative homomorphic. The multiplicative homomorphic property is that given a ciphertext for M_1 and a ciphertext for M_2, a ciphertext for $M_1 \times M_2$ can be generated without knowing the secret key x, M_1, and M_2. Let $C_1 = \mathrm{Enc}(pk, M_1)$ and $C_2 = \mathrm{Enc}(pk, M_2)$, where

$$C_1 = (g^{r_1}, h, M_1 \tilde{e}(g^x, h^{r_1})) \text{ and}$$
$$C_2 = (g^{r_2}, h, M_2 \tilde{e}(g^x, h^{r_2})).$$

A new ciphertext C which is an encryption of $M_1 \times M_2$ under the public key pk is computed as follows:

$$\begin{aligned} C &= (g^{r_1} g^{r_2}, h, M_1 \tilde{e}(g^x, h^{r_1}) M_1 \tilde{e}(g^x, h^{r_2})) \\ &= (g^{r_1 + r_2}, h, M_1 M_2 \tilde{e}(g^x, h^{r_1 + r_2})). \end{aligned}$$

Theorem 1. *The above threshold public key encryption system is chosen plaintext secure (CPA secure) under the decisional bilinear Diffie-Hellman assumption in the standard model.*

2.3 Decentralized Erasure Code

A decentralized erasure code [8] is a random linear code with a sparse generator matrix. Let the message be $\vec{I} = (m_1, m_2, \ldots, m_k)$, the generator matrix $G = [g_{i,j}]_{1 \leq i \leq k, 1 \leq j \leq n}$, and the codeword be $\vec{O} = (w_1, w_2, \ldots, w_n)$. The elements of $\vec{I}$ and $\vec{O}$ and entries of G are all over a finite field $\mathbb{F}$ of size p. The generator matrix G constructed by an encoder is as follows: First, for each row, the encoder randomly marks an entry as 1 and repeats this process for $an \ln k/k$ times with replacement (an entry can be marked multiple times), where a is a constant. Second, the encoder randomly sets a value from $\mathbb{F}$ for each marked entry. The encoding process is expressed as $\vec{I} \cdot G = \vec{O}$. As for the decoding, a decoder receives k columns $j_1, j_2, \ldots, j_k$ of G and the corresponding codeword elements $w_{j_1}, w_{j_2}, \ldots, w_{j_k}$. The decoding process is computed as follows:

$$\begin{aligned} &[m_1, m_2, \ldots, m_k] \\ &= \left[w_{j_1}, w_{j_2}, \ldots, w_{j_k} \right] \begin{bmatrix} g_{1,j_1} & g_{1,j_2} & \cdots & g_{1,j_k} \\ g_{2,j_1} & g_{2,j_2} & \cdots & g_{2,j_k} \\ \cdots & \cdots & \cdots & \cdots \\ g_{k,j_1} & g_{k,j_2} & \cdots & g_{k,j_k} \end{bmatrix}^{-1}. \end{aligned}$$

A decoding is successful if and only if the $k \times k$ submatrix formed by the k-chosen columns is invertible. Thus, the probability of a success decoding is the probability of the chosen submatrix being invertible. It has been shown in [8] that the probability is at least $1 - k/p - o(1)$, where the randomness is introduced by the random choices for marked entries, the random values for marked entries, and the random choices for k columns.

Since the decoder only requires k columns of G and their corresponding codeword elements to decode, this code is resilient to $n - k$ erasure errors. Moreover, the code is decentralized because each codeword element w_i can be independently generated by a different party, when an input vector is given and a generator matrix is marked (but its coefficients are not chosen yet). Consider the following distributed networked storage system, where there are n servers. The owner wants to store k messages M_i, $1 \leq i \leq k$. For each M_i, the owner randomly selects v servers with replacement and sends a copy of M_i to each of them. Each server randomly selects a coefficient for each received ciphertext and performs a linear combination of all received ciphertexts. Those coefficients chosen by a server form a column of the matrix and the result of the linear combination is a codeword element. Because there are n servers, a $k \times n$ generator matrix and a codeword are implicitly formed. Each server can perform the computation independently. This makes the code decentralized.

A variant of the decentralized erasure code. We fix a cyclic multiplicative group $\mathbb{G}$ with prime order p. The message domain is $\mathbb{G}$. The generation of the generator matrix G is the same as the above decentralized erasure code except that the entries of G are over $\mathbb{Z}_p$. The encoding process is to generate $w_1, w_2, \ldots, w_n \in \mathbb{G}$, where $w_i = m_1^{g_{1,i}} m_2^{g_{2,i}} \ldots m_k^{g_{k,i}}$. An example is shown in Fig. 2. Two messages are stored into three storage servers. The first step of the decoding process is to compute the inverse of a $k \times k$ submatrix K of G. Let $K^{-1} = [d_{i,j}]_{1 \le i,j \le k}$. The second step of the decoding process is to compute $m_i = w_{j_1}^{d_{1,i}} w_{j_2}^{d_{2,i}} \dots w_{j_k}^{d_{k,i}}$, where $j_1, j_2, \dots, j_k$ are the indices of columns of K in G. Therefore, a sufficient condition for a success decoding of the variant decentralized erasure code is that the $k \times k$ submatrix K is invertible. Similar to the decentralized erasure code, the probability of a success decoding is at least $1 - k/p - o(1)$.

3 Secure Decentralized Erasure Codes

We assume that there are n storage servers which store data and m key servers which own secret key shares and perform partial decryption. We consider that the owner has the public key $pk = g^x$ and shares the secret key x to m key servers with a threshold t, where $m \ge t \ge k$. Let the k messages be $M_1, M_2, \dots, M_k$. We use $h_{\mathrm{ID}} = H(M_1 \| M_2 \| \cdots \| M_k)$ as the identifier for this set of messages, where $H : \{0,1\}^* \to \mathbb{G}_1$ is a secure hash function.

The storage process and the retrieval process are described in the following:

- *Storage process.* To store k messages, the storage process is as follows:
  - *Message encryption.* The owner encrypts all k messages via the threshold public key encryption Π with the same h_{ID}, where $h_{\mathrm{ID}} = H(M_1 \| M_2 \| \cdots \| M_k)$ is the identifier for the set of messages $M_1, M_2, \dots, M_k$. Let the ciphertext of M_i be
  $$C_i = (\alpha_i, \beta, \gamma_i) = \left(g^{r_i}, h_{\mathrm{ID}}, M_i \tilde{e}\left(g^x, h_{\mathrm{ID}}^{r_i}\right)\right),$$
  where $r_i \in_R \mathbb{Z}_p, 1 \le i \le k$.
  - *Ciphertext distribution.* For each C_i, the owner randomly chooses v storage servers (with replacement) and sends each of them a copy of C_i.
  - *Decentralized encoding.* For all received ciphertexts with the same message identifier h_{ID}, the storage server SS_j groups them as N_j. The storage server SS_j selects a random coefficient $g_{i,j}$ from $\mathbb{Z}_p$ for each $C_i \in N_j$ and sets $g_{i,j} = 0$ for $C_i \notin N_j$. This step forms a generator matrix $G = [g_{i,j}]_{1 \le i \le k, 1 \le j \le n}$ of the decentralized erasure code.

  Each storage server SS_j computes the following (A_j, B_j):
  $$A_j = \prod_{C_i \in N_j} \alpha_i^{g_{i,j}} \text{ and } B_j = \prod_{C_i \in N_j} \gamma_i^{g_{i,j}},$$
  and stores
  $$\sigma_j = (A_j, h_{\mathrm{ID}}, B_j, (g_{1,j}, g_{2,j}, \dots, g_{k,j})).$$
  In fact, $(A_j, h_{\mathrm{ID}}, B_j)$ is a ciphertext for $\prod_{1 \le i \le k} M_i^{g_{i,j}}$ since
  $$\begin{aligned}&(A_j, h_{\mathrm{ID}}, B_j)\\ &= \left(\prod_{C_i \in N_j} (g^{r_i})^{g_{i,j}}, h_{\mathrm{ID}}, \prod_{C_i \in N_j} \left(M_i \tilde{e}\left(g^x, h_{\mathrm{ID}}^{r_i}\right)\right)^{g_{i,j}}\right)\\ &= \left(g^{\prod_{C_i \in N_j} r_i g_{i,j}}, h_{\mathrm{ID}}, \right.\\ &\left.\left(\prod_{C_i \in N_j} M_i^{g_{i,j}}\right) \tilde{e}\left(g^x, h_{\mathrm{ID}}^{\prod_{C_i \in N_j} r_i g_{i,j}}\right)\right)\\ &= \left(g^{\tilde{r}}, h_{\mathrm{ID}}, \left(\prod_{C_i \in N_j} M_i^{g_{i,j}}\right) \tilde{e}\left(g^x, h_{\mathrm{ID}}^{\tilde{r}}\right)\right),\end{aligned}$$
  where $\tilde{r} = \prod_{C_i \in N_j} r_i g_{i,j}$.
- *Retrieval process.* To retrieve k messages, the retrieval process is as follows:
  - *Retrieval command.* The owner sends a command to the m key servers with the message identifier h_{ID}.
  - *Partial decryption.* Each key server KS_i randomly queries u storage servers with the message identifier h_{ID} and obtains at most u stored data σ_j from the storage servers. Then, the key server KS_i performs ShareDec on each received ciphertext by its secret key share sk_i to obtain a decryption share of the ciphertext. Assume that KS_i receives σ_j. KS_i decrypts the ciphertext $(A_j, h_{\mathrm{ID}}, B_j)$ as a decryption share $\zeta_{i,j} = (A_j, h_{\mathrm{ID}}, h_{\mathrm{ID}}^{sk_i}, B_j)$, and sends the following to the owner:
  $$\tilde{\zeta}_{i,j} = \left(A_j, h_{\mathrm{ID}}, h_{\mathrm{ID}}^{sk_i}, B_j, \left(g_{1,j}, g_{2,j}, \dots, g_{k,j}\right)\right).$$
  - *Combining and decoding.* The owner chooses $\tilde{\zeta}_{i_1,j_1}, \tilde{\zeta}_{i_2,j_2}, \dots, \tilde{\zeta}_{i_t,j_t}$ from all received data $\tilde{\zeta}_{i,j}$ and computes $h_{\mathrm{ID}}^{sk} = h_{\mathrm{ID}}^{f(0)} = h_{\mathrm{ID}}^x$ by the Lagrange interpolation over exponents, where $i_1 \ne i_2 \ne \cdots \ne i_t$ and $\mathrm{S} = \{i_1, i_2, \dots, i_t\}$:
  $$h_{\mathrm{ID}}^x = \prod_{i \in \mathrm{S}} \left(h_{\mathrm{ID}}^{sk_i}\right)^{\prod_{r \in \mathrm{S}, r \ne i} \frac{-i}{r-i}}.$$
  If the number of the received $\tilde{\zeta}_{i,j}$ is more than t, the owner randomly selects t out of them. If the number is less than t, the retrieval process fails. After having h_{ID}^x, the owner reconsiders all received data and chooses $\tilde{\zeta}_{i_1,j_1}, \tilde{\zeta}_{i_2,j_2}, \dots, \tilde{\zeta}_{i_k,j_k}$ with $j_1 \ne j_2 \ne \cdots \ne j_k$. By using h_{ID}^x, the owner decrypts $\zeta_{i,j}$ as w_j for all $(i,j) \in \{(i_1, j_1), (i_2, j_2), \dots, (i_k, j_k)\}$:
  $$w_j = \frac{B_j}{\tilde{e}(A_j, h_{\mathrm{ID}}^x)} = \prod_{C_l \in N_j} M_l^{g_{l,j}}.$$
  The owner then computes
  $$K^{-1} = [d_{i,j}]_{1 \le i,j \le k},$$
  where $K = [g_{i,j}]_{1 \le i \le k, j \in \{j_1, j_2, \dots, j_k\}}$. If K is not invertible, the retrieval process fails. Otherwise,

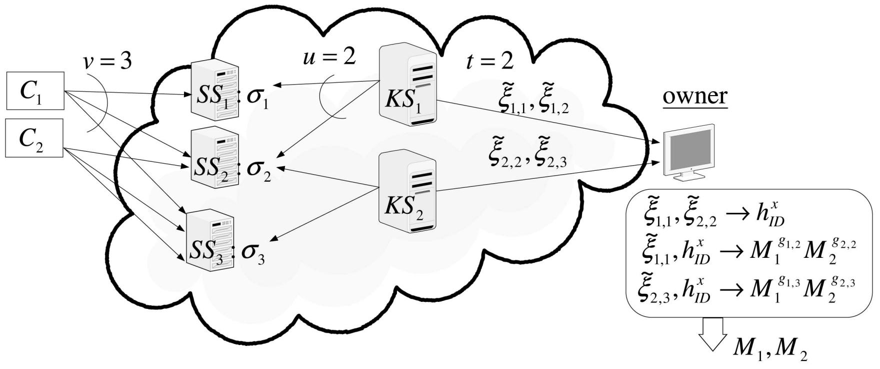
$v=3$
$u=2$
$t=2$
C_1
C_2
$SS_1:\sigma_1$
$SS_2:\sigma_2$
$SS_3:\sigma_3$
KS_1
KS_2
$\tilde{\xi}_{1,1},\tilde{\xi}_{1,2}$
$\tilde{\xi}_{2,2},\tilde{\xi}_{2,3}$
owner
$\tilde{\xi}_{1,1},\tilde{\xi}_{2,2}\to h_{ID}^x$
$\tilde{\xi}_{1,1},h_{ID}^x\to M_1^{g_{1,2}}M_2^{g_{2,2}}$
$\tilde{\xi}_{2,3},h_{ID}^x\to M_1^{g_{1,3}}M_2^{g_{2,3}}$
M_1,M_2

Fig. 3. A storage system using the secure decentralized erasure code.

the owner successfully obtains M_i, $1 \le i \le k$, by the following computation:

$$\begin{aligned} &w_{j_1}^{d_{1,i}} w_{j_2}^{d_{2,i}} \cdots w_{j_k}^{d_{k,i}} \\ &= M_1^{\sum_{l=1}^k g_{1,j_l} d_{l,i}} M_2^{\sum_{l=1}^k g_{2,j_l} d_{l,i}} \dots M_k^{\sum_{l=1}^k g_{k,j_l} d_{l,i}} \\ &= M_1^{\tau_1} M_2^{\tau_2} \dots M_k^{\tau_k} \\ &= M_i, \end{aligned}$$

where $\tau_r = \sum_{l=1}^k g_{r,j_l} d_{l,i} = 1$ if $r = i$ and $\tau_r = 0$ otherwise.

An example is given in Fig. 3. In the ciphertext distribution step, the ciphertext C_1 is distributed to SS_1, SS_2, and SS_3. The ciphertext C_2 is distributed to SS_2 and SS_3 only. After receiving $\tilde{\zeta}_{1,1}$, $\tilde{\zeta}_{1,2}$, $\tilde{\zeta}_{2,2}$, and $\tilde{\zeta}_{2,3}$, the owner computes h_{ID}^x from $\tilde{\zeta}_{1,1}$ and $\tilde{\zeta}_{2,2}$. By using h_{ID}^x, the owner computes the encoded messages, $M_1^{g_{1,2}} M_2^{g_{2,2}}$ and $M_1^{g_{1,3}} M_2^{g_{2,3}}$, and decodes them to get messages M_1 and M_2.

Our design uses two techniques. First, for retrieving messages, the decryption process can be performed *before* the decoding process. Second, the decryption process can be performed by the key servers independently. The first technique comes from the multiplicative homomorphic property of our encryption scheme. For those k messages, a fixed message identifier h_{ID} is used. As a result, the set of ciphertexts is multiplicative homomorphic. An encoding result of ciphertexts $C_1, C_2, \dots, C_k$ is also an encryption of an encoding result of messages $M_1, M_2, \dots, M_k$. As for the second key technique, the design of the encryption scheme embeds the decryption power at the value h_{ID}^x, while h_{ID} is the message identifier. With h_{ID}^x, the owner can decrypt all ciphertexts marked with the message identifier h_{ID}. A key server KS_i can compute a share $h_{\mathrm{ID}}^{sk_i}$ of h_{ID}^x. With at least t key servers, h_{ID}^x can be computed.

4 Analysis

We analyze the computation cost, the storage cost, and the probability of a success retrieval. Let the bit length of the element in the group $\mathbb{G}_1$ be l_1 and $\mathbb{G}_2$ be l_2.

4.1 Computation Cost

We measure the computation cost in the number of pairing operations, modular exponentiations in $\mathbb{G}_1$ and $\mathbb{G}_2$, modular multiplications in $\mathbb{G}_1$ and $\mathbb{G}_2$, and arithmetic operations over $GF(p)$. These operations are denoted as Pairing, Exp_1, Exp_2, Mult_1, Mult_2, and F_p, respectively. We consider the cost for k messages together since the storage process and retrieval process are designed for a set of k messages. The cost is listed in Table 1. In fact, F_p has much lower cost than Mult_1 and Mult_2. One Exp_1 is about $1.5\lceil \log_2 p \rceil \mathrm{Mult}_1$ on average (by using the fast square and

TABLE 1
Computation Cost of Each Step in Our Design

Operations	Computation cost
Message encryption (for k messages)	k Pairing + $2k$ Exp_1 + k Mult_2
Decentralized encoding (for each SS)	k Exp_1 + k Exp_2 + $(k-1)$ Mult_1 + $(k-1)$ Mult_2
Partial decryption (for t KS)	t Exp_1
Combining	k Pairing + k Mult_2 + $\mathrm{O}(t^2)$ F_p
Decoding	k^2 Exp_2 + $(k-1)k$ Mult_2 + $\mathrm{O}(k^3)$ F_p

- *Pairing: a pairing computation of* $\tilde{e}$.
- Exp_1 *and* Exp_2*: a modular exponentiation computation in* $\mathbb{G}_1$ *and* $\mathbb{G}_2$*, respectively.*
- Mult_1 *and* Mult_2*: a modular multiplication computation in* $\mathbb{G}_1$ *and* $\mathbb{G}_2$*, respectively.*
- F_p*: an arithmetic operation in* $GF(p)$.

multiply algorithm). That is, when p is about 1,000 bits, one Exp_1 is about 1,500 Mult_1 on average. Similarly, Exp_2 is about $1.5\lceil\log_2 p\rceil\mathrm{Mult}_2$ on average.

Since, in practice, the coefficients can be chosen from a smaller set, the measure of the computation cost of the Exp_1 and Exp_1 is an overestimation. Pairing is considered as a more expensive operation than Exp. However, some improved algorithms [13], [14] are proposed for accelerating the pairing computation.

In the storage process, for each message encryption, generating α_i requires one Exp_1, and generating γ_i requires one Exp_1, one Pairing, and one Mult_2. Hence, in the message encryption step for k messages, the cost is (k Pairing + $2k\,\mathrm{Exp}_1 + k\,\mathrm{Mult}_2$). In the ciphertext distribution step, no computation occurs. In the encoding step, each SS_i encodes all received messages. Here, we use a worse cast estimation that each SS_i receives k messages. To compute A_i, SS_i requires $k\,\mathrm{Exp}_1$ and $(k-1)\,\mathrm{Mult}_1$ while to compute B_i, the cost is $k\,\mathrm{Exp}_2$ and $(k-1)\,\mathrm{Mult}_2$.

For the partial decryption step, each KS_i performs one Exp_1 to get $h_{\mathrm{ID}}^{sk_i}$. For a success retrieval, t key servers would be sufficient; hence, for this step, we consider the total cost of t key servers. That is, $t\,\mathrm{Exp}_1$. For the combining and decoding step, we split it into two substeps: the combining substep and the decoding substep. The combining substep includes the computation of h_{ID}^{x} and the computation of codeword elements w_js from the decryption shares $\tilde{\zeta}_{i,j}$s. The computation of h_{ID}^{x} is a Lagrange interpolation over exponents in $\mathbb{G}_1$, which requires $O(t^2)\mathrm{F}_p$, $t\,\mathrm{Exp}_1$, and $(t-1)\mathrm{Mult}_1$. Computing w_j from A_j, B_j, and h_{ID}^{x} requires one Pairing and one modular division, which takes 2 Mult_2. The decoding substep includes the matrix inversion and the computation of messages M_is from codeword elements w_js. The matrix inversion takes $O(k^3)$ arithmetic operations over $GF(p)$, and the decoding for each message takes $k\,\mathrm{Exp}_2$ and $(k-1)\mathrm{Mult}_2$.

4.2 Storage Cost

The storage cost in a key server for a user is $\lceil\log_2 p\rceil$ because the key server only requires to store the secret key share. The main storage cost lies on the storage servers.

We measure the storage cost in bits as the average cost in a storage server for a message bit. To store k messages, each storage server SS_j stores $(A_j, h_{\mathrm{ID}}, B_j)$ and the coefficient vector $(g_{1,j}, g_{2,j}, \ldots, g_{k,j})$. The total cost in a storage server is $(2l_1 + l_2 + k\lceil\log_2 p\rceil)$ bits, where $A_j, h_{\mathrm{ID}} \in \mathbb{G}_1$, and $B_j \in \mathbb{G}_2$; hence, the average cost for a message bit is $(2l_1 + l_2 + k\lceil\log_2 p\rceil)/kl_2$ bits, which is dominated by $\lceil\log_2 p\rceil/l_2$ for a sufficient large k. In practicality, $g_{i,j}$s are chosen from a much smaller set than $\mathbb{Z}_p$. Then, we can use fewer bits to represent $g_{i,j}$s. This reduces the storage cost in each storage server.

4.3 Probability of a Success Retrieval

When n and k are fixed, u and v affect the probability of a success retrieval. We investigate the relations of these parameters for the success probability. The results are given in Theorems 2 and 3.

To retrieve all k messages, the key servers have to get k stored data $\sigma_{j_1}, \sigma_{j_2}, \ldots, \sigma_{j_k}$ from k different storage servers $\mathrm{SS}_{j_1}, \mathrm{SS}_{j_2}, \ldots, \mathrm{SS}_{j_k}$ and apply ShareDec to acquire $\tilde{\zeta}_{i_1,j_1}.\tilde{\zeta}_{i_2,j_2}, \ldots, \tilde{\zeta}_{i_k,j_k}$. Furthermore, a $k \times k$ matrix K formed by the coefficient vectors in $\tilde{\zeta}_{i_1,j_1}, \tilde{\zeta}_{i_2,j_2}, \ldots, \tilde{\zeta}_{i_k,j_k}$ needs to be invertible in order to solve the k messages. The random process is on the selection of distinct $\mathrm{SS}_{j_1}, \mathrm{SS}_{j_2}, \ldots, \mathrm{SS}_{j_k}$ by the key servers and the coefficient vectors in $\sigma_{j_1}, \sigma_{j_2}, \ldots,$ and σ_{j_k}. Let E_1 be the event that less than k distinct storage servers are queried by the key servers. For the generator matrix G implicitly generated by the owner and the storage servers, let E_2 be the event that the submatrix K of k columns $j_1, j_2, \ldots, j_k$ of G is noninvertible. Thus, the probability of a success retrieval by the owner is

$$1 - \Pr[\mathrm{E}_1] - \Pr[\mathrm{E}_2|\overline{\mathrm{E}}_1]\Pr[\overline{\mathrm{E}}_1]. \tag{1}$$

We analyze suitable settings of $m, v,$ and u, where $n = ak^{3/2}$ and $n = ak$, respectively, and the results are listed in the following:

1. $n = ak^{3/2}$, $a > \sqrt{2}$, $m \geq t \geq k > 1$, $v = bk^{1/2}\ln k$, $u = 2$ with $b > 5a$,
2. $n = ak$, $a > 1$, $m = t = k > 1$, $v = b_1 \ln k$, $u = b_2 \ln k$ with $b_1 > 5a$ and $b_2 > 4 + 3/\ln a$.

We image a networked storage system that consists of a large number of storage servers. The number k of stored messages each time is much less than n. Thus, the first setting of $n = ak^{3/2}$ is better than the second setting of $n = ak$. Although, in the regular coding theory, the constant information rate for the second setting may be preferred, the first setting is more suitable for the application to practical networked storage systems.

Theorem 2. *Assume that there are k messages, n storage servers, and m key servers where $n = ak^{3/2}$, $m \geq t \geq k > 1$ and a is a constant with $a > \sqrt{2}$. For $v = bk^{1/2}\ln k$ and $u = 2$ with $b > 5a$, the probability of a success retrieval is at least $1 - k/p - o(1)$.*

Proof. To analyze $\Pr[\mathrm{E}_1]$, we consider that each storage server is a bin and each key server has u balls, where $u = 2$. When a key server queries a storage server, we consider that the key server throws a ball into the bin. Because the key servers make queries randomly, those balls are randomly thrown into n bins. The probability that less than k bins contain balls is

$$\begin{aligned}
\Pr[\mathrm{E}_1] &\leq C_{k-1}^{n}\left(\frac{k-1}{n}\right)^{2m} \\
&= \frac{n(n-k+2)}{(k-1)1}\frac{(n-1)(n-k+3)}{(k-2)2} \\
&\quad \cdots \frac{(n-\lfloor\frac{k-2}{2}\rfloor)(n-\lceil\frac{k-2}{2}\rceil)}{\lceil\frac{k-1}{2}\rceil\lfloor\frac{k-1}{2}\rfloor}\left(\frac{k-1}{n}\right)^{2m} \\
&\leq \frac{2n(n-k+2)^{\frac{k-1}{2}}}{k}\left(\frac{k-1}{n}\right)^{2m} \qquad (2)\\
&\leq \left(2a^2k^2 - 2ak^{3/2} + 4ak^{1/2}\right)^{\frac{k-1}{2}}\left(\frac{k-1}{n}\right)^{2k} \\
&\qquad\qquad (\text{because } n = ak^{3/2}) \\
&\leq \left(\frac{2a^2k^2 - 2ak^{3/2} + 4ak^{1/2}}{a^4k^2}\right)^{\frac{k}{2}}\left(\frac{k-1}{k}\right)^{2k} \\
&= o(1) \qquad\qquad (\text{because } a > \sqrt{2}).
\end{aligned}$$

The event E_2 under the condition $\overline{\mathrm{E}}_1$ can be modeled by forming a perfect matching in the random bipartite graph H with respect to G. The random bipartite graph H is constructed as follows: Let each ciphertext C_i be a vertex $v_{1,i}$ and V_1 be the set of all vertices for C_is. Let each storage server SS_j be a vertex $v_{2,j}$ and V_2 be the set of all vertices for SS_js. When a ciphertext C_i is distributed to the storage server SS_j, there is an edge $(v_{1,i}, v_{2,j})$. The matrix K induces a subgraph H' of the bipartite graph H. The subgraph H' consists of all vertices in V_1, a subset $V_2' \subset V_2$ that V_2' is a subset of queried storage servers and $|V_2'| = k$, and edges $(v_{1,i}, v_{2,j})$ for all $v_{1,i} \in V_1$ and $v_{2,j} \in V_2'$. If H' has no perfect matching, K is not invertible. If H' has a perfect matching, K is noninvertible if and only if $det(K) = 0$. The value of $det(K)$ depends on the random coefficients chosen by the storage servers. Let E_3 be the event that H' has no perfect matching, and E_4 be the event that $det(K) = 0$. We have

$$\begin{aligned}\Pr[\mathrm{E}_2|\overline{\mathrm{E}}_1] &= \Pr[\mathrm{E}_3|\overline{\mathrm{E}}_1] + \Pr[\mathrm{E}_4|\overline{\mathrm{E}}_3 \wedge \overline{\mathrm{E}}_1]\Pr[\overline{\mathrm{E}}_3|\overline{\mathrm{E}}_1] \\ &\le \Pr[\mathrm{E}_3|\overline{\mathrm{E}}_1] + \Pr[\mathrm{E}_4|\overline{\mathrm{E}}_3 \wedge \overline{\mathrm{E}}_1].\end{aligned} \tag{3}$$

We analyze the probability of E_3 conditioned on $\overline{\mathrm{E}}_1$ by using the Hall's Lemma in the following form [8]:

Lemma 1 (Hall's Lemma). *Let H' be a bipartite graph with vertex sets V_1 and V_2', where $|V_1| = |V_2'| = k$. If H' has no isolated vertex and no perfect matching, then there exists a set* $\mathrm{A} \subset V_1$ or $\mathrm{A} \subset V_2$ *such that:*

- $2 \le |A| \le \frac{k+1}{2}$.
- *The number of neighbors of* A *is* $|\mathrm{A}| - 1$.
- *The subgraph induced by* A *and its neighbors is connected.*

Hence, there are two cases that H' has no perfect matching. First, H' has at least one isolated vertex. Second, H' has no isolated vertex and a set A satisfies the above conditions. Let E_I be the event that H' has at least one isolated vertex and E_A be the event that some set A satisfies the conditions. We obtain

$$\Pr[\mathrm{E}_3|\overline{\mathrm{E}}_1] \le \Pr[\mathrm{E}_\mathrm{I}|\overline{\mathrm{E}}_1] + \Pr[\mathrm{E}_\mathrm{A}|\overline{\mathrm{E}}_1]. \tag{4}$$

Starting from E_I, we consider each vertex in V_2 as a bin and each edge from V_1 to V_2 as a ball. When an edge connects to a vertex in V_2, a ball is thrown into the bin. Consider the subset B of the bins corresponding to the subset V_2' of V_2. Thus, B contains k bins. E_I means that there is one or more empty bins in B. For a fixed bin in B, the probability of the bin being empty is $(1 - 1/n)^{bk^{3/2}\ln k}$ since there are $bk^{3/2}\ln k$ balls. By using the union bound on k bins, we have the probability of E_I conditioned on $\overline{\mathrm{E}}_1$ as

$$\begin{aligned}\Pr[\mathrm{E}_\mathrm{I}|\overline{\mathrm{E}}_1] &\le k(1-1/n)^{bk^{3/2}\ln k} \\ &= k\left(1 - \frac{1}{ak^{3/2}}\right)^{ak^{3/2}\cdot\frac{b}{a}\ln k} && (\text{because } n = ak^{3/2}) \\ &\le k(e^{-\frac{b}{a}\ln k}) && (\text{because } 1 - x \le e^{-x}) \\ &= \left(\frac{1}{k}\right)^{\frac{b}{a}-1} \\ &= o(1) && (\text{because } b > 5a).\end{aligned} \tag{5}$$

As for $\Pr[\mathrm{E}_\mathrm{A}|\overline{\mathrm{E}}_1]$, we separate the event into two subevents by $\mathrm{A} \subset V_1$ or $\mathrm{A} \subset V_2'$. Thus,

$$\begin{aligned}\Pr[\mathrm{E}_\mathrm{A}|\overline{\mathrm{E}}_1] &= \Pr[\mathrm{E}_\mathrm{A} \text{ and } \mathrm{A} \subset V_1|\overline{\mathrm{E}}_1]\Pr[\mathrm{A} \subset V_1] \\ &\quad + \Pr[\mathrm{E}_\mathrm{A} \text{ and } \mathrm{A} \subset V_2'|\overline{\mathrm{E}}_1]\Pr[\mathrm{A} \subset V_2'] \\ &\le \Pr[\mathrm{E}_\mathrm{A} \text{ and } \mathrm{A} \subset V_1|\overline{\mathrm{E}}_1] \\ &\quad + \Pr[\mathrm{E}_\mathrm{A} \text{ and } \mathrm{A} \subset V_2'|\overline{\mathrm{E}}_1].\end{aligned}$$

For $\Pr[\mathrm{E}_\mathrm{A} \text{ and } \mathrm{A} \subset V_2'|\overline{\mathrm{E}}_1]$, we further divide the event into subevents according to the size of A and use the union bound again. Consider a set $A \subset V_2'$ with $|A| = i$. The event E_A conditioned on $\overline{\mathrm{E}}_1$ can be overestimated by the event that $\Gamma(A) \subset V_1$ and $|\Gamma(A)| = i - 1$. In other words, there is a set $A' \subset V_1$ with $|A'| = i - 1$ such that all vertices in $V_1 \backslash A'$ only connect to vertices in $V_2 \backslash A$. Thus, we have

$$\begin{aligned}&\Pr[\mathrm{E}_\mathrm{A} \text{ and } \mathrm{A} \subset V_2'|\overline{\mathrm{E}}_1] \\ &\le \sum_{i=2}^{(k+1)/2} \Pr[\mathrm{E}_\mathrm{A}, \mathrm{A} \subset V_2' \text{ and } |A| = i|\overline{\mathrm{E}}_1] \\ &= \sum_{i=2}^{(k+1)/(2)} C_i^k C_{i-1}^k \left(\frac{n-i}{n}\right)^{(k-i+1)bk^{1/2}\ln k} \\ &\le \sum_{i=2}^{(k+1)/(2)} \left(\frac{ek}{i}\right)^{2i} \left(\frac{n-i}{n}\right)^{(k-i+1)bk^{1/2}\ln k} \\ &\qquad\qquad \left(\text{because } C_i^k \le \left(\frac{ek}{i}\right)^i\right) \\ &\le k \max_i \left\{ \left(\frac{ek}{i}\right)^{2i} \left(\frac{n-i}{n}\right)^{(k-i+1)bk^{1/2}\ln k} \right\} \\ &= \max_i \left\{ \exp\left(2i(1 - \ln i) \right.\right. \\ &\quad \left.\left. + \ln k\left[b(k-i+1)k^{1/2}\ln\left(\frac{n-i}{n}\right) + 2i + 1\right]\right)\right\}.\end{aligned} \tag{6}$$

To achieve $\Pr[\mathrm{E}_\mathrm{A} \text{ and } \mathrm{A} \subset V_2'|\overline{\mathrm{E}}_1] = o(1)$ as $k \to \infty$, it is sufficient to have

$$b(k-i+1)k^{1/2}\ln\left(\frac{n-i}{n}\right) + 2i + 1 < 0. \tag{7}$$

Since $(n-i)/n = 1 - i/n < e^{-(i/n)}$, we have

$$b(k-i+1)k^{1/2}\left(\frac{-i}{n}\right) + 2i + 1 < 0. \tag{8}$$

By (8), we need

$$b > \frac{(2i+1)ak^{3/2}}{(k-i+1)k^{1/2}i} = \frac{(2i+1)ak}{(k-i+1)i}. \tag{9}$$

Since $b > 5a$, for $2 \le i \le (k+1)/2$, (9) holds. It implies that

$$\Pr[\mathrm{E}_\mathrm{A} \text{ and } \mathrm{A} \subset V_2'|\overline{\mathrm{E}}_1] = o(1),$$

as $k \to \infty$. Similarly, we can get a lower bound for b from the case of $A \subset V_1$ and the bound is satisfied by $b > 5a$.

For $\Pr[\mathrm{E}_4 \mid \overline{\mathrm{E}}_3 \wedge \overline{\mathrm{E}}_1]$, that is, $det(A) = 0$, we treat each coefficient, randomly chosen from Z_p, in the matrix K as a variable. Thus, $det(K)$ is a multivariate function. Since there is a perfect matching in the induced graph H', $det(K)$ is a nonzero function and the degree of $det(K)$ is k. From the Schwartz-Zippel Theorem, the probability that the randomly chosen coefficients make $det(K) = 0$ is no more than k/p, i.e., $\Pr[\mathrm{E}_4 \mid \overline{\mathrm{E}}_3 \wedge \overline{\mathrm{E}}_1] \le k/p$. Thus, we have

$$\Pr[\mathrm{E}_2 \mid \overline{\mathrm{E}}_1] \le k/p + o(1),$$

and conclude the proof of Theorem 2. □

For another setting for v and u, where $n = ak$ and $m = k$, we have the following theorem:

Theorem 3. *Assume that there are k messages, n storage servers, and m key servers, where $n = ak$ for a fixed constant $a > 1$ and $m = t = k > 1$. For $v = b_1 \ln k$, $u = b_2 \ln k$, $b_1 > 5a$, and $b_2 > 4 + 3/\ln a$, the probability of a success retrieval is at least $1 - k/p - o(1)$, where p is the size of the used group.*

Proof. By the proof of Theorem 2, we analyze two events E_1 and E_2 similarly. We have

$$\Pr[\mathrm{E}_2] < k/p + o(1).$$

To bound E_1, we start with

$$\Pr[\mathrm{E}_1] \le C^n_{k-1}\left(\frac{k-1}{n}\right)^{b_2 k \ln k}.$$

By the bound for C^n_{k-1} in the proof of Theorem 2 and $n = ak$, we obtain

$$\begin{aligned}\Pr[\mathrm{E}_1] &\le \left(\frac{2n(n-k+2)}{k-1}\right)^{\frac{k+1}{2}}\left(\frac{k-1}{n}\right)^{b_2 k \ln k}\\ &\le (4a^2k)^{\frac{k+1}{2}}\left(\frac{k}{ak}\right)^{b_2 k \ln k}\\ &= a^{k+1+[\log_a(4k)]\frac{k+1}{2} - b_2 k \ln k}\\ &= o(1) \qquad \left(\text{because } b_2 > 4 + \frac{3}{\ln a}, k > 1\right),\end{aligned}$$

as $k \to \infty$. Therefore, the probability of a success retrieval is

$$1 - \Pr[\mathrm{E}_1] - \Pr[\mathrm{E}_2 \mid \overline{\mathrm{E}}_1]\Pr[\overline{\mathrm{E}}_1] \ge 1 - k/p - o(1).$$

□

5 Conclusion

We have introduced a secure decentralized erasure code and our secure distributed networked storage system. Our system provides both of the storage service and the key management service. Our construction is fully decentralized: each encrypted message is distributed independently; each storage server performs the encoding process in a decentralized way; each key server queries the storage servers independently. Moreover, the secure distributed networked storage system guarantees the privacy of messages even if all storage servers are compromised. Our storage system securely stores data for a long period of time on untrusted storage servers in the distributed network structure.

Appendix

We extend the standard CPA security game for the threshold public key encryption scheme, and then, prove that our threshold public key encryption scheme is CPA secure.

A. Definition

The threshold CPA security game consists of a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$.

- *Setup:* $\mathcal{C}$ does the following:
  - Run Setup(λ) to get $\mu = (p, \mathbb{G}_1, \mathbb{G}_2, \tilde{e}, g)$.
  - Run KeyGen(μ) to get a key pair (pk, sk) and run ShareKeyGen on (sk, t, n) to get $sk_i, 1 \le i \le n$, where t and n are randomly chosen.
  - Send (μ, pk, t, n) to $\mathcal{A}$.
- *Key share query:* $\mathcal{A}$ queries $(t-1)$ secret key shares from $\mathcal{C}$ and gets $sk_{q_1}, sk_{q_2}, \ldots, sk_{q_{t-1}}$, where $q_1, q_2, \ldots, q_{t-1} \in [1, n]$.
- *Challenge:* $\mathcal{A}$ chooses two messages M_0 and M_1, where $M_0 \ne M_1$, and sends them to $\mathcal{C}$. $\mathcal{C}$ encrypts M_b as C, where b is randomly selected from $\{0,1\}$, and sends C to $\mathcal{A}$.
- *Output:* $\mathcal{A}$ outputs a bit b' for guessing b.

The advantage of $\mathcal{A}$ is defined as $Adv_{\mathcal{A}} = |\Pr[b' = b] - 1/2|$. A threshold public key encryption scheme is CPA secure if and only if for any probabilistic polynomial-time algorithm $\mathcal{A}$, $Adv_{\mathcal{A}}$ is a negligible function in λ.

B. Proof of Theorem 1

We prove that if no probabilistic polynomial time algorithm solves the decisional bilinear Diffie-Hellman problem with advantage ϵ, then no polynomial time algorithm wins the CPA security game against our encryption scheme with advantage 2ϵ.

Proof. We prove by contradiction. Assume that there is an algorithm $\mathcal{A}$ winning the CPA security game against our encryption scheme with advantage 2ϵ. We can construct an algorithm $\mathcal{A}'$ solving the decisional bilinear Diffie-Hellman problem with advantage ϵ as follows:

- *Setup.* The input of $\mathcal{A}'$ is $(g, g^x, g^y, g^z, \mathbb{Q})$ with public parameters $(\tilde{e}, \mathbb{G}_1, \mathbb{G}_2, p)$. Then, $\mathcal{A}'$ sends (μ, pk, t, n) to $\mathcal{A}$, where $\mu = (p, \mathbb{G}_1, \mathbb{G}_2, \tilde{e}, g)$, $pk = g^x$, t is a threshold value, and n is the number of secret key shares. This implicitly sets $sk = x$.
- *Key share query.* To answer $\mathcal{A}$'s queries $q_1, q_2, \ldots, q_{t-1}$ for $(t-1)$ secret key shares, $\mathcal{A}'$ sets $sk_{q_1}, sk_{q_2}, \ldots, sk_{q_{t-1}}$ as random values and sends them to $\mathcal{A}$. Wlog, assume that $q_1, q_2, \ldots, q_{t-1}$ are all different.
- *Challenge.* $\mathcal{A}$ gives two messages M_0 and M_1. $\mathcal{A}'$ randomly selects $b \in \{0,1\}$ and encrypts M_b as

$$C = \mathrm{Enc}(pk, M_b) = (g^y, g^z, M_b\mathbb{Q}).$$

- *Output.* $\mathcal{A}'$ sends C to $\mathcal{A}$ and gets $\mathcal{A}$'s output b'. If $b' = b$, then $\mathcal{A}'$ outputs 0 for guessing that $\mathbb{Q} = \mathbb{Q}_0 = \tilde{e}(g,g)^{xyz}$. If $b' \ne b$, then $\mathcal{A}'$ outputs 1 for guessing that $\mathbb{Q} = \mathbb{Q}_1 = \tilde{e}(g,g)^r$.

When $\mathbb{Q} = \mathbb{Q}_0 = \tilde{e}(g,g)^{xyz}$, C is a ciphertext of M_b; thus, $\mathcal{A}$ has advantage 2ϵ winning the game, i.e., $\Pr[b' = b \mid \mathbb{Q} = \tilde{e}(g,g)^{xyz}] = 1/2 + 2\epsilon$. When $\mathbb{Q} = \mathbb{Q}_1 = \tilde{e}(g,g)^r$ for some random r, the distributions of $(g^y, g^z, M_0\mathbb{Q})$ and $(g^y, g^z, M_1\mathbb{Q})$ are identical because for any r, there exists r' such that $M_0\tilde{e}(g,g)^r = M_1\tilde{e}(g,g)^{r'}$. Thus, we have $\Pr[b' = b|\mathbb{Q} = \tilde{e}(g,g)^r] = 1/2$. The advantage of $\mathcal{A}'$ is

$$\begin{aligned} &\Big| \Pr[\mathcal{A}' \to 0 \mid \mathbb{Q} = \mathbb{Q}_0] \Pr[\mathbb{Q} = \mathbb{Q}_0] \\ &\quad + \Pr[\mathcal{A}' \to 1 \mid \mathbb{Q} = \mathbb{Q}_1] \Pr[\mathbb{Q} = \mathbb{Q}_1] - \frac{1}{2} \Big| \\ &= \left| \left(\frac{1}{2} + 2\epsilon\right) \times \frac{1}{2} + \frac{1}{2} \times \frac{1}{2} - 1/2 \right| \\ &= \epsilon. \end{aligned}$$

□

ACKNOWLEDGMENTS

This research was supported in part by projects NSC-98-2221-E-009-068-MY3, NSC-98-2218-E-009-020, and TL-99G109.

REFERENCES

[1] J. Kubiatowicz, D. Bindel, Y. Chen, S.E. Czerwinski, P.R. Eaton, D. Geels, R. Gummadi, S.C. Rhea, H. Weatherspoon, W. Weimer, C. Wells, and B.Y. Zhao, "Oceanstore: An Architecture for Global-Scale Persistent Storage," *Proc. Ninth Int'l Conf. Architectural Support for Programming Languages and Operating Systems (ASPLOS),* vol. 35, pp. 190-201, 2000.

[2] S.C. Rhea, C. Wells, P.R. Eaton, D. Geels, B.Y. Zhao, H. Weatherspoon, and J. Kubiatowicz, "Maintenance-Free Global Data Storage," *IEEE Internet Computing,* vol. 5, no. 5, pp. 40-49, Sept. 2001.

[3] F. Dabek, M.F. Kaashoek, D. Karger, R. Morris, and I. Stoica, "Wide Area Cooperative Storage with cfs," *Proc. 18th Symp. Operating Systems Principles (SOSP),* pp. 202-215, 2001.

[4] S. Acedański, S. Deb, M. Médard, and R. Keettor, "How Good Is Random Linear Coding Based Distributed Networked Storage," *Proc. First Workshop Network Coding, Theory, and Applications—NetCod,* 2005.

[5] C. Gkantsidis and P. Rodriguez, "Network Coding for Large Scale Content Distribution," *Proc. IEEE INFOCOM,* vol. 4, pp. 2235-2245, 2005.

[6] R. Ahlswede, N. Cai, S.-Y.R. Li, and R.W. Yeung, "Network Information Flow," *IEEE Trans. Information Theory,* vol. 46, no. 4, pp. 1204-1216, 2000.

[7] S.-Y.R. Li, R.W. Yeung, and N. Cai, "Linear Network Coding," *IEEE Trans. Information Theory,* vol. 49, no. 2, pp. 371-381, Feb. 2003.

[8] A.G. Dimakis, V. Prabhakaran, and K. Ramchandran, "Decentralized Erasure Codes for Distributed Networked Storage," *IEEE Trans. Information Theory,* vol. 52, no. 6, pp. 2809-2816, June 2006.

[9] A. Herzberg, S. Jarecki, H. Krawczyk, and M. Yung, "Proactive Secret Sharing or: How to Cope with Perpetual Leakage," *Proc. 15th Ann. Int'l Cryptology Conf.—CRYPTO,* pp. 339-352, 1995.

[10] C. Cachin, K. Kursawe, A. Lysyanskaya, and R. Strobl, "Asynchronous Verifiable Secret Sharing and Proactive Cryptosystems," *Proc. Ninth ACM Conf. Computer and Comm. Security (CCS),* pp. 88-97, 2002.

[11] R. Canetti and S. Goldwasser, "An Efficient ıthreshold Public Key Cryptosystem Secure against Adaptive Chosen Ciphertext Attack," *Proc. Int'l Conf. Theory and Application of Cryptographic Techniques,* pp. 90-106, 1999.

[12] D. Boneh, X. Boyen, and S. Halevi, "Chosen Ciphertext Secure Public Key Threshold Encryption without Random Oracles," *Proc. Topics in Cryptology (CT-RSA),* pp. 226-243, 2006.

[13] P.S.L.M. Barreto, B. Lynn, and M. Scott, "Efficient Implementation of Pairing-Based Cryptosystems," *J. Cryptology,* vol. 17, no. 4, pp. 321-334, 2004.

[14] V.S. Miller, "The Weil Pairing, and Its Efficient Calculation," *J. Cryptology,* vol. 17, no. 4, pp. 235-261, 2004.

Hsiao-Ying Lin received the BS degree in computer science and engineering from the National Sun Yat-Sen University in 2003, and the MS degree in computer science in 2005 from the National Chiao Tung University, where she is currently working toward the PhD degree at the Department of Computer Science. Her current research interests include applied cryptography and information security. She is a student member of the IEEE.

Wen-Guey Tzeng received the BS degree in computer science and information engineering from the National Taiwan University, Taiwan, 1985, and the MS and PhD degrees in computer science from the State University of New York at Stony Brook, in 1987 and 1991, respectively. He joined the Department of Computer and Information Science (now, Department of Computer Science), National Chiao Tung University, Taiwan, in 1991. He now serves as a chairman of the department. His current research interests include Cryptology, Information Security, and Network Security. He is a member of the IEEE.