Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X

IJCSMC, Vol. 2, Issue. 11, November 2013, pg.348 – 355

RESEARCH ARTICLE

A Protected Erasure-Code Based Cloud Repository System

R. Deepika Sagar[1], T. Madhumathi[2]

PG Scholar, Department Software and Engineering, TKR College of Engineering and Technology, Hyderabad, A.P-500 097, India
Email: ramarapudeepika@gmail.com

Assistant Professor, Department Information Technology and Engineering, TKR College of Engineering and Technology, Hyderabad, A.P-500 097, India

Abstract—A cloud storage system, consisting of a collection of storage servers, provides long term storage services over the Internet. Storing data in a third party's cloud system causes serious concern over data confidentiality. General encryption schemes protect data Privacy, but also limit the functionality of the storage system because a few operations are supported over encrypted data. Constructing a protected storage system that supports multiple functions is challenging when the storage system is distributed and has no central authority. a threshold proxy encryption scheme and integrate it with a decentralized erasure code such that a protected distributed storage system is formulated. The distributed storage system not only supports protected and robust data storage and retrieval, but also lets a user forward his data in the storage servers to another user without retrieving the data back. The Major technical contribution is that the proxy encryption scheme supports encoding operations over encrypted messages as well as forwarding operations over encoded and encrypted messages. This method fully integrates encrypting, encoding, and forwarding .Here Examine and suggests suitable Arguments for the number of copies of a message dispatched to storage servers and the number of storage servers queried by a key server.

Keywords:-Decentralized erasure code, proxy encryption, threshold cryptography, protected storage system.

1 Introduction

As high-speed networks and ubiquitous Internet access become available in recent years, many services are provided on the Internet such that users can use them from anywhere at any time. For example, the email service is probably the most popular one. Cloud computing is a concept that treats the resources on the Internet as a unified entity, a cloud. Users just use services without being concerned about how computation is done and storage is managed. Here focus on designing a cloud storage system for robustness, confidentiality, and fun-tonality. A cloud storage system is considered as a large scale distributed storage system that consists of many independent storage servers. Data robustness is a major requirement for storage systems. There have been many proposals of storing data over storage servers 1, 2,3,4,5, one way to provide data robustness is to replicate a message such that each storage server stores a copy of the message. It is very robust because the message can be retrieved as long as one storage server survives. Another way is to encode a message of k symbols into a codeword of n symbols by erasure coding. To store a message, each of its codeword symbols is stored in a different storage server. A storage server failure corresponds H.-Y. To an erasure error of the codeword symbol. As long as the number of failure servers is under the tolerance threshold of the erasure code, the message can be recovered from the codeword symbols stored in the available storage servers by the decoding process. This provides a tradeoff between the storage size and the tolerance threshold of failure servers. A decentralized erasure code is an erasure code that independently computes each codeword symbol for a message. Thus, the encoding process for a message can be split into *n* parallel tasks of generating codeword symbols. A decentralized erasure code is suitable for use in a distributed storage system. After the message symbols are sent to storage servers, each storage server independently computes a codeword symbol for the received message symbols and stores it. This finishes the encoding and storing process. The recovery process is the same. Storing data in a third party's cloud system causes serious concern on data confidentiality. In order to provide strong confidentiality for messages in storage servers, a user can encrypt messages by a cryptographic method before applying an erasure code method to encode and store messages. When he wants to use a message, he needs to retrieve the codeword symbols from storage servers, decode them, and then decrypt them by using cryptographic keys. There

Storage servers cannot directly forward a user's messages to another one. The owner of messages has to retrieve, decode, decrypt and then forward them to another user. Here the problem of forwarding data to another user by storage servers directly. considering the system model that consists of distributed storage servers and key servers. Since storing cryptographic keys in a single device is risky, a user distributes his cryptographic key to key servers that shall perform cryptographic functions on behalf of the user. These key servers are highly protected by security mechanic-isms. To well fit the distributed structure of systems, require that servers independently perform all operations. With this consideration, a new threshold surrogate encryption scheme and integrate it with a protected decentralized code to form a protected distributed storage system. The encryption scheme supports encoding operations over encrypted messages and forwarding operations over encrypted and encoded messages. The tight integration of encoding, encryption, and forwarding makes the storage system efficiently meet the requirements of data robustness, data confidentiality, and data forwarding. Accomplishing the integration with consideration of distributed structure is challenging. Our system meets the requirements that storage servers independently perform encoding and encryption and key servers independently perform partial decryption. Moreover, consider the system.
This setting allows more flexible adjustment between the number of storage servers and robustness. Our contributions. Assume that there are n distributed storage servers and m key servers in the cloud storage system. A message is divided into k blocks and represented as a vector of k symbols. Our contributions are as follows: Construct a protected cloud storage system that supports the function of protected data forwarding by using a threshold proxy encryption scheme. The encryption scheme supports decentralized erasure codes over encrypted messages and forwarding operations over encrypted and encoded messages. System is highly distributed where storage servers independently encode and forward messages and key servers independently perform partial ecryption. We present a general setting for the Arguments of protected cloud storage system. Our parameter setting ofn = ak^c supersedes the previous one of n = ak\/k,where c > 1.5 and a > \/2 [6]. Our result n = ak^c allows the number of storage servers be much greater than the number of blocks of a message. In practical systems, the number of storage servers is much more than k. The sacrifice is to slightly increase the total copies of an encrypted message symbol sent to storage servers. Nevertheless, the storage size in each storage server does not increase because each storage server stores an encoded result which is a combination of encrypted message symbols.

2.1 Distributed Storage Systems

At the early years, the Network-Attached Storage and the Network File System provide extra storage devices over the network such that a user can access the storage devices via network connection. Afterward, many improvements on scalability, robustness, efficiency, and security were proposed A decentralized architecture for storage systems offers good scalability, because a storage server can join or leave without control of a central authority. To provide robust-ness against server failures, a simple method is to make replicas of each message and store them in different servers. However, this method is expensive as z replicas result in times of expansion. One way to reduce the expansion rate is to use erasure codes to encode messages . A messages encoded as a codeword, which is a vector of symbols, and each storage server stores a codeword symbol. A storage server failure is modeled as an erasure error of the stored codeword symbol. Random linear codes support distributed encoding, that is, each codeword symbol is independently computed. To store a message of k blocks, each storage server linearly combines the blocks with randomly chosen coefficients and stores the codeword symbol and coefficients. To retrieve the message, a user queries k storage servers for the stored codeword symbols and coefficients and solves the linear system. Dimakis et al. Considered the case that n = ak for a fixed constant a. They showed that distributing each block of a message to v randomly chosen storage servers is enough to have a probability 1 — k/p — o(1) of a successful data retrieval, where v = b ln k, b > 5 a, and p is the order of the used group. The sparsity parameter v = b ln k is the number of storage servers which a block is sent to. The larger v is, the communication cost is higher and the successful retrieval probability is higher. The system has a light data confidentiality because an attacker can compromise k storage servers issues by presenting a protected decentralized erasure code for the networked storage system. In addition to storage servers, their system consists of key servers, which hold cryptographic key shares and work in a distributed way. In their system, stored messages are encrypted and then encoded. To retrieve a message, key servers query storage servers for the user. As long as the number of available key servers is over a threshold t, the message can be successfully retrieved with an overwhelming probability. One of their results shows that when there are n storage servers with n = ak\/k, the parameter v is bVk l n k with b > 5a, and each key server queries 2 storage servers for each retrieval request, the probability of a successful retrieval is at least 1 — k/p — o(1).

2.2 Proxy Encryption Schemes.

In a proxy rencryption.scheme, a proxy server can transfer a cipher text under a public key PK_A to a new one under another public key PK_B by using the re-encryption key $RK_{A!B}$. The server does not know the plaintext during transformation. Ateniese et al. proposed some proxyre-encryption schemes and applied them to the sharing systems. In their work, messages are first encrypted by the owner and then stored in a storage server. When a user protected cloud storage

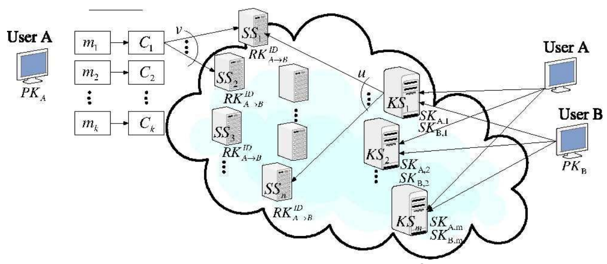

Fig. 1. A general system model of our work.

wants to share his messages, he sends a re-encryption key to the storage server. The storage server re-encrypts the encrypted messages for the authorized user. Thus, their system has data confidentiality and supports the data forwarding function. Our work further integrates encryption, re-encryption, and encoding such that storage robust-ness is strengthened. Type-based proxy re-encryption schemes proposed by Tang [17] provide a better granularity on the granted right of are-encryption key. A user can decide which type of messages and with whom he wants to share in this kind of proxy re-encryption schemes. Key-private proxy re-encryption schemes are proposed by Ateniese et al. [18]. In a key-private proxy re-encryption scheme, given a re-encryption key, a proxy server cannot determine the identity of the recipient. This kind of proxy re-encryption schemes provides higher privacy guarantee against proxy servers. Although most proxy re-encryption schemes use pairing operations, there exist proxy re-encryption schemes without pairing [19].

2.3 Integrity Checking

Functionality Another important functionality about cloud storage is the function of integrity checking. After a user stores data into the storage system, he no longer possesses the data at hand. The user may want to check whether the statistics are properly stored in storage servers.

3. Development

3.1 System Model

As shown in Fig. 1, our system model consists of users, n storage servers SSi, SS2,SS_n, and m key servers KSi, KS_2,, KS_m. Storage servers provide storage services and key servers provide key management services. They work independently. Our distributed storage system consists of four phases: system setup, data storage, data forwarding, and data retrieval. These four phases are described as follows. In the system setup phase, the system manager chooses system Arguments and publishes them. Each user A is assigned a public-secret key pair (PK_A, SK_A). User A distributes his secret key SK_A to key servers such that each key server KS_i holds a key share $SK_{A,S}$, 1 < i < m. The key is shared with a threshold t. In the data storage phase, user A encrypts his message M and dispatches it to storage servers. A message M is decomposed into k blocks m_1,m_2, ...,m_k and has an identifier ID. User A encrypts each block mi into a ciphertext C_i and sends it to v randomly chosen storage servers. Upon receiving cipher texts from a user, each storage server linearly combines them with randomly chosen coefficients into a codeword symbol and stores it. Note that a storage server may receive less than k message blocks and we assume that all storage servers know the value k in advance. In the data forwarding phase, user A forwards his encrypted message with an identifier ID stored in storage servers to user B such that B can decrypt the forwarded message by his secret key. To do so, A uses his secret key SK_A and B's public key PK_B to compute a re-encryption key $RKA©_B$ and then sends $RKA\^{}_B$ to all storage servers. Each storage server uses the re-encryption key to re-encrypt its codeword symbol for later retrieval requests by B. The re-encrypted codeword symbol isthe combination of ciphertexts under B's public key. In order to distinguish re-encrypted codeword symbols from in tactones, call them original codeword symbols and re-encrypted codeword symbols, respectively. In the data retrieval phase, user A requests to retrieve a message from storage servers. The message is either stored by him or forwarded to him. User A sends a retrieval request to key servers. Upon receiving the retrieval request and executing a proper authentication process with user A, each key server KS_i requests u randomly chosen storage servers to get codeword symbols and does partial decryption on the received codeword symbols by using the key share $SKA_{;i}$. Finally, user A combines the partially decrypted codeword symbols to obtain the original message M.

System recovering.

When a storage server fails, a new one is added. The new storage server queries k available storage servers, linearly combines the received codeword symbols asa new one and stores it. The system is then recovered.

3.2 Peril Model

Data confidentiality for both data storage and data forwarding. In this threat model, an attacker wants to break data confidentiality of a target user. To do so, the attacker colludes with all storage servers, nontarget users, and up to (t — 1) key servers. The attacker analyzes stored messages in storage servers, the secret keys of nontarget users, and the shared keys stored in key servers. Note thatthe storage servers store all re-encryption keys provided by users. The attacker may try to generate a new re-encryption key from stored re-encryption keys. We formally model this attack by the standard chosen plaintext attack[1] of the surrogate1. Systems against chosen ciphertext attacks are more protected than systems against the chosen plaintext attack. Here, we only consider the chosen plaintext attack because a homomorphic encryption re-encryption scheme in a threshold version, as shown in the challenger C provides the system Arguments. After the attacker A chooses a target user T, the challenger gives him (t — 1) key shares of the secret key SKT of the target user T to model (t — 1) compromised key servers. Then, the attacker can query secret keys of other users and all re-encryption keys except those from T to other users. This models compromised nontarget users and storage servers. In the challenge phase, the attacker chooses two messages..MQ and M_1 with the identifiers IDo and ID_1, respectively. The challenger throws a random coin b and encrypts the message M_b with T's public key PKT . After getting the cipher text from the challenger, the attacker outputs a bit b'for guessing b. In this game, the attacker wins if and only ifb' = b. The advantage of the attacker is defined as 11/2 — Pr[b' = b]|.A cloud storage system modeled in the above is protected if no probabilistic polynomial time attacker wins the game with a nonnegligible advantage. A protected cloud storage system implies that an unauthorized user or server cannot get the content of stored messages, and a storage server cannot generate re-encryption keys by himself. If a storage server can generate a re-encryption key from the target user to another user B, the attacker can win the security game by re-encrypting the ciphertext to B and decrypting the re-encrypted ciphertext using the secret key SK_B. Therefore, this model addresses the security of data storage and data forwarding.

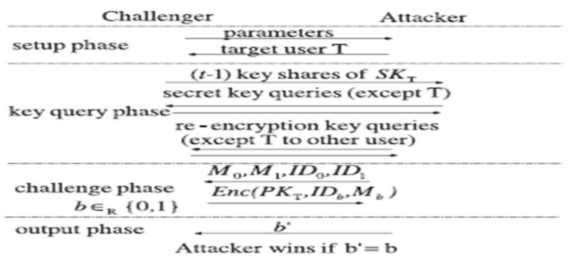

Fig. 2. The security game for the chosen plaintext attack.

3.3 A Clear Cut Solution

A straightforward solution to supporting the data forwarding function in a distributed storage system is as follows: when the owner A wants to forward a message to user B, he downloads the encrypted message and decrypts it by using his secret key. He then encrypts the message by using B's public key and uploads the new cipher text. When B wants to retrieve the forwarded message from A, he downloads the cipher text and decrypts it by his secret key. The whole data forwarding process needs three communication rounds for A's downloading and uploading and B's downloading. The communication cost is linear in the length of the forwarded message. The computation cost is the decryption and encryption for the owner A, and the decryption for user B. Proxy re encryption schemes can significantly decrease communication and computation cost of the owner. In a proxy re-encryption scheme, the owner sends a re-encryption key to storage servers such that storage servers perform there-encryption operation for him. Thus, the communication cost of the owner is independent of the length of forwarded message and the computation cost of re-encryption is taken care of by storage servers. Proxy re-encryption schemes. significantly reduce the overhead of the data forwarding function in a protected storage system.

4 Constructions of Protected Cloud Storage Systems

Before presenting our storage system, we briefly introduce the algebraic setting, the hardness assumption, an erasure code over exponents, and our approach. **Bilinear map.** Let **(G_1** and **(G_2** be cyclic multiplicativegroups^2 with a prime order p and g 2 **G** be a generator. Amap ~ : **(G_1** x **(G_1** ! **(G_2** is a bilinear map if it is efficiently computable and has the properties of bilinearity and nondegeneracy: for any x, y 2 **'Z*$_p$,** ~(g^x, g^y) = ~(g, g)^{xy} and~(g,g) is not the identity element in **(G_2.** Let Gen(1^A) be an algorithm generating (g, e, **<G_1, G**$_2$,p), where A is the length of p. Let x 2R X denote that x is randomly chosen from the set X.

Decisional bilinear Diffie-Hellman assumption.
This assumption is that it is computationally infeasible to distinguish the distributions (g, g^x, g^y, g^z, ~(g,g)^{xyz}) and (g, g^x, g^y, g^z, e(g, g)^r), where x,y,z,r 2R r**Z**p. Formally, for any probabilistic polynomial time algorithm A, the following is
negligible (in A):$^{/}$Pr[A(g;gx, gV; gZ; Q) = b: x, y, z r 2r
Q0 = e(g,g)xyz;Q1 = e(g,g)r;b 2r f0,1g] — 1/2/

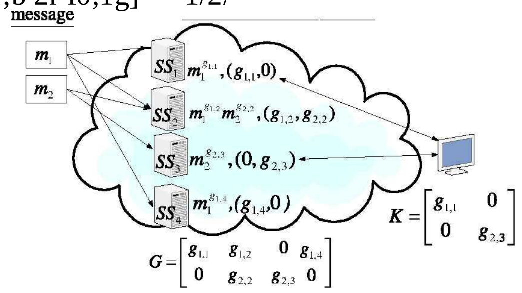
message
m_1
m_2
SS_1 $m_1^{g_{1,1}}, (g_{1,1}, 0)$
SS_2 $m_1^{g_{1,2}} m_2^{g_{2,2}}, (g_{1,2}, g_{2,2})$
SS_3 $m_2^{g_{2,3}}, (0, g_{2,3})$
SS_4 $m_1^{g_{1,4}}, (g_{1,4}, 0)$
$K = \begin{bmatrix} g_{1,1} & 0 \\ 0 & g_{2,3} \end{bmatrix}$
$G = \begin{bmatrix} g_{1,1} & g_{1,2} & 0 & g_{1,4} \\ 0 & g_{2,2} & g_{2,3} & 0 \end{bmatrix}$

Fig. 3. A storage system with random linear coding over exponents.

4.1 Erasure coding over exponents.

The user a supports a group operation on encrypted plaintexts without decryption D(SK, E(PK, *mi)* © E(PK, *m_2))* = *mi* • *m_2*,where E is the encryption function, D is the decryption .function, and (PK, SK) is a pair of public key and secret key. Given two coefficients g_i and g_2, two message symbols. m_i and m_2 can be encoded to a codeword symbol mf^1 mf in the encrypted form C = E(PK,m_i)^{gi} © E(PK,m_2)^{s2} = E(PK,mf • mf*)*.Thus, a multiplicative homomorphic encryption scheme supports the encoding operation over encrypted messages. We then convert a proxy re-encryption scheme with multiplicative homomorphic property into a threshold version. A secret key is shared to key servers with a threshold value t via the Shamir secret sharing scheme [26], where t > k. In our system, to decrypt for a set of k message symbols, each key server independently queries 2 storage servers and partially decrypts two encrypted codeword symbols. As long as t key servers are available, k codeword symbols are obtained from the partially decrypted ciphertexts.

4.1 A Protected Cloud Storage System with ProtectedForwarding

System setup.
The algorithm SetUp(i^T) generates the system Arguments g. A user uses Key Gen(g) to generate his public and secret key pair and ShareKeyGen(-) to share his secret key to a set of m key servers with a threshold t,where k < t < m. The user locally stores the third component of his secret key. Setup(i^A). Run Gen(i^A) to obtain (g,h,e, **<G_i, G**$_2$,p), where e : **(G_i** x **(G_i** ! **(G_2** is a bilinear map, g and h.are generators of **G**i , and both **G**i and **G**2 have the prime order p. Set g = (g, h, e, **G_i, G**$_2$,p, f), where f :%*$_p$ x{0, l}* ! T**Z** is a one-way hash function.
KeyGen(g). For a user A, the algorithm selects.^{a}1^{,a}2, ^{a}3 ■PKA = (g^{ai} ,h^{a2}), SKA = (ai,a2,a3).Share Key Gen(SKA, t, m). This algorithm shares thesecret key SKA of a user A to a set of m key servers. by using two polynomials /A,$_1$(Z) and fA,$_2$(z) ofdegree (t _ l) over the finite field GF(p)fA,i(z) = ai + viz + V2Z^2 H h vt_iz$^{t\ i}$(mod p),fA,2 (z) = a_^{f} + Viz + V2Z^2 H h Vt_iZ$^{t\_i}$(mod p),where v_i,v_2,..., v_{t_i} /**Z**p. The key share of the secret key SK_A to the key server KS^ is SK_{Ai} =(fA,i(i),fA,2(i)), where i < i < m..
Data storage. When user A wants to store a message of blocks m_i,m_2,... ,m_k with the identifier ID, he computes the identity token T = $h^{f\ (a3,ID)}$ and performs the encryption algorithm Enc(-) on T and k blocks to get k original ciphertexts C_i,C_2,... ,C_k. An original cipher text is indicated by a leading bit b = 0. User A sends each cipher text $C_i$to v randomly chosen storage servers. A storage server receives a set of original cipher texts with the same identity token T from A. When a ciphertext C_i is not received, the storage server inserts C_i = (0, 1,T, i) to the set. The special format of (0, 1,T, 1) is a mark for the absence of C_i. Thestorage server performs Encode(-) on the set of k ciphertexts and stores the encoded result (codeword symbol).Enc(PKA, T, m_1,m_2,..., m_k). For 1 < i < k, this algorithm

computes^{C}i = $^{(0,}$ Oi$^{,\wedge,}$ji$^{)}$ = (^{0,}g^{ri,T,m}i$^{e(}$g^{ai} ,T^{ri} ^where r$_i$ 2R ^{r}**Z*$_p$,** 1 < i < k and 0 is the leading bit indicating an original ciphertext. Encode(C$_1$,C$_2$, ...,C$_k$). For each ciphertext C$_i$, the algorithm randomly selects a coefficient g$_i$. If someciphertext C$_i$ is (0,1, T, 1), the coefficient g$_i$ is set to 0.Let C$_i$ = (0,a$_i$,P,j$_i$). The encoding process is tocompute an original codeword symbol C^{c} '= ^{0,} nw ^ & *iK^f.*
= ^0, ^<=1^{9ITI}, T, Y mf e (g^{ai}, T)^<-1 ^{giri}^j
= (^{0,}g$^{r',T,We(}$g$^{,T)alr),}$
where W ,=$_i$ mg and r' = j=$_i$ g^i. The en-coded result is (C',gi,g2,..., g$_k$).

Data forwarding

User A wants to forward a message to another user B. He needs the first component ai of his secret key. If A does not possess a$_f$, he queries key servers for key shares. When at least t key servers respond, A recovers the first component ai of the secret key SKA via theKeyRecover( ) algorithm. Let the identifier of the message.be ID. User A computes the re-encryption key RKA^$_B$ viathe ReKeyGen( ) algorithm and protectedly sends the re-encryption key to each storage server. By using RKA^$_B$, a storage server re-encrypts the original codeword symbol C^{0}with the identifier ID into a re-encrypted codeword symbol C" via the ReEnc(-) algorithm such that C" is decryptable by using B's secret key. A re-encrypted codeword symbol is indicated by the leading bit b = l. Let the public key PK$_B$ of user B be (g^{bl},h^{b2}).

• KeyRecover

(SKA,$_n$, SKA,,2 ,..., SKA$_a$).

$$a_1 = \sum_{s \in \mathrm{T}} \left(f_{\mathrm{A},1}(s) \prod_{s' \in \mathrm{T}/\{s\}} \frac{-s'}{s-s'} \right) \bmod p.$$

ReKeyGen($\mathrm{PK_A}, \mathrm{SK_A}, \mathrm{ID}, \mathrm{PK_B}$). This algorithm se-
ects $e \in_R \mathbb{Z}_p^*$ and computes

$$\mathrm{RK}_{\mathrm{A}\to\mathrm{B}}^{\mathrm{ID}} = ((h^{b_2})^{a_1(f(a_3,\mathrm{ID})+e)}, h^{a_1 e}).$$

ReEnc($\mathrm{RK}_{\mathrm{A}\to\mathrm{B}}^{\mathrm{ID}}, C'$). Let $C' = (0, \alpha, \beta, \gamma) = (0, g^{r'}, \tau,$
$W\tilde{e}(g^{a_1}, \tau^{r'}))$ for some r' and some W, and $\mathrm{RK}_{\mathrm{A}\to\mathrm{B}}^{\mathrm{ID}} =$
$h^{b_2 a_1(f(a_3,\mathrm{ID})+e)}, h^{a_1 e})$ for some e. The re-encrypted
codeword symbol is computed as follows:

$$\begin{aligned} C'' &= (1, \alpha, h^{b_2 a_1(f(a_3,\mathrm{ID})+e)}, \gamma \cdot \tilde{e}(\alpha, h^{a_1 e})) \\ &= (1, g^{r'}, h^{b_2 a_1(f(a_3,\mathrm{ID})+e)}, W\tilde{e}(g,h)^{a_1 r'(f(a_3,\mathrm{ID})+e)}). \end{aligned}$$

5. Examination

Storage cost : To store a message of k blocks, a storage server SS$_j$ stores a codeword symbol ðb; _j; _; _jÞ and the coefficient vector ðg$_{1;j}$; g$_{2;j}$; . . . ; g$_{k;j}$Þ. They are total of ð1 þ2l$_1$ þ l$_2$ þ kl$_3$Þ bits, where _j; _ 2 GG$_1$ and _j 2 GG$_2$. The average cost for a message bit stored in a storage server isð1 þ 2l$_1$ þ l$_2$ þ kl$_3$Þ=kl$_2$ bits, which is dominated by l$_3$=l$_2$ for a sufficiently large k. In practice, small coefficients, i.e.,l$_3$ l$_2$, reduce the storage cost in each storage server.
Computation cost. We measure the computation cost by the number of pairing operations, modular exponentiations in GG$_1$ and GG$_2$, modular multiplications in GG$_1$ and GG$_2$, and arithmetic operations over GFðpÞ. These operations are denoted as Pairing, Exp$_1$, Exp$_2$, Mult$_1$, Mult$_2$, and F$_p$, respectively. The cost is summarized in Table 1. Computing an F$_p$ takes much less time than computing a Mult$_1$ or a

Operation	Computation cost
[illegible]	[illegible]
Encode (for each storage server)	k Expi + k Exp_2 + (k - 1) Multi + (k - 1) $Mult_2$
KeyRecover	0(F) *F_p*
ReKeyGen	1 Expi
ReEnc (for each storage server)	1 Pairing +1 $Mult_2$
ShareDec (for t key servers)	t Expi
Combine - Pairing: a pairing con	k Pairing + t Multi + (t - *1) Expi+0(t*^{2} + k^3) *Fp* + k^2 *Exp_2* + (k + *1*)k *$Mult_2$* *lputation of e.*

TABLE 1
The Computation Cost of Each Algorithm in Our Protected Cloud Storage System

Expi and EXP2: a modular exponentiation computation in Gi and G2, respectively. Multi and Mult2: a modular multiplication computation in Gi and G2, respectively.F_p: an arithmetic operation in **GF(p).**$Mult_2$. The time of computing an Exp_x is 1.5[logp] times as much as the time of computing a $Mult_{1r}$ on average, a Exp_1, a Pairing, and a $Mult_2$. Hence, for k blocks of a message, the cost is (k Pairing + 2k Exp_1 + k $Mult_2$). For the Encode(-)algorithm, each storage server encodes k cipher texts at most. The cost is k Exp_1 + (k — 1) $Mult_1$ for computing a and k Exp_2 + (k — 1) $Mult_2$ for computing 7.In the data forwarding phase, a user runs Key Recover()and Rekeyed( ) and each storage server performs.ReEnc(-). In the Key Recover() algorithm, the computation. cost is O(t^2) F_p. In the Rekeyed() algorithm, the computation cost is a Exp_1. In the Rend(-) algorithm, the computation cost is a Pairing and a $Mult_1$.In the data retrieval phase, each key server runs the Shared(-) algorithm and the user performs the Combine(-) algorithm. In the Shared(-) algorithm, each key server performs a Exp_1 to get f^{skb} for a codeword symbol. For a successful retrieval, t key servers would be sufficient; hence, for this step, the total cost of t key servers is t Exp_1. In the Combine(-) algorithm, it needs the computation of the Lagrange interpolation over exponents. in **<G_1,** the computation of the encoded blocks wjs from the partially decrypted codeword symbols G./s, and the decoding computation which needs to perform the matrix inversion and recovery of blocks m_i's from the encoded blocks wjs. The Lagrange interpolation over exponents in **(G_1** needs O(t^2) F_p, t Exp_1, and (t — 1) $Mult_1$. Computing an encoded block wj needs one Pairing and one modular division, which takes 2 $Mult_2$. As for the decoding. Computation, the matrix inversion takes O(k^3) arithmetic operations over GF(p), and the decoding for each block takes k Exp_2 and (k — 1) $Mult_2$.

Correctness. There are two cases for correctness. The owner A correctly retrieves his message and user B correctly retrieves a message forwarded to him. The correctness of encryption and decryption for A can be seen in (1). The correctness of re-encryption and decryption for B can be seen in (2). As long as at least k storage servers are available, a user can retrieve data with an overwhelming probability. Thus, our storage system tolerates n — k server failures.

Theorem 1. Assume that there are k blocks of a message, nstorage servers, and m key servers, where n = ak^c, m > *t* > k,c > *1.5* and a is a constant with a > \[2. For v = $bk^{c—1}$ *ln k*and u = *2* with b > 5a, the probability of a successful retrieval is at least *1* — k=p — *o(1).*

Security. The data confidentiality of our cloud storage system is guaranteed even if all storage servers, nontarget users, and up to (t — 1) key servers are compromised by the attacker. Recall the security game illustrated in Fig. 2. The proof for Theorem 2 is provided in Appendix B, available in the online supplementary material.

Theorem 2. Our cloud storage system described in Section 4.1 is protected under the threat model in Section 3.2 if the decisional bilinear Diffie-Hellman assumption holds.

6. Conclusion

A cloud storage system consists of storage servers and key servers. We integrate a newly proposed threshold proxy re-encryption scheme and erasure codes over exponents. The threshold proxy re-encryption scheme supports encoding, forwarding, and partial decryption operations in a distributed way. To decrypt a message of k blocks that are encrypted and encoded to n codeword symbols, each key server only has to partially decrypt two code word symbols .By using the threshold proxy encryption scheme; A protected cloud storage system that provides protected data storage and protected data forwarding functionality in a decentralized structure. Moreover, each storage server independently performs encoding and re-encryption and each key server independently perform partial decryption. This storage system and some newly proposed content. Addressable file systems and storage system .are highly compatible. Our storage servers act as storage nodes in a content addressable storage system for storing content addressable blocks.

References

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

2)P. Druschel and A. Rowstron, "**PAST: A Large-Scale, PersistentPeer-to-Peer Storage Utility**," *Proc. Eighth Workshop Hot Topics in Operating System (HotOS VIII),* pp. 75-80, 2001.

3).A. Adya, W.J. Bolosky, M. Castro, G. Cermak, R. Chaiken, J.R.Douceur, J. Howell, J.R. Lorch, M. Theimer, and R. Wattenhofer,"**Farsite: Federated, Available, and Reliable Storage for an Incompletely** Trusted Environment," *Proc. Fifth Symp. Operating System Design and Implementation (OSDI),* pp. 1-14, 2002.

4) A. Haeberlen, A. Mislove, and P. Druschel, "**Glacier: Highly Durable, Decentralized Storage Despite Massive Correlated Fail- ures,**" *Proc. Second Symp. Networked Systems Design and Implemen-tation (NSDI),* pp. 143-158, 2005.

5).Z. Wilcox-O'Hearn and B. Warner, "**Tahoe: The Least-AuthorityFilesystem**," *Proc. Fourth ACM Int'l Workshop Storage Security and Survivability (StorageSS),* pp. 21-26, 2008.

6)H.-Y. Lin and W.-G. Tzeng, "A **Protected Decentralized Erasure Code for Distributed Network Storage**," *IEEE Trans. Parallel and Distributed Systems,* vol. 21, no. 11, pp. 1586-1594, Nov. 2010.

7).D.R. Brownbridge, L.F. Marshall, and B. Randell, "**The Newcastle Connection or Unixes of the World Unite**!," *Software Practice andExperience,* vol. 12, no. 12, pp. 1147-1162, 1982.

8).R. Sandberg, D. Goldberg, S. Kleiman, D. Walsh, and B. Lyon,"**Design and Implementation of the Sun Network** Filesystem,"*Proc. USENIX Assoc. Conf.,* 1985.

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

10)M. Mambo and E. Okamoto, "**ProxyCryptosystems: Delegation of the Power to Decrypt Ciphertexts**," *IEICE Trans. Fundamentals of Electronics, Comm. and Computer Sciences,* vol. E80-A, no. 1, pp. 54-63, 1997.

11)M. Blaze, G. Bleumer, and M. Strauss, "**Divertible Protocols and Atomic ProxyCryptography**," *Proc. Int'l Conf. Theory and Applica-tion of Cryptographic Techniques (EUROCRYPT),* pp. 127-144, 1998.

12)G. Ateniese, K. Fu, M. Green, and S. Hohenberger, "**ImprovedProxyRe-Encryption Schemes with Applications to ProtectedDistributed Storage**," *ACM Trans. Information and System Security,*
vol. 9, no. 1, pp. 1-30, 2006.

13)Q. Tang, "**Type-Based ProxyRe-Encryption and Its Construction,"*Proc. Ninth Int'l*** *Conf. Cryptology in India: Progress in Cryptology(INDOCRYPT),* pp. 130-144, 2008.

14)G. Ateniese, K. Benson, and S. Hohenberger, "**Key-Private SurrogateRe-Encryption**," *Proc. Topics in Cryptology (CT-RSA),* pp. 279-294,2009.

Authors:

1. R. Deepika Sagar, PG Scholar, Department of Computer Science and Engineering, TKR College of Engineering and Technology Hyderabad, A.P-500 097, India

2. T. Madhumathi, Assistant Professor, Department of Computer Science and Engineering, TKR College of Engineering and Technology Hyderabad, A.P-500 097, India