A Protected Cloud Storage System with Safe Information Forwarding

C. Kavinilavu,
Computer Science and Engineering, PRIST University, Trichy.

Abstract

In cloud storage system provide several storage services over the internet. Data's are stored in third party's cloud storage system. It causes over data confidentiality. Encryption schemes to protect the data confidentiality but also the functionality to be limited in storage system because few operations only supported over encrypted data. In this paper, propose a threshold proxy re-encryption scheme and integrate a erasure code such that distributed storage system. In this storage system not only support secure and robust data it also support forwarding data from one user to another user without retrieving the data .In proposed system support encoding operation as well as forwarding over encoded and encrypted messages. In this system support encryption, encoding and ffpforwarding operations. .To analyzes and suggests suitable parameters for the number of copies of a message dispatched to storage servers and the number of storage servers queried by a key server. These parameters allow more flexible adjustment between the number of storage servers and robustness.

Index Terms—Decentralized erasure code, proxy re-encryption, threshold cryptography, secure storage system.

1. Introduction

Data robustness is a major requirement for storage systems. There have been many proposals of storing data over storage servers. 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 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. 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 are three problems in the above straightforward integration of encryption and encoding. First, the user has to do most computation and the communication traffic between the user and storage servers is high. Second, the user has to manage his cryptographic keys. If the user's device of storing the keys is lost or compromised, the security is broken. Finally, besides data storing and retrieving, it is hard for storage servers to directly support other functions. For example, 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. In this paper, address the problem of forwarding data to another user by storage servers directly under the command of the data owner. With this consideration, to propose a new threshold proxy re-encryption scheme and integrate it with a secure decentralized code to form a secure 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 a distributed structure is challenging. Our system meets the requirements that storage servers independently perform Encoding and re-encryption and key servers independently perform partial decryption. Moreover, we consider the system

in a more general setting than previous works. This setting allows more flexible adjustment between the number of storage servers and robustness.

2. Related Work

Briefly review distributed storage systems, proxy re-encryption schemes, and integrity checking mechanisms. [1] 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 robustness 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 z times of expansion. The server does not know the plaintext during transformation. 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. [2] Proposed some proxy re-encryption schemes and applied them to the sharing function of secure storage systems. In their work, messages are first encrypted by the owner and then stored in a storage server. When a user 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 robustness is strengthened. A user can decide which type of messages and with whom he wants to share in this kind of proxy re-encryption schemes. 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 [4]. 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 data are properly stored in storage servers. The concept of provable data possession [3], later, public auditability of stored data is addressed in [5]. Nevertheless all of them consider the messages in the clear text form.

3. System Model

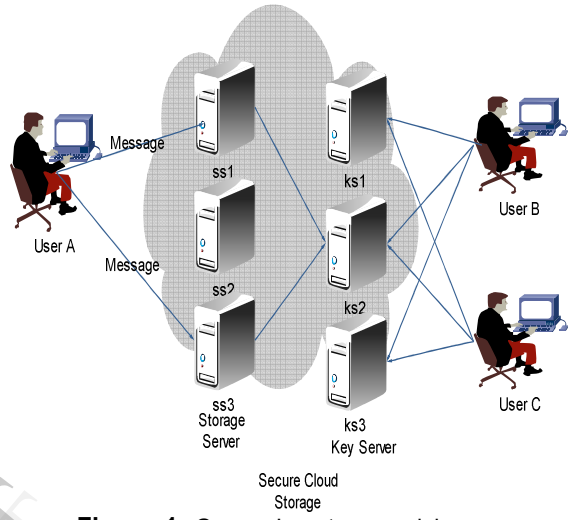

Figure 1. General system model

As shown in Fig. 1,our system model consists of users, n storage servers SS_1 ; SS_2 ; . . . ; SS_n , and m key servers KS_1 ; 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.

3.1. System Setup

In the system setup phase, the system manager chooses system parameters and publishes them. Each user A is assigned a public-secret key pair (PK_A; SK_A) and ShareKeyGen (). User A distributes his secret key SK_A to key servers such that each key server KS_i holds a key share SK_A; and the public key is kept by user.

3.2. Data Storage

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 m_i into a ciphertext C_i and sends it to v randomly chosen storage

servers. Upon receiving ciphertexts 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 assume that all storage servers know the value k in advance.

3.3. Proxy Re-encryption Scheme

The proxy re-encryption scheme supports encoding, forwarding, and partial decryption operations in a distributed way. By using the threshold proxy re-encryption scheme, we present a secure cloud storage system that provides secure data storage and secure data forwarding functionality in a decentralized structure. Messages are first encrypted by the owner and then stored in a storage server. When a user 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. Proxy re-encryption schemes can significantly decrease communication and computation cost of the owner. Proxy re-encryption schemes significantly reduce the overhead of the data forwarding function in a secure storage system.

3.4. Data Forwarding

A secure cloud storage system that supports the function of secure data forwarding by using a proxy re-encryption scheme. In the data forwarding phase, a user runs Key Recover() and Re Key Gen() send to storage server, and each storage server performs Re Enc().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 $RK^{ID}_{A\rightarrow B}$ and then sends $RK^{ID}_{A\rightarrow B}$ 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 is the combination of cipher texts under B's public key. In order to distinguish re-encrypted codeword symbols from intact ones, we call them original codeword symbols and re-encrypted codeword symbols, respectively.

3.5. Data Retrieval

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 SK_A; I. Finally, user A combines the partially decrypted codeword symbols to obtain the original message M.

4. Construction of Secure Cloud Storage System

Before presenting our storage system, 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 multiplicative groups with a prime order p and $g \varepsilon G_1$ be a generator. A map is a bilinear map $\tilde{e}:G_1xG_1\rightarrow G_2$ is efficiently computable and has the properties of bilinearity and Nondegeneracy: for any $x,y\varepsilon z^*_p$, $\tilde{e}(g^x,g^y) = \tilde{e}(g, g)^{xy}$ and $\tilde{e}(g,g)$ is not the identity element in G_2. Let Gen (1^λ) be an algorithm generating $(g,\tilde{e},G1,G2,p)$ where λ is the length of p. Let $x\varepsilon_R X$ denote that x is randomly chosen from the set X.

Decisional bilinear Diffie-Hellman assumption:

This assumption is that it is is computationally infeasible to distinguish the distributions $(g, g^x, g^y, g^z, \tilde{e}(g,g)^{xyz})$ and $(g, g^x, g^y, g^z\tilde{e}(g,g)^r)$ where x; y; z; $r\varepsilon_R Z^*_p$ probabilistic polynomial time algorithm A, the following is negligible (in λ).

$$|pr[A(g,g^x,g^y,g^z,Qb)=b:x,y,z,r\varepsilon_R Z^*_p,Q_0=\tilde{e}(g,g)^{xyz};Q_1=\tilde{e}(g,g)^r;b\varepsilon_R\{0,1\}]-½|.$$

Erasure coding over exponents:

Consider that the message domain is the cyclic multiplicative group G_2 described above. An encoder generates a generator matrix $G=[g_{i,j}]$ for $1\le i\le k, 1\le j\le n$ as follows: for each row, the encoder randomly selects an entry and randomly sets a value from Z^*_p to the entry. The encoder repeats this step v times with replacement for each row. An entry of a row can be selected multiple times but only set to one value. The values of the rest entries are set to 0. Let the message be $(m_1,m_2,...m_k)\varepsilon G^k_2$. The encoding process is to generate$(w_1,w_2...w_n)\varepsilon G^n_2$. Where $w_j=m_1^{g1j}m_2^{g2j}...m_k^{gkj}$ for $1\le j\le n$. The first step of the decoding process is to compute the inverse of a kxk submatrix k of G. Let k be $[g_{i,}j_i]$ for $1\le i,j_i\le k$. Let $k^{-1}=[d_{i,}j]_{i\le1,j\le k}$. The final step of the decoding process is to compute $m_i=w_{j1}^{d1,i}w_{j2}^{d2,i}$

for $1 \le i \le k$.

Our approach:

A threshold proxy re-encryption scheme with multiplicative homomorphic property. An encryption scheme is multiplicative homomorphic if it supports a group operation Θ on encrypted plaintexts without decryption

$$D(SK,E(PK,m_1)\ \Theta E(PK,m_1)) = m1 \cdot m2,$$

Where E is the encryption function, D is the decryption function, and(P_k,S_k) is a pair of public key and secret key. Given two coefficients g_1 and g_2, two message symbols m_1 and m_2 can be encoded to a codeword symbol $m_1^{g1}m_2^{g2}$ in the encrypted form

$$C=E(PK,m_1)^{g1}\Theta E(PK,m_2)^{g2}=E(PK,m_1^{g1}\cdot m_2^{g2})$$

Thus, a multiplicative homomorphic encryption scheme supports the encoding operation over encrypted messages. To 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 $t \ge k$. 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.

5. Secure Data Forwarding Algorithm

The algorithm SetUp(1^{λ}) generates the parameter μ. First, the user locally stores the secret key.

SetUp(1^{λ}): Run Gen(1^{λ}) to obtain $(g,h,\tilde{e},G_1,G_2,p)$,where $\tilde{e}$: $G_1 x\ G_1 \rightarrow G_2$ is a bilinear map, g and h are generators of G_1.Both G_1 and G_2 have the prime order P.Set $\mu=(g,h,\tilde{e},G_1,G_2,p,f)$, where f: $Z_p^* x \{0,1\}^* \rightarrow Z_p^*$ is a one way hash function.

KeyGen(μ): For a user A, the algorithm selects $a_1,a_2,a_3 \varepsilon_R Z_p^*$ and sets

$$PK_A = (g^{a1}, h^{a2}),\ SK_A = (a_1; a_2; a_3);$$

ShareKeyGen(SKA,t,m): This algorithm shares the secret key SK_A of a user A to a set m key servers by using two polynomials $f_{A,1}(z)$ and $f_{A,2}(z)$ of degree (t-1) over the finite field GF(p)

$$f_{A,1}(z)=a_1+v_1z+v_2z^2+...+v_{t-1}z^{t-1}(modp),$$

$$f_{A,2}(z)=a_2^{-1}+v_1z+v_2z^2+...+v_{t-1}z^{t-1}(modp),$$

where $v_1, v_2 ... v_{t-1} \varepsilon_R Z_p^*$.The key share of the secret key SK_A to the key server KS_i is $SK_{A,i}=(f_{A,1}(i), f_{A,2}(i))$,where $1 \le i \le m$. Next to compute the identity token and performs the encryption algorithm Enc(·) and encoding operation.

Enc($PK_A,\tau,m_1,m_2 \ldots m_k$): For $1 \le i \le k$, this algorithm computes

$$C_i=(0,\alpha_i,\beta,\gamma_i)=0,g^{ri},\tau,m_i,\tilde{e}(g^{\alpha 1},\tau^{ri})),$$

Where $ri \varepsilon_R Z_p^*$, $1 \le i \le k$ and 0 is the leading bit indicating an original cipertext.

Encode($C_1, C_2 \ldots, C_k$): For each cipertext C_i, the algorithm randomly selects a coefficient g_i. If some cipertext C_i is $(0,1,\tau,1)$,the coefficient g_i is set to 0. Let $C_i=(0, \alpha_i,\beta,\gamma_i)$ the encoding process is to compute an original codeword symbol C'.

$$C'= (0,\prod^k_{i=1}(\alpha_i^{gi}),\beta,\prod^k_{i=1}(\gamma_i^{gi}))$$

$$= (0,g^{r1},\tau,W\tilde{e}(g,\tau)^{a1r'})$$

Where $W=\prod^k_{i=1} m_i^{gi}$ and $r'=\Sigma^k_{i=1} g_i r_i$. The encoded result is $(C',g_1,g_2,\ldots g_k)$. If user A to forward a message to another user to perform KeyRecover(),ReKeyGen(), and ReEnc().

KeyRecover($SK_{A,i1},SK_{A,i2},\ldots,SK_{A,it}$): Let $T=\{i_1,i_2,i_t\}$.This algorithm recovers a_1 is lagrange interpolation as follows.

$$a_1 =\Sigma_{s\varepsilon T}(f_{A,1}(s)\prod_{s'\varepsilon T/\{s\}} -s'/s-s')mod\ p.$$

ReKeyGen(PK_A, SK_A,ID, PK_B): This algorithm selects $e\varepsilon_R Z_p^*$ and computes

$$RK^{ID}_{A\rightarrow B}=((h^{b2})^{a1(f(a3,ID)+e)},h^{a1,e}).$$

ReEnc($RK^{ID}_{A\rightarrow B}$,C'): In this algorithm to shares the codeword symbol let $C'=(0,\alpha,\beta,\gamma)=(0,g^{r'},\tau,W\tilde{e}(g^{a1},\tau^{r'}))$ for some r' and some W, and $RK^{ID}_{A\rightarrow B}=((h^{b2})^{a1(f(a3,ID)+e)},h^{a1,e})$ for some e. The re-encrypted codeword symbol is computed as follows:

$$C'=(1,\alpha, h^{b2,a1(f(a3,ID)+e)},\gamma.\tilde{e}(\alpha, h^{a1,e}))$$

$$=(1,g^{r'},h^{b2,a1(f(a3,ID)+e)},W\tilde{e}(g,h)^{a1r'(f(a3,ID)+e)}).$$

The leading bit indicates C' is a re-encrypted cipertext. If user A retrieves his own Message to perform ShareDec() and combine() operation.

ShareDec(SK_j,X_i): X_i is a codeword symbol, where $X_i= (b,\alpha,\beta,\gamma)$ and b is the indicator for original and re-encrypted codeword symbols. SK_j is a key share, where $SK_j=(sk_0,sk_1)$. By using the key share SK_j, the

partially decrypted codeword symbol $\zeta_{i,j}$ of X_i is generated as follows:

$\zeta_{i,j}=(b,\alpha,\beta,\beta^{skb},\gamma)$.

Combine($\zeta_{i_1,j_1},\zeta_{i_2,j_2}$, . . . ζ_{i_t,j_t}): Let a partially decrypted codeword symbol $\zeta_{i,j}$ be $(b, \alpha_{i,j},\beta_{i,j},\beta'_{i,j},\gamma_{i,j})$. This algorithm combines t partially decrypted codeword symbols, where $\beta_{i1,j1}=\beta_{i2,2}=\cdots=\beta_{it,jt}=\tau$, $j_1\neq j_2\neq\cdots\neq jt$ and there are at least k distinct values in $\{i_1,i_2,\ldots i_t\}$. Let $S_j=\{ j_1, j_2, \ldots j_t\}$ and $S=\{(i_1,j_1),(i_2,j_2)\ldots\ldots(i_t,j_t)\}$ Without loss of generality, let $S_1=\{i_1,i_2,\ldots i_k\}$ be k distinct values in $\{i_1,i_2,\ldots i_t\}$.

6. Analysis of Results

In the data storage phase, a user runs the Enc(·) algorithm and each storage server performs the Encode(·) algorithm. In the Enc(·) algorithm, generating each α_i requires a Exp_1, and generating each γ_i requires 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 ciphertexts at most. The cost is k Exp_1+(k-1) $Mult_1$ for computing α and k Exp_2+ (k-1) $Mult_2$ for computing γ.

In the data forwarding phase, a user runs KeyRecover(·) and ReKeyGen(·) and each storage server performs ReEnc(·). In the KeyRecover(·) algorithm, the computation cost is $O(t^2)$ F_p. In the ReKeyGen(·) algorithm, the computation cost is a Exp_1. In the ReEnc(·) algorithm, the computation cost is a Pairing and a $Mult_1$.

In the data retrieval phase, each key server runs the ShareDec(·) algorithm and the user performs the Combine(·) algorithm. In the ShareDec(·) algorithm, each key server performs a Exp_1 to get β^{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 w_j' from the partially decrypted codeword symbols $\zeta_{i,j}$',s, and the decoding computation which needs to perform the matrix inversion and recovery of blocks m_i' from the encoded blocks w_j',s. 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 w_j 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) $Mult_2$.

7. Conclusion

In this paper, consider a cloud storage system consists of storage servers and key servers. To 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 codeword symbols in our system. By using the threshold proxy re-encryption scheme, to present a secure cloud storage system that provides secure data storage and secure data forwarding functionality in a decentralized structure. Our storage system and some newly proposed 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. Our key servers act as access nodes for providing a front-end layer such as a traditional file system interface.

8. References

[1] 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 anIncompletely Trusted Environment," Proc. Fifth Symp. Operating System Design and Implementation (OSDI), pp. 1-14, 2002.

[2] G. Ateniese, K. Fu, M. Green, and S. Hohenberger, "Improved Proxy Re-Encryption Schemes with Applications to Secure Distributed Storage," ACM Trans. Information and System Security, vol. 9, no. 1, pp. 1-30, 2006.

[3] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L.Kissner, Z. Peterson, and D. Song, "Provable Data Possession at Untrusted Stores," Proc. 14th ACM Conf. Computer and Comm. Security (CCS), pp. 598-609, 2007.

[4] J. Shao and Z. Cao, "CCA-Secure Proxy Re-Encryption without Pairings," Proc. 12th Int'l Conf. Practice and Theory in Public Key Cryptography (PKC), pp. 357-376, 2009.

[5] C. Wang, Q. Wang, K. Ren, and W. Lou, "Privacy-Preserving Public Auditing for Data Storage Security in Cloud Computing,"Proc. IEEE 29th Int'l Conf. Computer Comm. (INFOCOM), pp. 525-533, 2010.