Journal of Information Hiding and Multimedia Signal Processing ©2013 ISSN 2073-4212
Ubiquitous International Volume **4**, Number **2**, April **2013**

Robust and Simple N-Party Entangled Authentication Cloud Storage Protocol Based on Secret Sharing Scheme

Hongfeng Zhu

Software College, Shenyang Normal University
No.253, HuangHe Bei Street, HuangGu District, Shenyang, P.C 110034 - China
zhuhongfeng1978@163.com

Tianhua Liu

Software College, Shenyang Normal University
No.253, HuangHe Bei Street, HuangGu District, Shenyang, P.C 110034 - China
liutianhua@sina.com

Dan Zhu

School of Foreign Languages, Shenyang Jianzhu University
zhudan413@gmail.com

Haiyang Li

Software College, Shenyang Normal University
No.253, HuangHe Bei Street, HuangGu District, Shenyang, P.C 110034 - China
oceanlee1982@163.com

Received November, 2012; revised January, 2013

Abstract. *Entangled cloud storage schemes think in different ways with traditional cloud storage secure schemes which provide high level security to force cloud storage servers for treating equally between normal users and influential users without discrimination. The current work on cloud secure storage mainly considers some low level rules, i.e. access control, fuzzy keyword search, data integrity checking, and identity-based cryptography. However, either these security mechanisms do not consider a subjectivity of cloud storage administrator (or called a potentially malicious cloud storage provider) to delete your data and even normal users' accounts just because you are a "normal user". In this paper, techniques are presented which aim at the cloud storage provider must provide the same quality service for each client. We develop a novel scheme, called EACS (entangled authenticated cloud storage), to settle the aforementioned typical problem. Its main idea is to use entangled method for designing a scheme which has four key policies: (i) N-clients can easily "entangle" their files into a single secret c to be store by a cloud storage provider S; (ii) Using secret c, each client may easily recovery their own original file respectively; (iii) If the server alters c in any way, no clients will be able to retrieve its original file (this policy is called all-or-noting-integrity). (iv) All the parties in the entangled scheme should be authenticated. Finally, we give a full specification of this scheme, including how to realize specific policies, how to design the scheme, how to prove the scheme's security.*

Keywords: Authentication, Cloud Storage, Secret Sharing, Entangled

1. **Introduction.** The terminology "cloud storage" refers to a paradigm shift in which mass data from a server are stored and managed through a client's web browser, with no preserved in local client of that data. Nowadays, outsourcing data storage provides several benefits, including improved scalability and accessibility, data replication and backup, and considerable cost saving. But people care more about cloud storage security problem, and they are not satisfied with the current cloud storage service because it only provides traditional secure services including confidentiality, integrity and availability. Furthermore, people are more interested in high level secure user experience about cloud storage: for each client, cloud storage server whether provides the same service? Or just because I'm a common user, the server gives me a high latency service and low secure level, even deletes my cloud stored files?

Three methods can solve above-mentioned problem:

(1) Provable Data Possession (PDP) [1] scheme: PDP scheme is eager for minimizing the file block accesses, the computation on the server, and the client-server communication. File blocks are signed by the clients through authenticated tags. During auditing, the remote server is challenged and proves possession of randomly picked file blocks by returning a short proof of possession. At last the clients need to be able to verify that a server has retained file metadata without retrieving the data from the server and without having the server access the entire file. Any data alteration or deletion will be detected with high probability. PDP is a hard security belonging to "after-the-fact-sense"(or we rather say that the server has the initiative).

(2) Proof of Retrievability (POR)[2] scheme: Error correction codes are included along with remote file blocks. POR wants to accomplish these checks without users having to download the files themselves. But it also belongs to "after-the-fact-sense".

(3) Entangled storage scheme (ECS)[3][4] makes all clients equal and with the same rights: ECS makes it financially inconvenient for a cloud provider to alter specific files and exclude certain "normal" customers, since doing so would destroy all entangled customers in the system, even those considered "important" and "profitable". Therefore, entangled storage schemes offer security "before-the-fact-sense" (or rather the clients have the initiative).

In addition, about cloud storage using secret sharing literatures, the first and the only one entangled storage scheme was proposed by Ateniese et al[3]. Their scheme is using Linear secret sharing to construct the entangled storage but its too complex and lacks of some secure properties. Recently some literatures using secret sharing in cloud storage to access other secure attributes, Wang et al [5] presents a data assured deletion approach in cloud storage: The scheme utilizes a key derivation tree to organize and manage keys which are pushed to DHT (Distributed Hash Table) network after partitioned by secret sharing scheme. Then dynamic property of DHT network makes keys be deleted periodically causing ciphertext can not be decrypted or accessed when authorized time expires and data assured deletion is implemented. Hong et al [6] puts forward a method called HCRE (hybrid cloud re-encryption) which is an efficient dynamic cryptographic access control in cloud storage: the key idea of HCRE is designed a secret sharing scheme to delegate the task of ABE (attribute-based encryption) re-encryption to the cloud service provider (CSP), which alleviates the administering burdens on the data owner.

In this paper, we put forward a new simple and efficient entangled authenticated cloud storage protocol (called $EACS$). We present our contributions below:

(1) $EACS$ is an Entangled-and-Recovery scheme that means each client who had a hand in the entanglement generation process should be able to recover its original file from secret c.

(2) $EACS$ has All-or-Nothing-Integrity Property: If the server or any other nodes alters secret c in any way, no clients will be able to retrieve its original file.

(3) $EACS$ achieves the high level security called Before-the-Fact-Sense: Someone/nodes can't do something because he already knows the consequences before the negative events (such as secret c could be altered) happened.

(4) $EACS$ realizes mutual authentication: All the parties in the $EACS$ protocol will authenticate each other in each step.

(5) $EACS$ protocol possesses strong security because it is based on Shamir's secret sharing scheme which is information theoretically secure without making any computational assumption.

(6) $EACS$ protocol owns expansibility property: As the size of the N-party including servers and clients increases linearly, the computation and communication to make it increases linearly too.

(7) $EACS$ is a freshness scheme: For each executing the $EACS$, each party (including servers, clients and Entangled Key Center–EKC) will select a random nonce to guarantee the freshness.

The remainder of this paper is organized as follows. In Section 2, we present some preliminaries. In Section 3, we give the process of the Entangled Authenticated Cloud Storage Protocol ($EACS$). In

Section 4, we give the security and efficiency analysis for EACS. Finally, we conclude the paper in Section 5.

2. **Preliminaries.** In this section, we present some fundamental backgrounds.

Definition 1 (Factoring Problem). Choose two large and different safe primes p and q (i.e., primes such that $p' = \frac{p-1}{2}$ and $q' = \frac{q-1}{2}$ are also primes) and compute $n = pq$. n is made publicly known. Factoring problem is defined to compute factors p and q such that $n = pq$.

Definition 2 (Factoring Assumption). It is computationally intractable to solve the Factoring Problem. Secret sharing schemes were introduced by both Blakley [7] and Shamir [8] independently in 1979 as a solution to safeguard cryptographic keys and have been studied extensively in many literatures[10 12]. In a secret sharing scheme, a secret s is divided into n shares and shared among n shareholders in such a way that, with any t or more than t shares, it is able to reconstruct this secret; but with fewer than t shares, it can't reconstruct the secret. Such a scheme is called a (t, n) secret sharing, expressed as (t, n)-SS.

Proposition 1. For any field F and any set of pairs $(x_1, y_1), ..., (x_t, y_t) \in F \times F$ where the X_i's are distinct, there exists exactly one polynomial g(x) over F of degree at most $t - 1$, such that $g(x_i) = y_i$ for $i = 1...t$. All coefficients of this polynomial can be efficiently computed from $(x_1, y_1), ..., (x_t, y_t)$.

Shamir's $(t, n) - SS$. In Shamir's [8] based on Lagrange interpolating polynomial, there are n shareholders $u = \{U_1, ..., U_n\}$ and a mutually trusted dealer D. The scheme consists of two algorithms:

(1) Share generation algorithm: dealer D does the following:

- dealer D first picks a polynomial $f(x)$ of degree $(t-1)$ randomly: $f(x) = a_0 + a_1 x + ... + a_{t-1} x^{t-1}$, in which the secret $s = a_0 = f(0)$ and all coefficients $a_0, a_1, ..., a_{t-1}$ are in a finite field $\mathbb{F}_p = GF(p)$ with p elements.
- D computes all shares: $s_i = f(i) \pmod{p}$ for $i = 1, ..., n$.
- Then, D outputs a list of n shares $(s_1, s_2, ..., s_n)$ and distributes each share s_i to corresponding shareholder P_i privately.

(2) Secret reconstruction algorithm: this algorithm takes any t shares $(s_{i_1}, ..., s_{i_t})$ as input, it can reconstruct the secret s as

$$s = f(0) = \sum_{i \in A} s_i \beta_i = \sum_{i \in A} s_i \left(\prod_{j \in A - \{i\}} \frac{x_j}{x_j - x_i} \right) \pmod{p},$$

where , $A = \{i_1, ... i_t\} \subseteq \{1, 2, ..., n\}$, β_i for $i \in A$ are Lagrange coefficients.

The above scheme satisfies the basic security requirements of secret sharing scheme as follows:

(1) With knowledge of any t or more than t shares, it can reconstruct the secret s easily;

(2) With knowledge of fewer than $(t-1)$ shares, it cannot reconstruct the secret s.

More precisely, secret sharing security model satisfies four properties:

Privacy: Take any subset I of the indices $\{1, 2, ..., n\}$ of size at most $t - 1$, and run S on input some $s \in S$. Then the probability distribution of $\{s_i | i \in I\}$ is independent of s. And unauthorized subsets of participants should be prevented from learning the secret.

Correctness (Recoverability): Take any subset J of the indices $\{1, 2, ..., n\}$ of size at least t, and run S on input some $s \in S$. Then s is uniquely determined by $\{s_i | i \in J\}$, and in fact there is an efficient algorithm that computes s from $\{s_i | i \in J\}$. In other words, authorized subsets of participants should be able to recover the secret by pooling their shares.

Adversary ability: The adversary in secret sharing scheme is a passive attacker who does not manipulate shares, and participants who either co-operate or do not co-operate in a reconstruction attempt. An adversary would have to get hold of at least t shares to steal the key, and on the other hand, as long as we loose no more than $n - t$ shares, there will still be enough information to reconstruct the key. So this is a solution that is at the same time robust against loss of information and information leakage.

Information-theoretic [9] security: secret sharing scheme has been studied in an information-theoretic security model, where the security is independent of the computing capabilities of an adversary. This can be relaxed and some schemes have been defined for computationally secure models where the scheme relies on the difficulty of a mathematical problem.

In Shamir's $(t, n) - SS$, the secret of each shareholder is just the $y - coordinate$ of $f(x)$ and the $x - coordinate$ is made publicly known. However, for security reason, we need to keep both $x - coordinate$ and $y - coordinate$ as each user's secret [13]. Furthermore, in Shamir's $(t, n) - SS$, the modulus p used for all computations is a prime number. In EACS protocol, to prevent insider attack, the modulus n

used for computations is a composite integer. Euclid's extended algorithm [14] can be used to compute modular inverse without factoring the composite modulus n.

3. The Entangled Authenticated Cloud Storage Protocol.

3.1. Setting parameters. The notation used hereafter is shown in Table 1.

TABLE 1. Notations

Symbol	Definition
EKC	Entangled Key Center
S	A cloud storage provider
U_i	The identity information of Clients
p , q	Two large and different safe primes
r_i , r_s, r_0	Random number
E_{U_i}	A secure symmetric encryption and a key chosen by U_i
E_k	A secure symmetric encryption with the key k
$File_i$	The entangled file chosen by U_i
(x_i, y_i)	A secret for U_i
(x_s, y_s)	A secret for S
$\oplus$	A bitwise Xor operator
$\|$	Means that two adjacent messages are concatenated
h	A secure one-way hash

3.2. Architecture of the $EACS$ protocol. The protocol $EACS$ architecture consists of three parties: the servers set, without loss of generality we just choose one cloud storage server; clients set; Trusted Third Party: EKC, entangled key center. The architecture of the scheme is illustrated in Figure1.

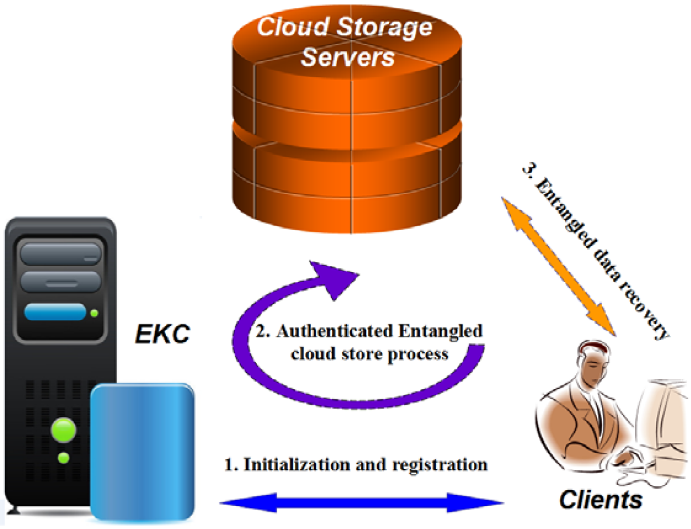
Cloud Storage Servers
EKC
2. Authenticated Entangled cloud store process
3. Entangled data recovery
1. Initialization and registration
Clients

FIGURE 1. Architecture of the $EACS$

3.3. Scheme Description. The $EACS$ consists of four phases: system initialization, user registration, entangled storage and recovery file.

System initialization. EKC randomly chooses two large and different safe primes p and q (i.e., p and q are primes such that $p' = (p-1)/2$ and $q' = (q-1)/2$ are also primes) and computes $n = pq$. n is made publicly known.

User registration. Each user is required to register at EKC to subscribe the key distribution service. During registration, EKC selects secret values secret (x_i, y_i) sent to each user U_i where $x_i, y_i \in Z_n^*$, and chooses a secret (x_s, y_s) sent to S where $x_s, y_s \in Z_n^*$.

Entangled Storage. The entangled storage process of $EACS$ just as Figure2 shows. This phase constitutes the core of the protocol and is performed whenever a group of users $U_1, ..., U_t$ decide to establish a entangled cloud storage for any cloud storage provider.

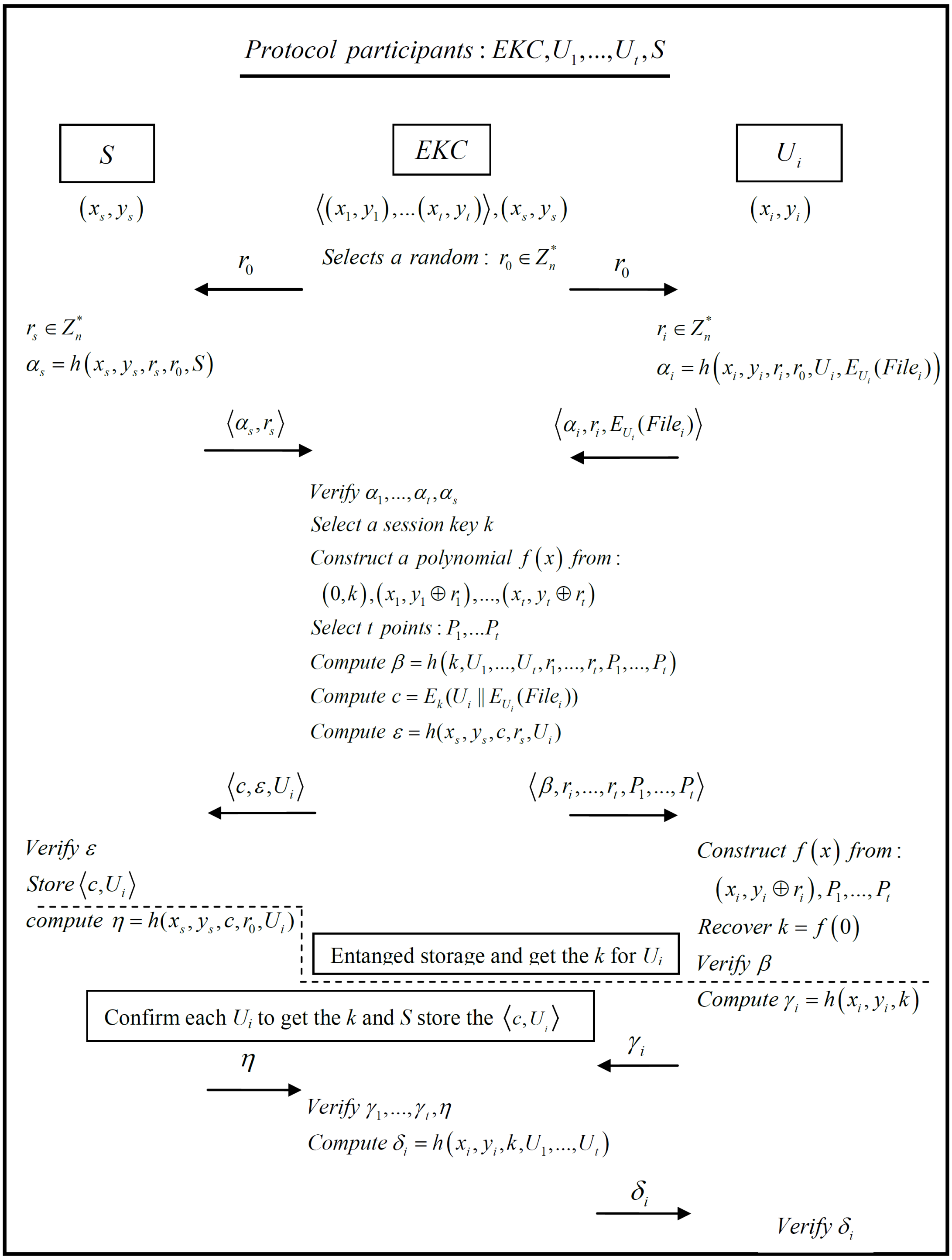
$Protocol\ participants: EKC, U_1, ..., U_t, S$
S
EKC
U_i
(x_s, y_s)
$\langle (x_1, y_1), ...(x_t, y_t) \rangle, (x_s, y_s)$
(x_i, y_i)
r_0
$Selects\ a\ random: r_0 \in Z_n^*$
r_0
$r_s \in Z_n^*$
$\alpha_s = h(x_s, y_s, r_s, r_0, S)$
$r_i \in Z_n^*$
$\alpha_i = h(x_i, y_i, r_i, r_0, U_i, E_{U_i}(File_i))$
$\langle \alpha_s, r_s \rangle$
$\langle \alpha_i, r_i, E_{U_i}(File_i) \rangle$
$Verify\ \alpha_1, ..., \alpha_t, \alpha_s$
$Select\ a\ session\ key\ k$
$Construct\ a\ polynomial\ f(x)\ from:$
$(0, k), (x_1, y_1 \oplus r_1), ..., (x_t, y_t \oplus r_t)$
$Select\ t\ points: P_1, ...P_t$
$Compute\ \beta = h(k, U_1, ..., U_t, r_1, ..., r_t, P_1, ..., P_t)$
$Compute\ c = E_k(U_i \| E_{U_i}(File_i))$
$Compute\ \varepsilon = h(x_s, y_s, c, r_s, U_i)$
$\langle c, \varepsilon, U_i \rangle$
$\langle \beta, r_i, ..., r_t, P_1, ..., P_t \rangle$
$Verify\ \varepsilon$
$Store \langle c, U_i \rangle$
$compute\ \ \eta = h(x_s, y_s, c, r_0, U_i)$
Entanged storage and get the k for U_i
$Construct\ f(x)\ from:$
$(x_i, y_i \oplus r_i), P_1, ..., P_t$
$Recover\ k = f(0)$
$Verify\ \beta$
$Compute\ \gamma_i = h(x_i, y_i, k)$
Confirm each U_i to get the k and S store the $\langle c, U_i \rangle$
γ_i
η
$Verify\ \gamma_1, ..., \gamma_t, \eta$
$Compute\ \delta_i = h(x_i, y_i, k, U_1, ..., U_t)$
δ_i
$Verify\ \delta_i$

FIGURE 2. The entangled storage process of $EACS$

Step1. A designated user of the group, called the initiator, sends a entangled-storage request to EKC. The request carries the list of participating users $\langle U_1, ..., U_t \rangle$ and the cloud storage provider S.

Step2. EKC selects a random $r_0 \in Z_n^*$ and broadcasts it along with the participant list $\langle U_1, ..., U_t, S \rangle$ in response to the request.

Step3. Each user U_i, for $i = 1, ..., t$, selects a random challenge $r_i \in Z_n^*$, computes $\alpha_i = h(x_i, y_i, r_i, r_0, U_i)$, and sends $\langle \alpha_i, r_i, E_{U_i}(File_i) \rangle$ to EKC where $E_{U_i}(File_i)$ is encrypted by U_i using a symmetric encryption and a secret key of his own choice. For example, we can use $h\left(U_i||r_i||x_i||y_i\right)$ to as a secret key. On the other side, S selects a random challenge $r_s \in Z_n^*$, computes $\alpha_s = h\left(x_s, y_s, r_s, r_0, S\right)$, and sends $\langle\alpha_s, r_s\rangle$ to EKC.

Step4. EKC checks the correctness of each α_i, α_s in the straightforward way. EKC aborts if any of the checks fails. EKC randomly selects a session key k and constructs by interpolation a $t-$th degree polynomial $f\left(x\right)$ passing through the $\left(t+1\right)$ points: $\left(x_1, y_1 \oplus r_1\right), ..., \left(x_t, y_t \oplus r_t\right)$ and $\left(0, k\right)$. After that, EKC selects t additional points $P_1, ..., P_t$ that lie on the polynomial $f\left(x\right)$. EKC then computes $\beta = h\left(k, U_1, ..., U_t, r_1, ..., r_t, P_1, ..., P_t\right)$, where h is a one-way hash function, and broadcasts $\langle\beta, r_1, ..., r_t, P_1, ..., P_t\rangle$ to the users. All computations with respect to $f\left(x\right)$ are performed modulo n.

The other side for S, after authenticated α_s, EKC will computes $c = E_k(U_i||E_{U_i}(File_i))$ and $\varepsilon = h(x_s, y_s, c, r_s, U_i)$, and sends $\langle c, \varepsilon, U_i\rangle$ to S.

Step5. Each U_i constructs the polynomial $f\left(x\right)$ from the $\left(t+1\right)$ points: $P_1, ..., P_t$ and $x_i, y_i \oplus r_i$. Then $x_i, y_i \oplus r_i$ recovers the session

key $k = f\left(0\right)$ and checks the correctness of β in the straightforward way. U_i aborts if the check fails. S verify ε and if the check corrects, and then S stores $\langle c, U_i\rangle$ in the right way. Next, S computes $\eta = h(x_s, y_s, c, r_0, U_i)$ and sends η to EKC for comfirmation.

Step6. Each U_i sends $\gamma_i = h\left(x_i, y_i, k\right)$ to EKC.

Step7. After receiving all γ_i and η, EKC verifies them. If all verify checks correctly, S sends $\delta_i = h\left(x_i, y_i, k, U_1, ..., U_t\right)$ to for $i = 1, ..., t$.

Step8. After receiving δ_i for each U_i, each U_i verifies his own δ_i to confirm for getting the right k. Finally, U_i deletes $File_i$ and stores $\langle k, E_{U_i}\rangle$ for extracting the *secretc* to get the $File_i$ later.

Recovery File. For any U_i, using an secure authenticated protocol such as [15-17] to exchange with S to get the *secretc*. Next, U_i get the $File_i$ by k and E_{U_i}.

4. Security and Efficiency Analysis for *EACS*.

4.1. Security analysis.

In this section, we first consider two types of security in our $EACS$ protocol, entangled security and hard security. Then, we prove that our proposed protocol achieves the security goals.

(1) Entangled Security (high level security)

Privacy of entanglement. That is, the client's files are private to both the other clients and the server. In our protocol, we adopt dual-encryption: one is symmetric encryption and a key chosen by U_i, the other is the whole encryption. Even the EKC, can't extract the each file of U_i.

Privacy of recovery. The inputs of clients $P_1, ..., P_t$ need to remain private also during the recovery process run by client P_i together with S. In the $EACS$ protocol just only executes a two-party secure authenticated protocol to get his own file, and not influences others (other clients and EKC).

All-or-Nothing-Integrity. The secure property means that if the server modifies the entanglement, nobody should be able to recover its original file anymore. From the Figure2 we can draw a conclusion that all the parties (servers, clients and EKC) in the protocol or outsiders can't get any $File_i$ from the *secretc* if anyone alters the *secretc*.

Before-the-Fact-Sense. The $EACS$ protocol supports the before-the-fact property because the server/administrator knows that there are any changes about the *secretc* will affects some big cheese inside the clients who can make own bear all the consequences.

(2) Hard Security (low level security)

Authentication. Mutual-authentication is provided through the tag of authentication in **step 4**. Authenticatied tag is a one-way hash output with the secret group key and all parties' random challenges as input. Because the group key is known only to authorized group parties (clients and servers) and EKC, unauthorized members cannot forge this tag. Any insider also cannot forge a group key without being detected since the group key is a function of the secret shared between each group member and EKC. In addition, any replay of P_i and authenticatied tag of EKC in **step 4** can be detected since the group key is a function of each group party's random challenge.

Confidentiality. Confidentiality is provided due to the security feature of a secret sharing scheme. However, for any unauthorized party (or outsider), there are only t points on $f(x)$ available. Thus, unauthorized party knows nothing about the group key. This property is information theoretically secure since there has no other computational assumption based on.

Freshness. Key freshness is ensured by EKC since a random group key is selected by EKC for each service request. In addition, the polynomial $f(x)$ used to recover the group key is a function of random challenge selected by each group member.

Replay attack. If the attacker attempts to reuse a compromised group key by replaying previously recorded key information from EKC, this attack cannot succeed in sharing this compromised group key with any group party since the group key is a function of each party's random challenge and the secret shared between group parties and EKC. A compromised group key cannot be reused if each party selects a random challenge for every agreement. Therefore, the replay attack is infeasible for the $EACS$ protocol.

Outsider attack. Although any attacker can impersonate a group member to issue a service request to EKC without being detected and EKC will respond by sending group key information accordingly; however, the group key can only be recovered by any group member who shares a secret with EKC. This security feature is information theoretically secure.

Insider attack. For a legal user or EKC involved in a process of $EACS$ protocol, he can just get the $U_i||E_{U_i}(File_i)$ and can't get any $File_i$ expect his own file. About S, he even can't deal with $c = E_k(U_i||E_{U_i}(File_i))$. So the $EACS$ protocol can resist insider attack.

4.2. **Efficiency analysis.** The $EACS$ protocol uses nested structure with two secret-key cryptography which adopts to encrypt long message and much more efficient than public-key cryptography. Moreover, the computation of Lagrange interpolating polynomial and Euclid's extended algorithm can be done instantly using modern person computer.

About the communication and the round of the $EACS$ protocol, we can say that as the size of the N-party including servers and clients increases linearly, the computation and communication to make it increases linearly too. So the $EACS$ protocol is a practical entangled cloud storage scheme.

5. **Conclusions.** The article puts forward a N-party entangled cloud storage $EACS$ protocol which adopts to secure one-way function to assure integrity and authentication, Shamir's $(t, n) - SS$ to assure confidentiality, and nested structure secret-key cryptography to assure entangled security. Security analysis for possible attacks and efficiency analysis are included. Next step we will study the entangle cloud signcryption storage in the future.

6. **Acknowledgement.** This research was supported by Liaoning Provincial Natural Science Foundation of China (Grant No. 20102202, 201102201) and Liaoning Baiqianwan Talents Program (Grant No.2011921046).

REFERENCES

[1] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and D. Song, Provable data possession at untrusted stores, *Proc. of the 14th ACM Conference on Computer and Communications Security*, pp. 598-609, 2007.

[2] A. Juels, and B. S. Kaliski Jr., PoRs: proofs of retrievability for large files, *Proc. of the 14th ACM Conference on Computer and Communications Security*, pp. 584-597, 2007.

[3] G. Ateniese, Ö. Dagdelen, I. Damgard, and D. Venturi, Entangled cloud storage, 2012, available at http://eprint.iacr.org/2012/511.pdf.

[4] J. Aspnes, J. Feigenbaum, *A. Yampolskiy, and S. Zhong, Towards a theory of data entanglement*, Spring, Berlin-Heidelberg, Germany, 2004.

[5] L. N. Wang, Z. W. Ren, R. w. Yu, F. Han, and Y. F. Dong, A data assured deletion approach adapted for cloud storage, *Journal of Acta Electronica Sinica*, vol. 40, no. 2, pp. 266-272, 2012.

[6] C. Hong, M. Zhang, and D. G. Feng, Achieving efficient dynamic cryptographic access control in cloud storage, *Journal of China Institute of Communications*, vol. 32, no. 7, pp. 125-132, 2011.

[7] G. R. Blakley, Safeguarding cryptographic keys, *Proc. of National Computer Conference*, pp. 313-317, 1979.

[8] A. Shamir, How to Share a Secret, *Communication of the ACM*, vol. 22, no. 11, pp. 612-613, 1979.

[9] E. Karnin, J. Greene, and M. Hellman, On secret sharing systems, *IEEE Trans. Information Theory*, vol. 29, no. 1, pp. 35-41, 1983.

[10] A. Beimel, and I. Orlov, Secret Sharing and Non-Shannon Information Inequalities, *IEEE Trans. Information Theory*, pp. 5634-5649, 2011.

[11] O. Farràs, and C. Padró, Ideal Hierarchical Secret Sharing Schemes, *IEEE Trans. Information Theory*, pp. 2284-2291, 2012.

[12] J. R. Metcalf-Burton, Improved upper bounds for the information rates of the secret sharing schemes induced by the Vámos matroid, *Journal of Discrete Mathematics*, vol. 311, no. 8-9, pp. 651-662, 2011.

[13] L. Harn, and C. Lin, Authenticated group key transfer protocol based on secret sharing, *IEEE Trans Computers*, vol. 59, no. 6, pp. 842-846, 2010.

[14] D. R. Stinson, *Theory and practice*, CRC Press, 2002.

[15] I. E. Liao, C. C. Lee, and M. S. Hwang, *Journal of Computer and System Sciences*, vol. 72, no. 4, pp. 727-740, 2006.

[16] C. C. Chang, and C. Y. Lee, *IEEE Trans. Industrial Electronics*, vol. 59, no. 1, pp. 629-637, 2012.

[17] M. K. Khan, S. K. Kim, and K. Alghathbar, Cryptanalysis and security enhancement of a more efficient & secure dynamic ID-based remote user authentication scheme. *Journal of Computer Communications*, vol. 34, no. 3, pp. 305-309, 2011.