IJDCST @October Issue- V-1, I-6, SW-30
ISSN-2320-7884 (Online)
ISSN-2321-0257 (Print)

A Secure Key Server based Data Storage and forwarding in Cloud

[1]A.Shalini, [2]SreeRama Srikanth

[1]Dept of CSE,Marri Laxman Reddy Institute of Technology ,Dundigal,Hyderabad-500043, A.P, INDIA

[2] Dept of CSE, Marri Laxman Reddy Institute of Technology ,Dundigal, Hyderabad-500043, A.P, INDIA

Abstract— Cloud Computing has been envisioned as the next generation architecture of IT Enterprise. In contrast to traditional solutions, where the IT services are under proper physical, logical and personnel controls, Cloud Computing moves the application software and databases to the large data centers, where the management of the data and services may not be fully trustworthy. General encryption schemes protect data confidentiality, but also limit the functionality of the storage system because a few operations are supported over encrypted data. This method supports encoding over encrypted messages using secure key servers and forwards the secure data. This method fully integrates encrypting, encoding, and forwarding. Suitable parameters are suggested for the number of copies of message sent to storage servers. The storage servers are queried by key servers. These parameters allow more flexible adjustment between the number of storage servers and robustness.

Keywords— (Encoding, Decentralized erasure code, proxy re-encryption, threshold cryptography, secure storage system.)

I. INTRODUCTION

In cloud computing, moving data into the cloud offers great convenience to the users since they do not have to worry about the complexities of direct hardware management. Users just use the services without being concerned about how computation is done and storage is managed. In this paper, we focus on designing a cloud storage system for robustness, confidentiality, and functionality. A cloud storage system is considered as a large-scale distributed storage system that consists of many independent storage servers. Security remains the critical issue that concerns potential clients. Data robustness is a major requirement for storage systems. There have been many proposals of storing data over storage servers.

In this paper we address the main aspects related to security of cloud storage. It presents an attempt to propose an effective and flexible security policy and procedures explicit to enhance the Data storage security and forwarding in the cloud. 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 to an erasure error of the codeword symbol. 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 results in risky data confidentiality. For good 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, we address the problem of forwarding data to another user by storage servers directly under the command of the data owner.

The system model has two servers distributed storage servers and key servers. It is risky to store cryptographic keys in a single device. The user distributes his cryptographic key to key servers that perform cryptographic functions on behalf of the user. The key servers are highly protected by security mechanisms. A new threshold proxy re-encryption scheme is proposed in this paper and integrated it with a secure decentralized code to form a secure distributed storage system. This encryption scheme supports encoding operations over encrypted messages and forwarding operations over encrypted and encoded messages. The integration of encoding, encryption, and forwarding makes the storage system to efficiently meet the requirements of data robustness, data confidentiality, and data forwarding. The proposed system meets the requirements that storage servers independently perform encoding and re - encryption and key servers independently perform partial decryption. This allows more flexible adjustment between the number of storage servers and robustness.

The contributions are as follows:
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:

- We construct a secure cloud storage system that supports the function of secure data forwarding by using a threshold proxy re-encryption scheme. The encryption scheme supports decentralized erasure codes over encrypted messages and forwarding operations over encrypted and encoded messages. Our system is highly distributed where storage servers independently encode and forward messages and key servers independently perform partial decryption.
- We present a general setting for the parameters of our
  secure cloud storage system. Our parameter setting of $n=ak^c$ supersedes the previous one of $n=ak\sqrt{k}$ where $c \geq 1.5$ and $a > \sqrt{2}$. 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 (a codeword symbol), which is a combination of encrypted message symbols.

II. RELATED WORK

We briefly review parallel and distributed storage systems, proxy re-encryption schemes, and integrity checking mechanisms.

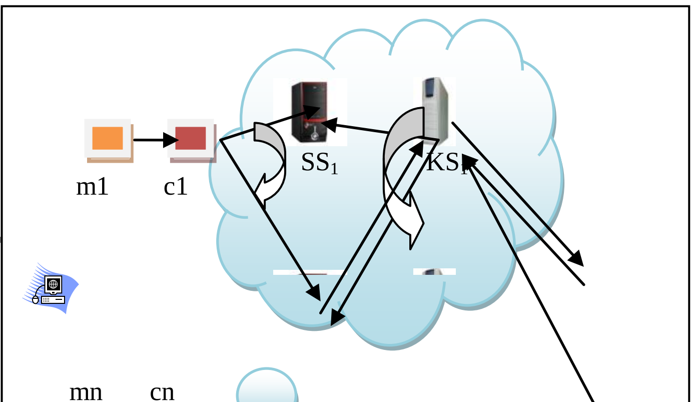
A) PARALLEL AND DISTRIBUTED STORAGE SYSTEMS

Parallel and Distributive storage systems has undergone impressive change over recent years. New architectures and applications have rapidly become the central focus of the discipline. These changes are often a result of cross fertilisation of parallel and distributed technologies with other rapidly evolving technologies. It is of paramount importance to review and assess these new developments in comparison with recent research achievements in the well-established areas of parallel and distributed computing, from industry and the scientific community. At the early years, the Network-Attached Storage (NAS) [2] and the Network File System (NFS) 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 [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. One way to reduce the expansion rate is to use erasure codes to encode messages [3]. A message is 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.

B) PROXY RE-ENCRYPTION SCHEMES

Proxy re-encryption (PRE) allows a semi-trusted proxy to convert a cipher- text intended for a user into a ciphertext for another user without learning anything about the underlying plaintext. Chunbo Ma et al. have proposed a group based proxy re-encryption scheme to convert a ciphertext from one group to another. Any group member can independently decrypt the ciphertexts encrypted to its group. Proxy re-encryption schemes are proposed by Mambo and Okamoto [14] and Blaze et al. [15]. A proxy server can transfer a cipher text under a public Key PKA to a new one under another public key PKB by using The re-encryption key RK A→B. The server does not know the Plaintext during transformation. In their work, message are first Encrypted by the owner and then stored in a storage server.

When a user wants to share his messages, he Confidentiality and supports the data forwarding function. Our work further integrates encryption, re-encryption, and encoding such that storage robustness is strengthened. Type based proxy re-encryption schemes [4] provide a better granularity on the granted right of a re-encryption key. By using this kind of proxy re-encryption schemes a user can decide which type of messages and with whom he wants to share. Key-private proxy re-encryption schemes are proposed by Ateniese et al. [18]. In a 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 against proxy servers. Although most proxy re-encryption schemes use pairing operations, there exist proxy re-encryption schemes without pairing [19].



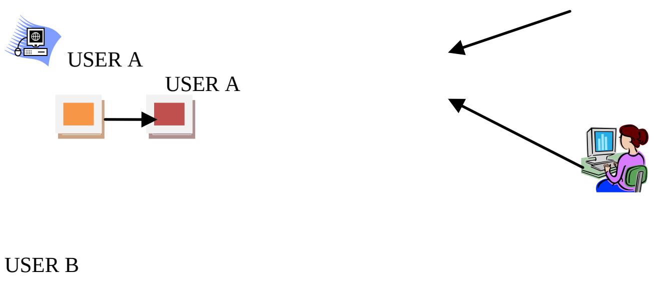


Fig. 1. A general system model of our work

C) *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 data are properly stored in storage servers. The concept of provable data possession [20], [21] and the notion of proof of storage [22], [23], [24] are proposed. Later, public auditability of stored data is addressed in [25]. Nevertheless all of them consider the messages in the clear text form.

III. SCENARIO

We present the scenario of the storage system, the threat model that we consider for the confidentiality issue, and a discussion for a straightforward solution.

A) *SYSTEM MODEL*

As shown in Fig. 1, our system model consists of users, n storage servers SS_1, SS_2, . . . , SSn, and m key servers KS_1, KS_2, . . . KSm. 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 parameters 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 $SK_{A,i}$, $1\leq i\leq m$. 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 cipher text Ci 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.

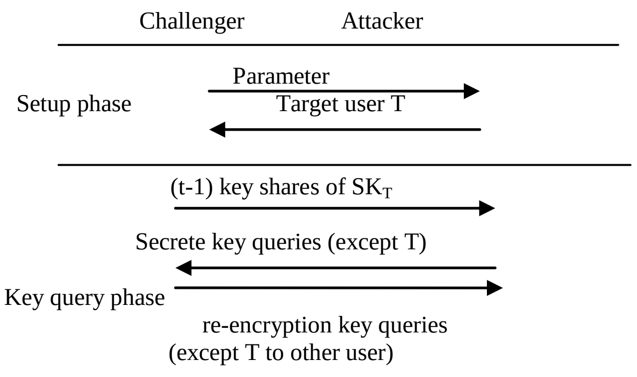
In the *data forwarding phase*, user A forwards his encrypted message with an identifier ID stored in storage servers to user B such that B can decrypt the forwarded message by his secret key. To do so, A uses his secret key SK_A and B's public key PK_B to compute a re-encryption key $RK^{ID}_{A\rightarrow B}$ and then sends $RK^{ID}_{A\rightarrow B}$ to all storage servers. Each storage server uses the 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 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 $SK_{A,i}$. Finally, user A combines the partially decrypted codewor 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 as a new one and stores it. The system is then recovered.

B) *THREAT MODEL*

We consider 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, non target users, and up to (t-1) key servers. The attacker analyzes stored messages in storage servers, the secret keys of non target users, and the shared keys stored in key servers. Note that the 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 of the proxy Re-encryption scheme in a threshold version, as shown in Fig 2. A cloud storage system modeled in the above is secure if no probabilistic polynomial time attacker wins the game with a non negligible advantage.



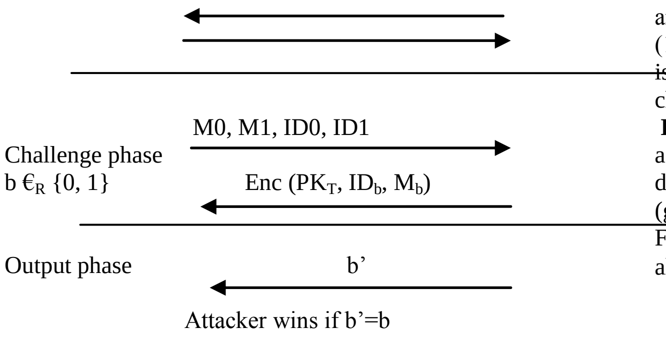


Fig. 2. The security game for the chosen plaintext attack.

A secure 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 reencrypted ciphertext using the secret key SK_B. Therefore, this model addresses the security of data storage and data forwarding.

C) A STRAIGHTFORWARD SOLUTION

A straightforward solution to supporting the data forwarding function in a distributed storage system is as follows: When the owner A wants to forwarding 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 reencryption scheme, the owner sends a re-encryption to storage servers such that storage server perform the re-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 secure storage system.

IV. CONSTRUCTION

Before presenting our storage system, we briefly introduce the algebraic setting, the hardness assumption, and 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 € G_1 be a generator. A map ẽ: $G_1 \times G_1 \rightarrow G_2$ is a bilinear map if it is efficiently computable and has the properties of bilinearity and non degeneracy: for any x, y € Zp*, ẽ (gx, gy) = ẽ (g, g) xy and ẽ (g, g) is not the identity element in G_2. Let Gen (1λ) be an algorithm generating (g, ẽ, G_1, G_2, p), where λ is the length of p. Let x €R 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 distribution (g, gx, gy, gz, ẽ (g, g) xyz) and (g, gx, gy, gz, ẽ (g, g) r) where x, y, z, r €R Zp*. Formally, for any probabilistic polynomial time algorithm Ȁ, the following is negligible (in λ):

| Pr [Ȁ (g, gx, gy, gz, Qb) = b: x, y, z, r €R Zp*,
Q0 = ẽ (g, g) xyz; Q1 = ẽ (g, g) r; b €R {0, 1}]- 1⁄2|.

Erasure coding over exponents: We consider that the message domain is the cyclic multiplicative group G2 described above. An encoder generates a generator matrix G = [$g_{i,\ j}$] for 1≤ i ≤ k, 1≤ j ≤ n as follows: for each row, the encoder randomly selects an entry and randomly sets a value form Zp* 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 value of the rest entries are set to 0. Let the message be (m_1, m_2….m_k) € G2k. The encoding process is to generate (w_1, w_2…..w_n) € G2k, where w_j = $m_1 g_i$, 1, $m_2 g_i$, 2….$m_k g_k$, j for 1≤ j≤ n. The first step of the decoding process is to compute the inverse of a k × k sub matrix K of G.

Our approach: We use a threshold proxy re-encryption scheme with multiplication homomorphism property. An encryption scheme is multiplicative homomorphism if it supports a group operation ⊙ on encrypted plaintexts without decryption

D (S_K, E (P_K, m_1) ⊙ E (P_K, m_2)) = m_1•m_2,

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 $m_1 g_1$, $m_2 g_2$ in the encrypted form

C=E (P_K, m_1) g_1 ⊙ E (P_K, m_2) g_2 = E (P_K, $m_1 g_1$•$m_2 g_2$).

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. 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 symbol.

A) ANALYSIS

We analyze storage and computation complexities, correctness, and security of our cloud storage system in this section. Let the bitlength of an element in the group G_1 be l_1 and G_2 be l_2. Let coefficient $g_{i,\ j}$ be randomly chosen from {0, 1} l3.

Storage Cost: To store a message of k blocks, a storage server SSj stores a codeword symbol (b, α_j, r, γ_j) and the coefficient vector (g_1, j, g_2, j,….,g_k, j). There are total of (1+2l_1 +l_2+kl_3) bits. The average cost for a message bit stored in a storage server is (1+2l_1 +l_2+kl_3)⁄kl_2 bits.

Computation Cost: We measure the computation cost by the number of pairing operation, modular exponentiations in G_1 and G_2, modular multiplications in G_1 and G_2, and arithmetic operation over GF (p). These operations are denoted as Pairing, Exp1, Exp2, Mult1, Mult2, and Fp, respectively. The cost is summarized in Table 1. Computing an Fp take much less time than computing a Mult1 or a Multi2. The time of computing an Exp1 is 1.5[log p] times as much as the time of computing a Multi1, on average, (by using the square and multiply algorithm). Similarly, the time of computing an Exp2 is 1.5[log p] times as much as the time of computing a Multi2, on average.

Table 1
The Computation Cost of Each Algorithm

Operation	Computation Cost
Enc	k Pairing + k Exp1 + k Mult2
Encode(for each storage server)	k Exp1 + Exp2 + (k-1) Mult1 + (k-1) Mult2
KeyRecover	O(t2) Fp
ReKeyGen	1 Exp1
ReEnc(for each storage server)	1 Pairing + 1 Mult2
ShareDec(for t key servers)	T Exp1
Combine	K Pairing + t Mult1 + (t-1) Exp1 + O(t2 + k3) Fp + k2 Exp2 + (k+1)k Mult2

- Pairing: a pairing computation of ẽ.
- Exp1 and Exp2: modular exponentiation computation in G1 and G2, respectively.
- Mult1 and Mult2: a modular multiplication computation in G1 and G2, respectively.
- Fp: an arithmetic operation in GF (p).

V. CONCLUSION

In this paper, we consider 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 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, we present a secure cloud storage system that provides secure data storage and secure data forwarding functionality in a decentralized structure. Moreover, each storage server independently performs encoding and re-encryption and each key server independently performs partial decryption. Our storage system and some newly proposed content addressable file systems and storage system [27], [28], [29] 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. Further study on detailed cooperation is required.

REFERENCES

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

[2] P. Druschel and A. Rowstron, "PAST: A Large-Scale, Persistent Peer-to-Peer Storage Utility," Proc. Eighth Workshop Hottopics inOperating System (HotOS VIII), pp. 75-80, 2001.

[3] 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

[4] Haeberlen, A. Mislove, and P. Druschel, "Glacier: HighlyDurable, Decentralized Storage Despite Massive Correlated Failures,"Proc. Second Symp. Networked Systems Design and implementation (NSDI), pp. 143-158, 2005.

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

[6] H.-Y. Lin and W.-G. Tzeng, "A Secure 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 and Experience, 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.1 985.

[9] M. Kallahalla, E. Riedel, R. Swaminathan, Q. Wang, and K. Fu, "Plutus: Scalable Secure File Sharing on Untrusted Storage," Proc. Second USENIX Conf. File and Storage Technologies (FAST), pp. 29-42, 2003.

[10] S.C. Rhea, P.R. Eaton, D. Geels, H. Weatherspoon, B.Y. Zhao, and J. Kubiatowicz, "Pond: The Oceanstore Prototype," Proc.Second USENIX Conf. File and storage Technologies (FAST), pp. 1-14, 2003.

[11] R. Bhagwan, K. Tati, Y.-C. Cheng, S. Savage, and G.M. Voelker,"Total Recall: System Support for Automated Availability Management," Proc. First Symp. Networked Systems Design and Implementation (NSDI), pp. 337-350, 2004.

[12] A.G. Dimakis, V. Prabhakaran, and K. Ramchandran, "Ubiquitous Access to Distributed Data in Large-Scale Sensor Networks through Decentralized Erasure Codes," Proc. Fourth Int'l Symp. Information Processing in Sensor Networks (IPSN), pp. 111-117, 2005.

[13] A.G. Dimakis, V. Prabhakaran, and K. Ramchandran, "DecentralizedErasureCodesforDistributed Networked Storage," IEEE Trans. Information Theory, vol. 52, no. 6 pp. 2809-2816, June 2006.

[14] M. Mambo and E. Okamoto, "Proxy Cryptosystems: Delegation of the Power to Decrypt Ciphertexts," IEICE Trans Fundamentals ofElectronics, Comm.and Computer Sciences, vol. E80-A, no. 1, pp. 54-63, 1997.

[15] M. Blaze, G. Bleumer, and M. Strauss, "Divertible Protocols and Atomic Proxy Cryptography," Proc. Int'l Conf. Theory and Application of Cryptographic Techniques (EUROCRYPT), pp. 127-144, 1998.

[16] 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.

[17] Q. Tang, "Type-Based Proxy Re-Encryption and Its Construction," Proc. Ninth Int'l Conf. Cryptology in India: Progress in Cryptology (INDOCRYPT), pp. 130-144, 2008.

[18] G. Ateniese, K. Benson, and S. Hohenberger, "Key-Private Proxy Re-Encryption," Proc. Topics in Cryptology (CT-RSA), pp. 279-294,2009.

[19] 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.

[20] 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.

[21] G. Ateniese, R.D. Pietro, L.V. Mancini, and G. Tsudik, "Scalable and Efficient Provable Data Possession," Proc. Fourth Int'l Conf. Security and Privacy in Comm. Netowrks (SecureComm), pp. 1-10, 2008.

[22] H. Shacham and B. Waters, "Compact Proofs of Retrievability," Proc. 14th Int'l Conf. Theory and Application of Cryptology and Information Security (ASIACRYPT), pp. 90-107, 2008.

[23] G. Ateniese, S. Kamara, and J. Katz, "Proofs of Storage from Homomorphic Identification Protocols," Proc. 15th Int'l Conf. Theory and Application of Cryptology and Information Security (ASIACRYPT), pp. 319-333, 2009.

[24] K.D. Bowers, A. Juels, and A. Oprea, "HAIL: A High-Availability and Integrity Layer for Cloud Storage," Proc. 16th ACM Conf. Computer and Comm. Security (CCS), pp. 187-198, 2009.

[25] 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.

[26] A. Shamir, "How to Share a Secret," ACM Comm., vol. 22, pp. 612-613, 1979.

[27] C. Dubnicki, L. Gryz, L. Heldt, M. Kaczmarczyk, W. Kilian, P.Strzelczak, J. Szczepkowski, C. Ungureanu, and M. Welnicki,"Hydrastor: A Scalable Secondary Storage," Proc. Seventh Conf. Fileand Storage Technologies (FAST), pp. 197-210, 2009.

[28] C. Ungureanu, B. Atkin, A. Aranya, S. Gokhale, S. Rago, G.Calkowski, C. Dubnicki, and A. Bohra, "Hydrafs: A High-Throughput File System for the Hydrastor Content-Addressable Storage System," Proc. Eighth USENIX Conf. File and StorageTechnologies (FAST), p. 17, 2010.

[29] W. Dong, F. Douglis, K. Li, H. Patterson, S. Reddy, and P. Shilane, "Tradeoffs in Scalable Data Routing for Deduplication Clusters," Proc. Ninth USENIX Conf. File and Storage Technologies (FAST), p. 2,2011.