

ONLINE SECURITY FOR PRIVATE CLOUD SECURITY

By

¹A.Haritha Deepthi, ²M.Mohanalatha, ³V.Sridevi

Department of Information Technology,
PSG Polytechnic College, Coimbatore, India

harithadeepthibtech@gmail.com, m.mohanalatha@gmail.com, sridevivavs@gmail.com

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 confidentiality, but also limit the functionality of the storage system because few operations are supported over encrypted data. Constructing a secure storage system that supports multiple functions are challenging when the storage system is distributed and has no central authority. Propose a threshold proxy re-encryption scheme and integrate it with a decentralized erasure code such that a secure distributed storage system is formulated. The distributed storage system not only supports secure and robust data storage and retrieval, but also let a user forward data in the storage servers to another user without retrieving the data back. The main technical contribution is that the proxy re-encryption scheme supports encoding operations over encrypted messages as well as forwarding operations over encoded and encrypted messages. Our method fully integrates encrypting, encoding, and forwarding. Analyze and suggest 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.

General Terms

Decentralized erasure code, proxy re-encryption, secure 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. 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. 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 failure corresponds 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 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. We consider 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 mechanisms.

2. ARCHITECTURE

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

2.1 Distributed Storage Systems

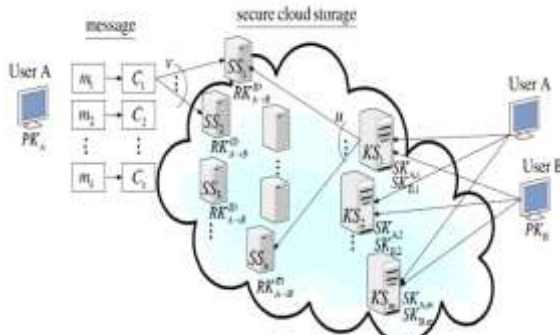


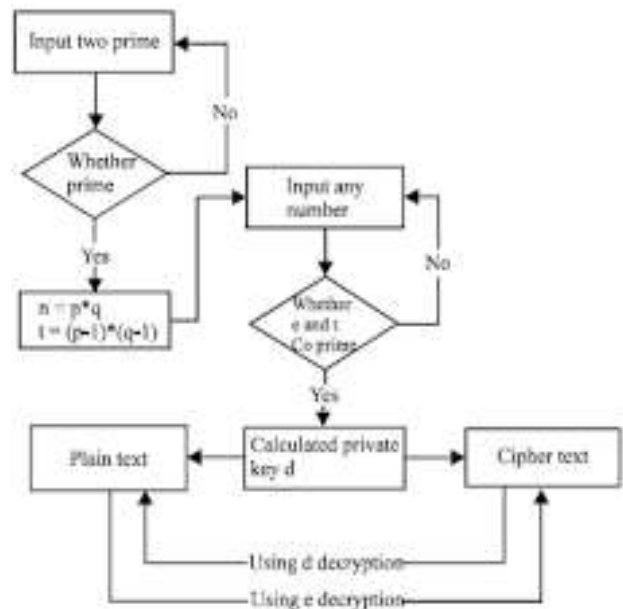
Fig 2.1:Architecture Diagram

At the early years, the Network-Attached Storage (NAS) 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. 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 message. 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 considered the case that $n \geq ak$ 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 $1 - \frac{1}{2^{b \ln k}}$ of a successful data retrieval, where $v \geq b \ln k$, $b > 5a$, and p is the order of the used group. The sparsity parameter $v \geq b \ln k$ 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 to get the message. Lin and Tzeng [6] addressed robustness and confidentiality issues by presenting a secure 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.

3. PROXY RE-ENCRYPTION

3.1 Proxy Re-Encryption Schemes

Proxy re-encryption schemes are proposed by Mambo and Okamoto and Blaze et al. In a proxy re-encryption 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 $RK_{A \rightarrow B}$. The server does not know the plaintext during transformation. Ateniese et al. 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. Type-based proxy re-encryption



3.1: Algorithm

schemes proposed by Tang provide a better granularity on the granted right of a re-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. 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.

3.2 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 and the notion of proof of storage are proposed. Later, public audit ability of stored data is addressed in . Nevertheless all of them

consider the messages in the clear text form.

4. 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.

4.1 System Model

As , our system model consists of users, n storage servers $SS_1; SS_2; \dots; SS_n$, and m key servers $KS_1; KS_2; \dots; KS_m$. 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 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 $m_1; m_2; \dots; m_k$ and has an identifier ID . User A encrypts each block m_i into a cipher text 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 $RK_{A \rightarrow B}$ and then sends $RK_{A \rightarrow B}$ 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. 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 . 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.

4.2 Analysis

We analyze storage and computation complexities, correctness, and security of our cloud storage system in this section. Let the bit-length of an element in the group GG_1 be l_1 and GG_2 be l_2 . Let coefficients $g_{i,j}$ be randomly chosen

from $f_0; 1; g_1; \dots; g_{l_1-1}$. Storage cost. To store a message of k blocks, a storage server SS_j stores a codeword symbol $\delta b; \underline{g}; \underline{g}$ and the coefficient vector $\delta g_1; g_2; \dots; g_k; j$. They are total of $\delta 1 + 2l_1 + 2l_2 + kl_3$ bits, where $\underline{g}; \underline{g} \in GG_1$ and $\underline{g} \in GG_2$. The average cost for a message bit stored in a storage server is $\delta 1 + 2l_1 + 2l_2 + kl_3 = kl_2$ 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 $GF(\delta p)$. These operations are denoted as Pairing, Exp1, Exp2, Mult1, Mult2, and Fp, respectively. Let coefficients $g_{i,j}$ be randomly chosen from $f_0; 1; g_1; \dots; g_{l_1-1}$. Storage cost. To store a message of k blocks, a storage server SS_j stores a codeword symbol $\delta b; \underline{g}; \underline{g}$ and the coefficient vector $\delta g_1; g_2; \dots; g_k; j$. They are total of $\delta 1 + 2l_1 + 2l_2 + kl_3$ bits, where $\underline{g}; \underline{g} \in GG_1$ and $\underline{g} \in GG_2$. The average cost for a message bit stored in a storage server is $\delta 1 + 2l_1 + 2l_2 + kl_3 = kl_2$ 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 $GF(\delta p)$. These operations are denoted as Pairing, Exp1, Exp2, Mult1, Mult2, and Fp, respectively. The cost is summarized in Table 1. Computing an Fp takes much less time than computing a Mult1 or a 1000 IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, VOL. 23, NO. 6, JUNE 2012 Mult2.

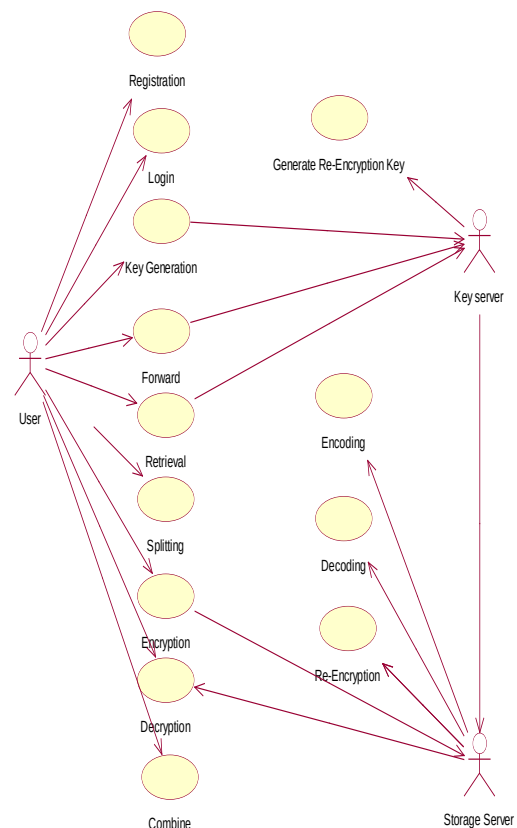


Fig 4.1: UsecaseDiagram

The time of computing an Exp1 is $1.5d \log p$ times as much as the time of computing a Mult1, on average, (by using the square-and-multiply algorithm). Similarly, the time of computing a Exp2 is $1.5d \log p$ times as much as the time of computing a Mult2, on average. In the data storage phase, a user runs the Enc δ _P algorithm and each storage server performs the Enc δ _P algorithm. In the Enc δ _P algorithm, generating each i requires a Exp1, and generating each j requires a Exp1, a Pairing, and a Mult2. Hence, for k blocks of a message, the cost is $(k \text{ Pairing} + 2k \text{ Exp1} + k \text{ Mult2})$. For the Enc δ _P algorithm, each storage server encodes k ciphertexts at most. The cost is $k \text{ Exp1} + \delta k - 1 \text{ Mult1}$ for computing $_$ and $k \text{ Exp2} + \delta k - 1 \text{ Mult2}$ for computing $_$. In the data forwarding phase, a user runs KeyRecover δ _P and ReKeyGen δ _P and each storage server performs ReEnc δ _P. In the ReEnc δ _P algorithm, the computation cost is a Pairing and a Mult1. In the data retrieval phase, each key server runs the ShareDec δ _P algorithm and the user performs the Combined δ _P algorithm. In the ShareDec δ _P algorithm, each key server performs a Exp1 to get $_skb$ 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 $t \text{ Exp1}$. In the Combined δ _P algorithm, it needs the computation of the Lagrange interpolation over exponents in GG_1 , the computation of the encoded blocks w_j 's from the partially decrypted codeword symbols $\sim_{i,j}$'s, and the decoding computation which needs to perform the matrix inversion and recovery of blocks m_i 's from the encoded blocks w_j 's. The Lagrange interpolation over exponents in GG_1 needs $O(\delta^2 \log p)$, $t \text{ Exp1}$, and $\delta t - 1 \text{ Mult1}$. Computing an encoded block w_j needs one Pairing and one modular division, which takes 2 Mult2 . As for the decoding computation, the matrix inversion takes $O(\delta^3 \log^3 p)$ arithmetic operations over $GF(\log p)$, and the decoding for each block takes $k \text{ Exp2}$ and $\delta k - 1 \text{ Mult2}$.

5. CONCLUSION

The Computation Cost of Each Algorithm in Our Secure Cloud Storage System 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, 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 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.

6. ACKNOWLEDGMENTS

The authors thank anonymous reviewers for their valuable comments. The research was supported in part by projects ICTL-100-Q707, ATU-100-W958, NSC 98-2221-E-009-068-MY3, NSC 99-2218-E-009-017-, and NSC 99-2218-E-009-

020.

7. REFERENCES

- [1] J. Kubiawicz, D. Bindel, Y. Chen, P. Eaton, D. Geels, R. Gummadi, S. Rhea, H. Weatherspoon, W. Weimer, C. Wells, and B. Zhao, "Oceanstore: An Architecture for Global-Scale Persistent Storage," Proc. Ninth Int'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 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 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., 1985.
- [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. Kubiawicz, "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, "Decentralized Erasure Codes for Distributed 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 of Electronics, 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. Networks (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.