TELKOMNIKA Indonesian Journal of Electrical Engineering
Vol. 12, No. 9, September 2014, pp. 6969 ~ 6977
DOI: 10.11591/telkomnika.v12i9.4692

Robust Provable Data Possession Protocol

Chao Feng*[1], Yixian Yang[2], Jing Zhang[3], Yang Xin[2]
[1]School of Information Science and Engineering, Shan Dong University,
Jinan 250100, China
[2]National Engineering Laboratory for Disaster Backup and Recovery,
Beijing University of Posts and Telecommunications, Beijing 100876, China
[3]School of Computer and Information Technology, Beijing Jiaotong University,
Beijing 100044, China
***Corresponding author, e-mail: chaojuan99@hotmail.com**

Abstract

Provable data possession (PDP) is a technique for ensuring the validity of data in storage outsourcing. The main issue is how to frequently, efficiently verify that an untrusted server is correctly storing its client's outsourced data. In this paper, we introduce a robust provable data possession protocol that allows a client that has stored he/her data at an untrusted server to verify the validity of data without retrieving it. The client preprocesses the data and sends it to an untrusted server for storage, while keeping a small amount of meta-data. Then the client generates probabilistic proofs of possession by sampling a random set of blocks from the server to prove that the stored data has not been tampered with or deleted, which drastically reduces I/O costs. In additions, by means of the careful integration of online codes and PDP (O-PDP), the scheme can recovers a small amount of the file, once it has been deleted. Finally, we conduct an experimental evaluation to study the performance, and robustness of O-PDP.

Keywords*: data outsourcing, provable data possession, online-code, robust*

1. Introduction

Outsourcing of data allows the data owner (client) with limited resources moves its data to a remote server (e.g. Cloud Storage Service Providers) which is supposed to correctly store the data and make it available to the client on demand. The mainly advantages of data outsourcing include reduced costs from savings in storage, maintenance and as well as increased availability. Unfortunately, the server that stores the client's data is untrusted. Therefore, the client need verify if their data has been tampered or deleted. How to verify the availability of data has turned into a critical issue in outsourcing data services.

A number of research issues in data outsourcing have been studied in the past twenty years. Early work concentrates on data authentication, i.e., how to efficiently verify that the server returns correct and complete results in response to its clients' queries [1-2]. The following work focused on outsourcing encrypted data and associated difficult problems mainly having to do with efficient querying over encrypted domain [3-6].

Ateniese et al. [7] firstly proposes a model called provable data possession (PDP). In this model, data (represented as a file F) is preprocessed by the client, and the client produces metadata used for verification purposes. The file F and metadata are then sent to an untrusted server, and the client can deletes the local copy of the file and store the metadata locally. By randomly choose a set of block from the file, the client makes a challenge, and then the server compute a proof corresponding to the challenge sent by the client, the client keeps some secret information to verify server's proof later. The author presents several variations of their scheme under different cryptographic assumptions. Juels et al. [8] proposes a model for proofs of retrievability (PORs), which focusing on static archival storage of large files. Their scheme's effectiveness rests largely on preprocessing steps before sending a file F to the server: "sentinel" blocks are randomly inserted to detect corruption, F is encrypted to hide these sentinels, and error-correcting codes are used to recover from corruption. As expected, the error-correcting codes improve the error-resiliency of their system. Unfortunately, the number of queries a client can perform is limited.

Received October 10, 2013; Revised Apr 11, 2014; Accepted May 6, 2014

According to Curtmola [9], a robust provable data possession means that can recovers the data, once it has been deleted. That's to say, to damage a file, an adversary must deletes a larger amount of data, which easily detected. Inspired by Juels [8], Bowers et al. [10] propose an optimized application of FEC codes. They establish the bounds under which a client is able to retrieve its data from the server, meanwhile, integrate error-correcting codes with spot checking can acquire a robust possession guarantee. This concept plays a more prominent role as the remote storage community moves from the theoretical to the practical. The same pairing of erasure coding with data checking has been used by remote storage systems that distribute or replicate data among many servers [11]. Indeed, the proposed use of FEC codes in the frameworks of [10] is not optimal and may lead to reduced performance. Meanwhile, spot checking are probabilistic in nature and cannot detect corruption of small parts of the data (e.g., 1 byte).

In this paper, we propose a robust provable data possession (PDP) protocol that provides probabilistic proof that the remote server stores a file correctly. We focus on the provable data possession (PDP) [7] framework, as being representative for remote data checking based on spot checking. And more importantly, PDP allows easy and immediate integration with online codes to improve the data possession guarantee: A file F is first encoded using an online code and PDP is then applied on the encoded file F (instead of F). The original PDP framework provides the ability to detect if the server corrupts a fraction of F. When combined with an appropriate online code, a PDP scheme can provides the following robust data possession guarantee for the encoded file F:

1) Protection against corruption of a large portion of F: The client will detect with high probability if the server corrupts more than a fraction of F

2) Protection against corruption of a small portion of F: The client will recover the data in F with high probability if the server corrupts at most a fraction of F

This paper's contribution is two-fold. Firstly, we propose a robust provable data possession (O-PDP) that provides proof of possession, which can detect corruption of small parts of the data with high probability. Secondly, through the careful integration of online codes and PDP, the scheme we proposed can recover the data when a small amount of the file has been deleted or missed.

We list the features of our PDP schemes (O-PDP) in Table 1. We also include a comparison of related techniques [7, 9], and [12]. Both schemes in [12] cannot provide a data possession guarantee, but improve storage complexity. Specifically, n: the number of data block in F ; c: the number of request data block.

Table 1. Features of Various PDP Schemes when the Server Misbehaves by Deleting a Fraction of an n-block file (e.g., 1% of n)

schemes	data possession	computational complexity	communication complexity	decoding complexiy	probability testing	robustness
[12]	NO	$O(1)$	$O(1)$	----	NO	NO
[7] E-PDP	Yes	$O(\mathrm{n})$	$O(c)$	----	YES	NO
[9]	Yes	$O(c)$	$O(c)$	$O(n\ln(n))$	YES	YES
O-PDP	Yes	$O(c)$	$O(c)$	$O(n)$	YES	YES

The rest of the paper is organized as follows. In Section 1, we present the notations used throughout the paper and describe the online codes informally. Section 2 introduces our main construction. In Section 3, we discuss its correctness. We support our theoretical claims with experiments that show the performance of our scheme in Section 4 and conclude in Section 5.

2. Spot Checking and Online Codes

In this section, we first describes the "spot checking" algorithm. We can prove data possession with high probability by verifying a small amount of blocks, which obviously improves the performance of scheme. After formalizing the spot checking algorithm, we give an overview of online codes. We refer the reader to [13] for a more detailed exposition. In the end, we give a formally definitons of the robust provable data possession.

2.1. Spot Checking

For an effective solution, the amount of block accesses at the server should be minimized, for the server may be involved in concurrent interactions with many clients. To improve the performance, our construction introduces a technique that allows the client makes a challenge by randomly choosing a subset of blocks, which named as "spot checking". We can prove data possession with high probability based on accessing a fraction of the file, which obviously improves the performance of scheme. Once the server deletes a fraction of the file, the client can detects server misconduct with high probability by verifying a small amount of blocks, independently of the total number of file blocks. As an example, for a file with $n = 100000$ blocks, if S has deleted 1% of the blocks, then C can detects server misconduct with probability more than 99% by asking proof of possession for only 1000 randomly selected blocks. For more details see Section 5.4.

2.2. Online Codes

In this section, we informally describe the online codes, which first proposed by [13]. Starting with a file F of size n data blocks. Online codes are parameterized by the block size and two variables, k and ¢. k and ¢ define the relation between the complexity and performance of the online codes. The authors suggest $k = 3$ and ¢ $= 0.005$. The algorithm consists of three phases.

2.2.1. Preprocess

First the n data blocks are translated into a composite message by appending some auxiliary blocks. Each auxiliary block is the exclusive-OR of some number of the original message blocks.

2.2.2. The Encoding Process

From the composite message, we generate encoding blocks of size $(1+k¢)n$. Encoding blocks are named as *check blocks* below. A check block is the exclusive-OR of i data blocks, which are selected uniformly and independently from a ordered set of all composite blocks; i is the *degree* of one check block. The degree is chosen randomly according to a appropriate probability distribution $p = (p_1, p_2, ...p_L)$, such that degree i is chosen with probability p_i, where $\sum_{1}^{L} p_i = 1$. The L is a constant.

2.2.3. The Decoding Process

For each check block e, all of whose data blocks are recovered, except for one. We call this data block m_x. We have $m_x = e \oplus m_1 \oplus \cdots \oplus m_{i-1}$, where $m_1, ..., m_{i-1}$ are the recovered data blocks that are corresponding to e. Apply this step until no more data blocks can be decoded. Upon receiving a certain number of check blocks some fraction of the composite message can be recovered. The composite message can be used to recover the original message. For more details in [14].

The main result of online codes is the following theorem, which is useful for our construction, especially, for robust.

Theorem 1. For a file F of size n data blocks, and parameters k and ¢, the probability that can recover a $1-¢$ fraction of the original message from check blocks is $(\frac{¢}{2})^{k+1}$.

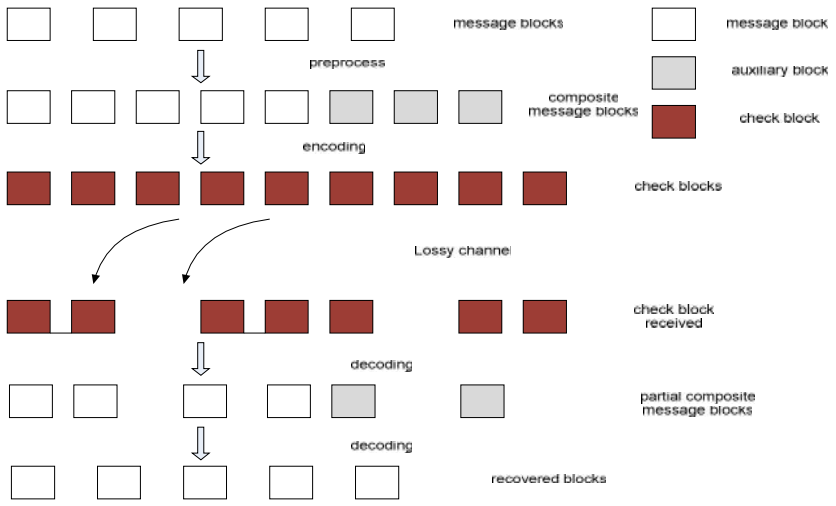

Figure 1. High Level View of Online Codes

2.3. Definition of O-PDP

In this paper, we propose a robust version of provable data possession, which adapts from [7].

Definition 1 Robust provable data possession (O-PDP). An O-PDP scheme is a collection of six polynomial time algorithms (Encode, KeyGen, TagBlock, ChalGen, GenProof, CheckProof) such that:

$Encode(F)$. Takes an original file F as input, and then encode into the encoded file F .

$KeyGen(1^k)$ .Takes the security parameter k as input, output (pk, sk) such that $pk = (N, g)$ and $sk = (e, d, ^3)$.

$TagBlock(pk,$sk$,m_i)$. It takes as inputs a public key pk , a secret key sk and a file block m_i , and returns the verification metadata (T_{m_i}, W_i) .

$ChalGen(c, k_1)$. Takes k_1 as input, k_1 is the key of a pseudo-random permutation ®, and then choose randomly a subset set_{chal} of size c .

$GenProof(c, set_{chal})$. Takes c and set_{chal} as inputs, output the $Ver = (T, °)$.

$CheckProof(pk, sk, set_{chal}, Ver)$. Run by the client to validate a proof of possession. It takes as inputs a public key pk , a secret key sk , a set set_{chal} of size c , and a proof of possession $Ver = (T, °)$. It returns whether Ver is a correct proof of possession for the blocks determined by set_{chal} .

3. The Construction

A robust provable data possession scheme incorporates mechanisms for mitigating arbitrary amounts of data corruption. We consider a notion of mitigation that includes the ability to both efficiently detect data corruption and recover the corrupted blocks. When data corruption is detected, the client can recover data in time. That's to say, a robust scheme ensures that no data will be lost. Formally, we define the robustness of provable data possession scheme as follows:

Definition 2 Robustness. A robust provable data possession scheme $O-PDP$ is a two-tuples (P,O) , where P is a provable data possession scheme for a file F , and O is a coding procedure(online code) that yield F applied on F . The provable data possession scheme is robust, if it has the following two properties: (I) the client can detect w.h.p if the server

corrupts more than a ¨ fraction of F; (II) the client can recover the data F w.h.p if the server corrupt at most a ¨ fraction of F.

The implementation of robust data possession guarantee for F can be described as follows: firstly, the file F is encoded using an online code and PDP is then applied on the encoded file F (instead of F). When combined with an appropriate online code, a PDP scheme can provides a robust data possession guarantee for the encoded file F. Compared with other rateless erasure codes, the decoding complexity of online codes is $O(n)$; meanwhile, the decoding complexity of R-S is $O(n^2)$.

3.1. The Notations

In this section, we introduce some notations used throughout the work. ª is the security parameter, and © is the key of a pseudo-random permutation. Let $p = 2p' + 1$ and $q = 2q' + 1$ be safe primes and let $N = pq$ be an **RSA** modulus. A file F is consist of a finite ordered collection of n blocks: $F = (m_1, m_2, ..., m_n)$. Let g be a generator of QR_N, which is the unique cyclic subgroup of $^*_N$ of order $p'q'$. All exponentiations are performed modulo N.

3.2. The Protocol

The protocol can be divided into four steps. Firstly, we preprocess the tag for each data block m_i of the file F and then store the file F and its tags (T_{m_i}, W_i) with a server S. Secondly, the client C generates a challenge by randomly choose a subset of file. Thirdly, using the challenge, the server S generates a proof of possession. Finally, the client C verifies the validity of the proof. It is worth noting that the client C stores on the server S a file F (instead of F), which is a finite ordered collection of $(1+k¢)n$ blocks: $F = (m_1, m_2, ..., m_{(1+k¢)n})$.

Let ® be a pseudo-random permutation and let h be a cryptographic hash function:

$h: \{0,1\}^{ª} \times \{0,1\}^{\log_2(p'q')} \to \{0,1\}^{\log_2(p'q')}$;

$®: \{0,1\}^{©} \times \{0,1\}^{\log_2(n)} \to \{0,1\}^{\log_2(n)}$

$KeyGen(1^k)$. The keys are generated as the following way. Firstly, choose randomly two distinct safe primes p' and q', compute $p = 2p' + 1$, $q = 2q' + 1$, let $N = pq$ be an **RSA** modulus. Secondly, choose randomly $a \leftarrow_R {}^*_N$ such that $\gcd(a \pm 1, N) = 1$, and compute $g = a^2$. In the end, compute e and d such that $ed \equiv 1 \bmod (p'q')$, where e is a secret prime such that $e > ª$ and $d > ª$, choose randomly $³ \leftarrow_R \{0,1\}^{©}$. Output (pk, sk) such that $pk = (N, g)$ and $sk = (e, d, ³)$.

$TagBlock(pk, sk, m_i)$. Preprocess a tag (T_{m_i}, W_i) for each block m_i of the file. Generate $W_i = ³ \| i$, compute $T_{m_i} = (W_i \cdot g^{h(m_i)})^d \bmod N$, output (T_{m_i}, W_i).

$ChalGen(c, k_1)$. Choose randomly a subset $i_j \in set_{chal}$ of $[1, 2, ..., (1+k¢)n]$, where $1 \le j \le c$, $i_j = ®_{k_1}(j)$, k_1 is the key of a pseudo-random permutation ®, and c is the size of set_{chal}. $chal = (c, k_1, S)$.

$GenProof(c, set_{chal})$. The server generates $i_j = ®_{k_1}(j)$, compute $T = \prod_{j=1}^{c} T_{m_{i_j}}$, $¯ = \prod_{j=1}^{c} W_{i_j} g^{h(m_{i_1}) + ... + h(m_{i_c})} \bmod N$. The proof is $Ver = (T, ¯)$.

$CheckProof(pk, sk, set_{chal}, Ver)$. The client checks the validity of the proof $Ver = (T, ¯)$. Compute ¦ $= T^{e}$, if ¦ $\bmod N = ¯$, then output "success". Otherwise output "failure".

4. Correctness

The scheme is based on the KEA1 assumption which was introduced by Damgard in [14]. In particular, Bellare and Palacio [15] provided a formulation of KEA1, that we adapt to work in the RSA setting.

Theorem 2. According to the KEA1-assumptions [14], the scheme we proposed can correctly verifies the validity of the server's proof for the encoding file F by checking if a certain relation holds between T and ° .

Proof: The client pre-computes $T_{m_i} = (W_i \cdot g^{h(m_i)})^d \bmod N$ for each data block m_i of the file F, then stores the file F and its tags (T_{m_i}, W_i) with a server S. The client generates a random challenge corresponding to a randomly selected set set_{chal}, and then the server S generates a proof $Ver = (T, °)$ for the set set_{chal} in response to the challenge. In the end, the client C verifies the validity of proof. According to the KEA1-assumptions, the server S generates $i_j = ®_{k_1}(j)$, and then compute $T = \prod_{j=1}^{c} T_{m_{i_j}}$, $° = \prod_{j=1}^{c} W_{i_j} g^{h(m_{i_1})+...+h(m_{i_c})} \bmod N$. The proof is $Ver = (T, °)$. The client C has a secret key e, he/her can computes ¦ $= T^e$. If ¦ $\bmod N = °$, then output "success". Otherwise output "failure".

5. Analysis

We measure the performance based on our implementation of O-PDP in Linux. All experiments were conducted on an Intel 2.2 GHz systems, and 2048 MB of RAM. The system runs ubuntu 12.04 TLS. Algorithms use the NTL version 5.5.2 with a modulus N of size 1024 bits and files have 24KB blocks. As a basis for comparison, we have also implemented the scheme of Ateniese et al. [7] which named as E-PDP.

5.1. Preprocess Time

In preparing a file for outsourced storage, the client firstly generates its tags for verify. In this experiment, preprocess time is the time of metadata generation, which does not include the time of loading data to the client and the time of transferring metadata to disk. Figure 2 shows that the pre-processing time as a function of file size for O-PDP and E-PDP. Compared with E-PDP, O-PDP exhibits slower pre-processing performance. In order to generate the per-block tags, O-PDP performs an exponentiation on every file block (including the auxiliary block) of file F rather than F.

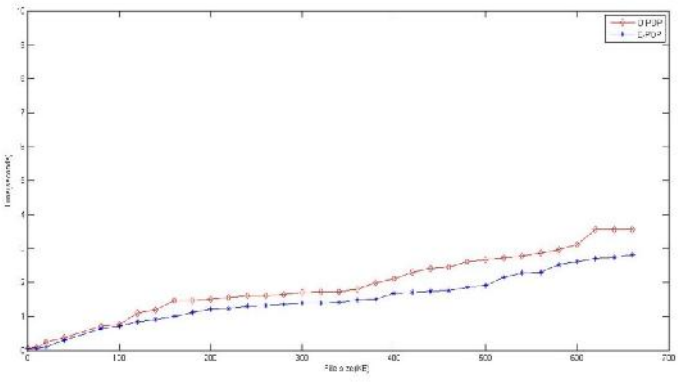

Figure 2. The Comparison of Preprocess Time

5.2. Challenge Time

Figure 3 shows the computation time as a function of block size when computing a proof for E-PDP and O-PDP. Note the logarithmic scale. Computation time includes the time to

access the memory blocks that contain file data in cache. We restrict this experiment to blocks of 768KB or less, because of the amount of block size suggested by E-PDP. E-PDP radically alters the complexity of data possession protocols. For blocks of 768KB, E-PDP is more than 11.5 times faster than O-PDP. For O-PDP performance grows linearly with the file size, because it exponentiates the entire file. According to the result, we can conclude that the integration of online code can make the PDP scheme robust, which also sacrifice the performance of scheme, especially, the challenge time.

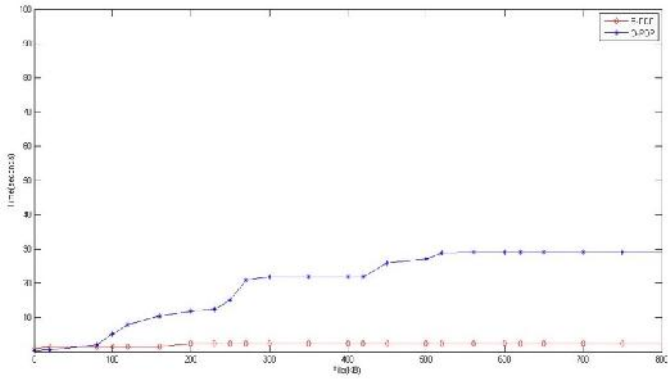

Figure 3. The Comparison of Challenge Time

5.3. Pre-processing vs. Challenge Time with Each Block Size

O-PDP also exponentiates data that was reduced modulo N but does not reap the same speed up, because it must do so for every block. Figure 4 shows that this trade-off indicates that the best balance occurs at natural file system and memory blocks sizes of 20-30 KB. In order to achieve the best balance, we can choose the size of one block is 24KB.

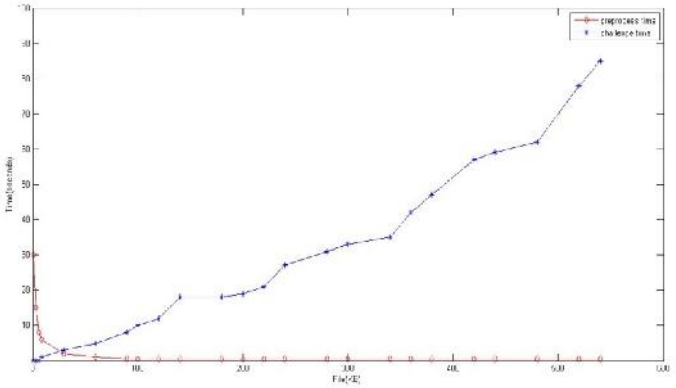

Figure 4. O-PDP's Pre-processing vs. Challenge Time with Block Size for a 16MB File

5.4. Robustness

The robustness of the scheme we proposed including two aspects:

1) The probability that client can detects the data corruption when the server corrupts more than a ¨ fraction of F. Our scheme allows that the server can proves possession of selected blocks of F. Spot checking greatly reduces the workload on the server, while still achieving detection of server misbehavior with high probability. We now analyze the probability as follows. Assume S deletes $t = n¨$ blocks out of the n-block file F. Let c be the number of different blocks for which C asks proof in a challenge. Let X be a discrete random variable that is defined to be the number of blocks chosen by C that match the blocks deleted by S. We

compute P_X, the probability that at least one of the blocks picked by C matches one of the blocks deleted by S. We have:

$$P_X = P\{X \geq 1\} = 1 - P\{X = 0\} = \prod_{i=0}^{c-1} 1 - \frac{n-i-t}{n-i} \quad (1)$$

Since,

$$\frac{n-i-t}{n-i} \geq \frac{n-i+1-t}{n-i-1} \quad (2)$$

We have:

$$1-(\frac{n-t}{n})^c \leq P_X \leq 1-(\frac{n-c+1-t}{n-c+1})^c \quad (3)$$

3) The probability of the data can be restored, which have been corrupted, but have not been detected. In the construction, c blocks were chosen randomly from the encoded files. We assume that the ratio of corrupted data block is ¨ (¨ $= 0.01$). Figure 5 shows that, the detection probability is more than 0.99. According to the previous parameters k=3,¢=0.005, and theorem 1, we conclude that the probability that data can be restored is more than 0.9999.

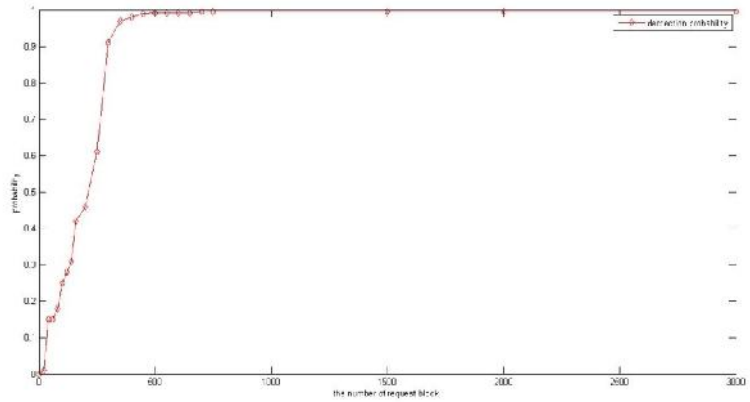

Figure 5. O-PDP Detection Probability vs. the Number of Request Blocks

6. Conclusion

In this paper, we propose a robust provable data possession protocol. Key component of our scheme is the careful integration of online codes and PDP. By introduce the online codes; the protocol can verify data possession without access the actual data file, and recovery data when it was corrupted. Experiments show that our scheme, which offers a probabilistic possession guarantee by sampling the server's storage, is practical to verify possession of large data sets. Given the practical importance of the online codes and the emergence of storage outsourcing service, we believe the study of robust provable data possession to be important and well motivated.

Acknowledgements

I would like to express my thanks and appreciation to my Doc adviser.Yixian Yang, for his encouragement and guidance in completing this work. This work is supported by the National Natural Science Foundation of China (Grant No. 61121061, 61161140320).

References

[1] Devanbu P, Gertz M, Martel C, *et al*. Authentic third-party data publication. *Journal of Computer Security*. 2003; 11(3): 291-314.

[2] Mykletun E, Narasimha M, Tsudik G. Authentication and integrity in outsourced databases. *ACM Transactions on Storage*. 2006; 2(2): 107-138.

[3] Hakan H, Bala L, Chen L, et al. *Executing sql over encrypted data in the database-service-provider model*. Proceedings of SIGMOD' 02. Madison. 2002: 216-227.

[4] Song D, Wagner D, Perrig A. *Practical techniques for searches on encrypted data*. Proceedings of IEEE S&P'00. Berkeley. 2000: 44-55.

[5] Boneh D, Di Crescenzo G, Ostrovsky R, et al. *Public key encryption with key-word search*. Proceedings of Eurocrypt 2004. Interlaken. 2004: 506-522.

[6] Golle P, Staddon J, Waters B. S*ecure conjunctive keyword search over encrypted data*. Proceedings of the Second International Conference ACNS 2004. Yellow Mountain. 2004: 31-45.

[7] Ateniese G, Burns R, Curtmola R, et al. *Provable data possession at untrusted stores*. Proceedings of ACM Conference on Computer and Communications Security. Alexandria. 2007: 598-609.

[8] Juels A, Kaliski B S. PoRs: proofs of retrievability for large files. Proceedings of ACM Conference on Computer and Communications Security. Alexandria. 2007: 584 -597.

[9] Curtmola R, Khan O, Burns R. *Robust remote data checking*. Proceedings of the 4th ACM international workshop on Storage security and survivability. New York. 2008: 63-68.

[10] Bowers K, Juels A, Oprea A. Proofs of retrievability: Theory and implementation. ePrint Archive Report, (2008/175), 2008.

[11] Kotla R, Alvisi L, Dahlin M. *Safestore: A durable and practical storage system*. In USENIX Annual Technical Conference. Santa Clara. 2007: 129-142.

[12] Golle P, Jarecki S, Mironov I. *Cryptographic primitives enforcing communication and storage complexity*. Proceeding of the 6th International Conference on Financial Cryptography. Bermuda. 2002: 120–135.

[13] Maymounkov P. Online codes. New York University, TR2003-883, 2003.

[14] Damgard I. *Towards practical public key systems secure against chosen ciphertext attacks*. Proceedings of the 11th Annual International Cryptology Conference on Advances in Cryptology. London. 1992: 445-456.

[15] Bellare M, Palacio A. *The knowledge-of-exponent assumptions and 3-round zero-knowledge protocols*. Proceedings of the CRYPTO '04, Lecture Notes in Computer Science. Santa Barbara. 2004: 273-289.