International Journal of Security and Its Applications
Vol.8, No.5 (2014), pp.97-108
http://dx.doi.org/10.14257/ijsia.2014.8.5.10

An Efficient Provable Data Possession based on Elliptic Curves in Cloud Storage

Hongyuan Wang [1], Liehuang Zhu* [1], Feng Wang [2], Yijia Lilong [1], Yu Chen [1], Chang Liu [1)]

[1]*Beijing Engineering Research Center of Massive Language Information Processing and Cloud Computing Application, School of Computer Science and Technology, Beijing Institute of Technology, Beijing,100081, China*
[2]*Naval Academy of Armament, Beijing, 100161, China*
E-mail: wanghongyuan@bit.edu.cn, liehuangz@bit.edu.cn, lionkingwf@hotmail.com, lilongyijia@bit.edu.cn, chenyulonger1@bit.edu.cn, changliu.bit@gmail.com

Abstract

With the widespread use of cloud computing and cloud storage, how to ensure the authenticity of data in remote storage is becoming a severe problem. Provable data possession (PDP) is a technique for a client to verify whether an untrusted server possesses the original data. Moreover, for the requirement of efficiency and instantaneity, clients need to verify the authenticity of stored data without having the server retrieving and accessing the entire file.

There are many PDP models to resolve these issues. In 2007, G. Ateniese et al. firstly defined a sampling model named S-PDP, which is regarded as "state-of-the-art" in this field. In terms of efficiency, we found that S-PDP can be improved by utilizing elliptic curve cryptography. Based on this observation, we propose a more efficient scheme which we call EC-PDP. We also implement EC-PDP and compare the performance with S-PDP. In our scheme, the metadata of the client is reduced by 84%, and the computation complexity which consists of the computational cost of pre-processing, generating a proof and verification is reduced by 13% approximately.

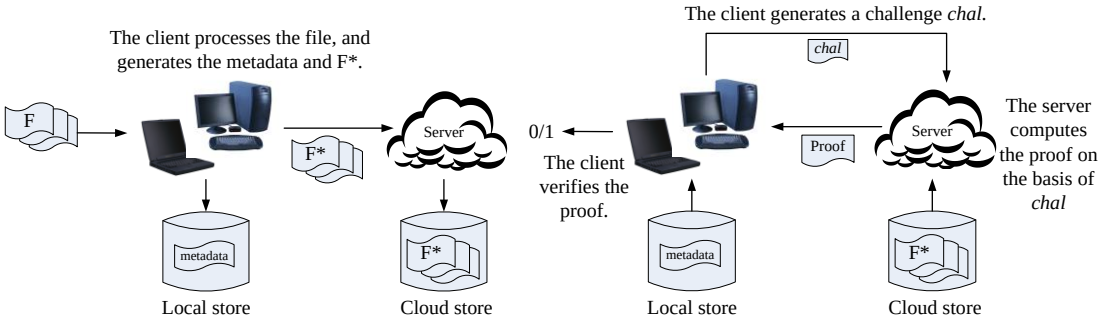
Keywords: *Provable data possession, Elliptic curve, Homomorphic verifiable tag, Provable security*

1. Introduction

With the development of computer technology, cloud computing has become a hot spot in the field of information science. Even though cloud storage has the advantage of high-capacity, low-pay, no limitation of places and time, availability, maintainability and scalability *etc.*, there are still many threatens from either internal or external. Verifying the authenticity of stored data is becoming a critical issue which attracts widespread attention.

Many systems prevent storage servers from modifying or losing data by verifying the authenticity of stored data, which named data possession. In general, a PDP protocol consists of two phases as Figure 1. In the phases of pre-process, the client pre-processes the file, and generates some metadata which is stored locally, then transmits the file and partial metadata to the cloud with deleting its local copy. In the phases of challenge and verification, the server generates a proof of file blocks determined by the challenge from the client, and then the client verifies the validity of the proof.

ISSN: 1738-9976 IJSIA
Copyright ⓒ 2014 SERSC



(1) The phases of pre-process (2) The phases of challenge and verification

Figure 1. Two Phases of a PDP Protocol

But it is insufficient to provide authenticity checks when accessing the data, because it is too late to recover the modified or lost data. Furthermore, to accessing the entire file is expensive in costs because that cloud storage server preserves large amounts of data which may be stored at remote sites.

We conclude that the clients need to verify the authenticity of stored data without having the server retrieving and accessing the entire file.

G. Ateniese et al. [1, 16] firstly meet these requirements for prove data possession. They define a model for provable data possession (PDP) that provides probabilistic proof generated by the storage server. The S-PDP and E-PDP proposed by G. Ateniese et al. are all based on RSA cryptography scheme, which has exponential calculation.

Our PDP schemes are based on elliptic curve cryptography (ECC) which applies additive operation to reduce computational cost. In our schemes, the client only stores an $O(1)$ amount of metadata, and the required bandwidth is also $O(1)$. The computation complexity and communication complexity are both declined.

2. Related Work

Memory checking protocols [5, 6] verify all reads and writes to a remote memory. For PDP is a restricted form of memory checking, it is much more difficult and expensive than proving data possession.

Golle, *et al.*, [7] were the first to propose a scheme that enforces storage complexity, based on a new assumption (n-Power Decisional Diffie-Hellman). It means that a server retains the information at least as large as the file received from the client, but not necessarily the original file. Golle et al. also briefly mention a scheme suggested by David Wagner, based on Merkle hash trees, which lowers the computational complexity of the server but at the expense of increased communication.

Deswarte, *et al.*, [8] and Filho, *et al.*, [9] use RSA-based hash functions to verify that a remote server stores data, and allows a client to perform multiple challenges using the same metadata. The shortcoming is the server's computation complexity, which need access all of the file's blocks and exponentiate the entire file.

Oprea, *et al.*, [10] propose a scheme based on tweakable block ciphers which allow a client to detect the modification of data. It does not require additional storage at the server, but requires the entire file to be retrieved, which means the file access and communication complexity are both linear of the data size per challenge.

Schwarz and Miller [11] propose a scheme to verify the storage of m/n erasure-coded data

 Copyright ⓒ 2014 SERSC

across multiple sites. But the shortcoming in this scheme is even more. The computation complexity at the server and the communication complexity are all linear of file blocks per challenge. Additionally, the security of the scheme is not proven.

Sebe, *et al.*, [12] give a protocol for integrity checking of remote file, which based on the Diffie-Hellman problem in Z_N. But in this scheme, the storage on the client is $O(n)$. Moreover, it requires the server to access the entire file to check integrity.

Yamamoto, *et al.*, [13] use the similar techniques to check data integrity by batch verification of homomorphic hash functions, which have been used in source authentication. Krohn, *et al.*, [14] give a good review of homomorphic hashing in source authentication. They compose multiple blocks into a single value by homomorphism. However, Source authentication techniques do not apply to provable data possession.

G. Ateniese, *et al.*, [16] firstly define a sampling model for provable data possession (PDP) that verify the authenticity of stored data without having the server retrieving and accessing the entire file. It provides probabilistic proof generated by the storage server. But the schemes are all based on RSA cryptography scheme, which has exponential calculation.

C.C. Erway, *et al.*, [17] propose a dynamic provable data possession model to support provable updates to stored data. B. Chen, *et al.*, [18] propose an efficient RDC model that relies on spot checking for both static and dynamic schemes.

Y. Zhu, *et al.*, [19, 20] propose a cooperative PDP (CPDP) scheme based on homomorphic verifiable response and hash index hierarchy to support scalability of service and data migration in hybrid cloud.

C. Hanser, *et al.*, [4] propose the first simultaneous privately and publicly verifiable PDP protocol based on elliptic curve (EC). It uses the same pre-process and the same metadata to achieve private and public verifiability. Though it is efficient than previous schemes, it has too many complex operations to practice for application.

Similar to PDP, Juels and Kaliski [15, 23] introduce the notion of proof of retrievability (POR), which allows a server to retrieve a file that was previously stored. The POR scheme [21, 22] uses sentinels hidden among regular file blocks in order to detect data modification. But the POR scheme can only be applied to encrypted files and only perform a limited number of challenges, which equal to the number of sentinels.

3. Provable Data Possession Schemes by EC (Elliptic Curve)

3.1 Preliminaries

3.1.1 Homomorphic Verifiable Tags (HVTs)

Before introducing the concept of homomorphic verifiable tags, we review the definition of homomorphic encryption, which will be used in our scheme.

Definition 1: (Homomorphic Encryption [3]) We say that a homomorphic encryption scheme ε is correct for circuits in C_ε if, for any key-pair (sk, pk) output by $KeyGen(1^\lambda)$, any circuit $C \in C_\varepsilon$, any plaintexts $m_1, \cdots, m_n$, and any ciphertext $\phi = \{\varphi_1, \cdots, \varphi_n\}$ with $\varphi_i \leftarrow Encrypt(pk, m_i)$, it is the case that:

$$\text{If } \Omega \leftarrow Evaluate(pk, C, \phi), \text{ then } Decrypt(sk, \Omega) \rightarrow C(m_1, \cdots, m_n).$$

The concept of homomorphic verifiable tags is mentioned by G. Ateniese, *et al.*,[1, 16] and they will be used to generate proof of possession in our schemes.

Given a file block *m*, let T_m be its homomorphic verifiable tag. These tags will be stored together with the file F on the server, and act as verification metadata for the file blocks. They

have the properties consist of unforgeability, homomorphism and blockless verification.

3.1.2 The Definition of a PDP Scheme

Definition 2: (provable data possession scheme [1, 16]) A PDP scheme is a collection of four polynomial-time algorithms (KeyGen, TagGen, GenProof, VerifProof) as follows:

$KeyGen(1^{\lambda}) \rightarrow (pk, sk)$ is a probabilistic algorithm run by the client to generate the public key and private key. It takes a security parameter *k* as input, and returns a pair of keys (pk, sk).

$TagGen(pk, sk, m) \rightarrow T_m$ is an algorithm run by the client to generate the metadata which used to verify the proof. It takes the public key *pk*, the private key *sk* and a file block *m* as input, and returns the corresponding verification tags T_m.

$GenProof(pk, F, chal, \Sigma) \rightarrow \rho$ is an algorithm run by the server to generate a proof of possession. The input consists of a public key *pk*, an ordered collection *F* of blocks, a challenge *chal* and an ordered collection Σ of tags which generated in *TagGen* to verify the possession proof. It returns a proof ρ for the blocks that determined by the challenge *chal*.

$VerifProof(pk, sk, chal, \rho) \rightarrow \{"success", "failure"\}$ is an algorithm run by client to verify a proof of possession. It takes the public key *pk*, the private key *sk*, a challenge *chal* and a proof of possession ρ as input, and returns whether ρ is a correct proof for the blocks determined by *chal*.

A PDP system is constructed from a PDP scheme in two phases:

Pre-process: the client *C* runs *KeyGen* and *TagGen*, and stores (*pk*,*sk*). Then *C* sends *pk*, *F* and Σ to the server *S* to store and deletes *F* and Σ from its local storage.

Challenge and verify: *C* generates a challenge *chal* and sends it to *S*. Then *S* runs *GenProof* and sends the proof ρ to *C*. Finally, *C* checks the validity of the proof ρ by running *VerifProof*.

The second phase can be executed many times in order to insure whether *S* still possesses the file blocks.

3.2 EC-PDP

The scheme is based on the ECDLP assumption, which is believed to be more difficult than the general discrete logarithm problem, and in fact, no one has come up with an efficient algorithm to solve it.

Definition 3: (The elliptic curve discrete logarithm problem (ECDLP) [2]) given an elliptic curve *C* defined over flied F_q and two points $P, Q \in C$, find an integer *x* such that $Q = xP$.

In order to evaluate the level of data possession, we give the definition of strong guarantee of PDP.

Definition 4: (strong guarantee of PDP) we say a PDP scheme with strong guarantee, when the proof of possession is valid if and only if that the server possesses each one of the *c* blocks in the challenge.

EC-PDP meets the requirement of strong guarantee of PDP. The detailed scheme is as follows.

 Copyright ⓒ 2014 SERSC

Let *h* and *f* be a pseudo-random function, let *per* be a pseudo-random permutation, and let H be a cryptographic hash function.

$h:\{0,1\}^{*} \rightarrow \{0,1\}^{|m|}$

$f:\{0,1\}^{\lambda} \times \{0,1\}^{\log_2(n)} \rightarrow \{0,1\}^{\ell}$

$per:\{0,1\}^{\lambda} \times \{0,1\}^{\log_2(n)} \rightarrow \{0,1\}^{\log_2(n)}$

The elliptic curve *C* is defined over a finite field F_q where $q = p^n$ is large (and *p* is prime). Let λ and ℓ be a security parameter.

$\boldsymbol{KeyGen}(1^{\lambda}) \rightarrow (pk, sk)$:

1. Let a point $G(x,y)(x,y \in F_q)$ on *C* be the basis point.
2. Let $d \xleftarrow{R} \{0,1\}^{\lambda}$ be the secret key, and let $P(x,y) = d \cdot G(x,y)$ be the public key.
3. Output (pk, sk).

$\boldsymbol{TagGen}(pk, sk, m) \rightarrow T_m$:

1. Let $d = sk$ and $P(x,y) = pk$.
2. Let $k \xleftarrow{R} \{0,1\}^{\lambda}$ be a coefficient, and *i* be the unique index of each file block.
3. Compute $G'(x,y) = k \cdot G(x,y)$ and $P_i'(x,y) = k \cdot P(x,y) + m_i + h(i)$.
4. Output $T_{m_i} = P_i'(x,y)$ and $G'(x,y)$ (note that $G'(x,y)$ is send only once).

$\boldsymbol{GenProof}(pk, F, chal, \Sigma) \rightarrow \rho$:

1. Let $F = (m_1, \cdots, m_n)$ and $\Sigma = (T_{m_1}, \cdots, T_{m_n})$.
2. Let $P(x,y) = pk$ and $(c, k_1, k_2) = chal$.
3. For $1 \le j \le c$, compute the indices of blocks sampled to generate proof: $i_j = per_{k_1}(j)$, and compute the coefficients: $a_j = f_{k_2}(j)$.
4. Compute: $G_c'(x,y) = (a_1 + \cdots + a_c) \cdot G'(x,y) = k \cdot G(x,y) \cdot (a_1 + \cdots + a_c)$.
5. Compute: $P_c'(x,y) = a_1 \cdot T_{m_{i_1}} + \cdots + a_c \cdot T_{m_{i_c}} = a_1 \cdot P_{i_1}'(x,y) + \cdots + a_c \cdot P_{i_c}'(x,y)$
   $= k \cdot P(x,y) \cdot (a_1 + \cdots + a_c) + a_1 m_{i_1} + \cdots + a_c m_{i_c} + a_1 h(i_1) + \cdots + a_c h(i_c)$.
6. Compute: $v = H(a_1 m_{i_1} + \cdots + a_c m_{i_c})$.
7. Output $\rho = \{G_c'(x,y), P_c'(x,y), v\}$.

$\boldsymbol{VerifProof}(pk, sk, chal, \rho) \rightarrow \{"success", "failure"\}$:

1. Let $d = sk$ and $(c, k_1, k_2) = chal$.
2. Compute $\{i_1, \cdots, i_c\} \Leftarrow per_{k_1}(1, \cdots, c)$ and $\{a_1, \cdots, a_c\} \Leftarrow f_{k_2}(1, \cdots, c)$.
3. Compute: $\delta = P_c'(x,y) - d \cdot G_c'(x,y) - \{a_1 h(i_1) + \cdots + a_c h(i_c)\}$
   $= \{k \cdot P(x,y) \cdot (a_1 + \cdots + a_c) + a_1 m_{i_1} + \cdots + a_c m_{i_c} + a_1 h(i_1) + \cdots + a_c h(i_c)\}$
   $- d \cdot \{k \cdot G(x,y) \cdot (a_1 + \cdots + a_c)\} - \{a_1 h(i_1) + \cdots + a_c h(i_c)\}$
   $= a_1 m_{i_1} + \cdots + a_c m_{i_c}$
4. If $H(\delta) = v$, then output "*success*", otherwise "*failure*".

Theorem 1: Under the ECDLP assumption, EC-PDP guarantees data possession in the random oracle model.

Copyright ⓒ 2014 SERSC

Proof overview: under the ECDLP assumption, we reduce the security of scheme 1 to the security of the ECC problem, and then reduce it to the difficulty of discrete logarithm problem. We assume there exists an adversary A that wins the data possession game, then there must exist an adversary B that can break the security of ECC and the work out the discrete logarithm problem.

3.3 EC-PDP with Weaker Guarantee

Scheme 2 is more efficient but offering weaker guarantees. Scheme 1 can guarantee that the server possesses each one of the *c* blocks in the challenge if the proof is valid, while scheme 2 only guarantees the possession of the sum of the blocks rather than each one of the blocks in the challenge. Even so, scheme 2 is secure and practical.

If the server pre-computes and stores the sums of all possible combinations of *c* blocks, the server could successfully forge any valid proof. But it is infeasible for the server to misbehave by this way. For example, if the total number of the file blocks is *n*=1000 and the challenge number of file blocks is *c*=100, then the number of all the possible values is $C_{1000}^{100} \approx 10^{140}$. Maybe it is better for server to attempt to find the modulus.

In scheme 2, all the coefficients a_i are equal to 1, as follows:

1. In ***GenProof***, the server compute:

$$G_c'(x, y) = c \cdot G'(x, y) = k \cdot G(x, y) \cdot c$$

$$P_c'(x, y) = T_{m_{i_1}} + \cdots + T_{m_{i_c}} = P_{i_1}'(x, y) + \cdots + P_{i_c}'(x, y) = k \cdot P(x, y) \cdot c + m_{i_1} + \cdots + m_{i_c} + h(i_1) + \cdots + h(i_c)$$

$$v = H(m_{i_1} + \cdots + m_{i_c})$$

2. In ***VerifProof***, the client compute:

$$\delta = P_c'(x, y) - d \cdot G_c'(x, y) - \{h(i_1) + \cdots + h(i_c)\} = m_{i_1} + \cdots + m_{i_c}$$

Scheme 2 reduces the computation on both the client and the server.

It is clear that scheme 2 guarantees weak data possession in the random oracle model under the ECC assumption.

4. System Implementation and Performance Evaluation

The important performance parameters of a PDP scheme cover the computation complexity, communication complexity and block access complexity.

The computation complexity includes the computational cost of pre-processing at *C*, generating a proof of possession at *S* and verifying a proof at *C*.

The communication complexity indicates the amount of data transferred between *S* and *C*.

The block access complexity indicates the number of file blocks that *S* has to access to generate a proof of possession.

We compare the performance of various PDP schemes as Table 1:

 Copyright ⓒ 2014 SERSC

Table 1. The Compare of the Performance of Various PDP Schemes

	[7] SC	MHT-SC	B-PDP	[11]	[12]	S-PDP	E-PDP	Scheme1	Scheme2
data possession	No	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes
support sampling	No	No	No	No	No	Yes	Yes	Yes	Yes
type of PDP	deterministic					probabilistic			
pre-processing	$O(n)$	$O(n)$	$O(n)$	$O(n)$	$O(n)$	$O(n)$	$O(n)$	$O(n)$	$O(n)$
generating a proof	$O(n)$	$O(1)$	$O(n)$	$O(n)$	$O(1)$	$O(1)$	$O(1)$	$O(1)$	$O(1)$
verifying	$O(1)$	$O(1)$	$O(1)$	$O(1)$	$O(1)$	$O(1)$	$O(1)$	$O(1)$	$O(1)$
block access	$O(n)$	$O(\log n)$	$O(n)$	$O(n)$	$O(n)$	$O(1)$	$O(1)$	$O(1)$	$O(1)$
communication	$O(1)$	$O(\log n)$	$O(1)$	$O(n)$	$O(1)$	$O(1)$	$O(1)$	$O(1)$	$O(1)$
client storage	$O(1)$	$O(1)$	$O(1)$	$O(1)$	$O(n)$	$O(1)$	$O(1)$	$O(1)$	$O(1)$

4.1 Implementation

All experiments were conducted on an Intel 2.30 GHz Core i7-3610QM IV system with 8192MB of RAM. The system runs windows7 service pack1. Algorithms use the crypto library of OpenSSL version 0.9.8b and files have 4KB blocks. Experiments store files on a NTFS on a Samsung 840EVO MZ-7TE120 120GB SSD. All experiments are conducted 20 times averagely because results varied little across trials.

4.2 Security

The security parameter sizes of 1024 bits for RSA and 160 bits for ECC can achieve the same level of security. On this account, our schemes reduce the computational complexity.

As for the accuracy, our PDP schemes adopt sampling method [1, 16], which achieves detecting misbehavior with high probability.

Let n be the total number of file blocks on the server S, t be the number of the modified or deleted file blocks, and c be the number of challenging file blocks chosen by client C.

Let X be the number of file blocks that chosen by C and match the file blocks modified or deleted. Let P_X be the probability that at least one modified or deleted file blocks are chosen by C, i.e. the probability that C detects the misbehavior of S.

We have that:

$$P_X = P\{X \geq 1\} = 1 - P\{X = 0\}$$

$$= 1 - \frac{n-t}{n} \cdot \frac{n-1-t}{n-1} \cdot \frac{n-2-t}{n-2} \cdots \cdot \frac{n-c+1-t}{n-c+1}$$

We can get: $1 - \left(\frac{n-t}{n}\right)^c \leq P_X \leq 1 - \left(\frac{n-c+1-t}{n-c+1}\right)^c$.

It is interesting that the client can detect the misbehavior of server by asking proof of constant number of blocks with a high probability. In experiments, let $t = 1\%$ of n, and P_X is at least 99%, and then the number of challenge blocks is 460.

4.3 Computation Complexity

In the evaluation of computation complexity, we emphasize on B-PDP [8, 9], S-PDP [1, 16] and EC-PDP.

4.3.1 Pre-processing Performance

In the phases of pre-processing, the client generates the metadata for generating and

Copyright ⓒ 2014 SERSC

verifying proof of possession. In this experiment, we measure the time of the pre-process of different PDP schemes, *i.e.*, the time of generate keys and verification tags.

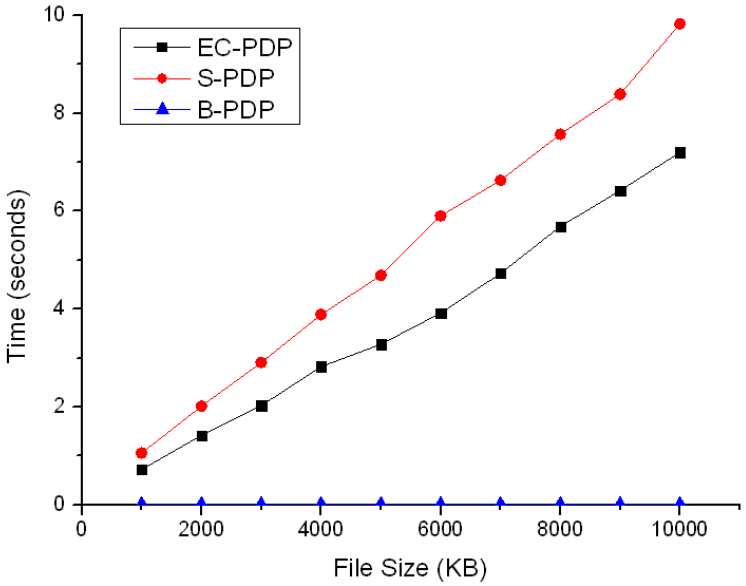


Figure 2. Performance of Pre-Processing

4.3.2 Challenge Performance

In the phases of challenge, the client generates a challenge *chal* and sends it to the server, and then the server generates a proof of possessing file blocks determined by the *chal*. In this experiment, we compare the time of challenge of different schemes.

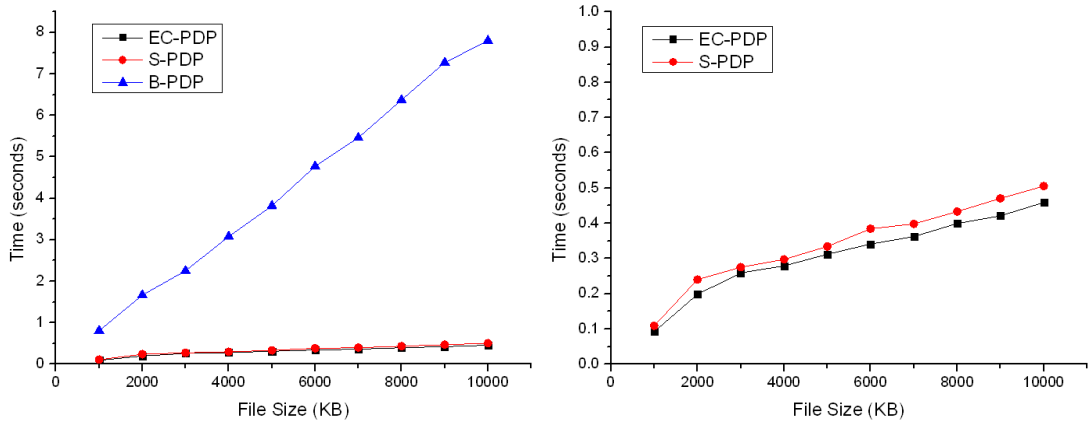


Figure 3. Performance of Challenge

4.3.3 Verification Performance

In the phases of verification, the client verifies the proof of possession given by the server. In the experiment, we compare the time of verifying a same proof of possession in different schemes.

Copyright ⓒ 2014 SERSC

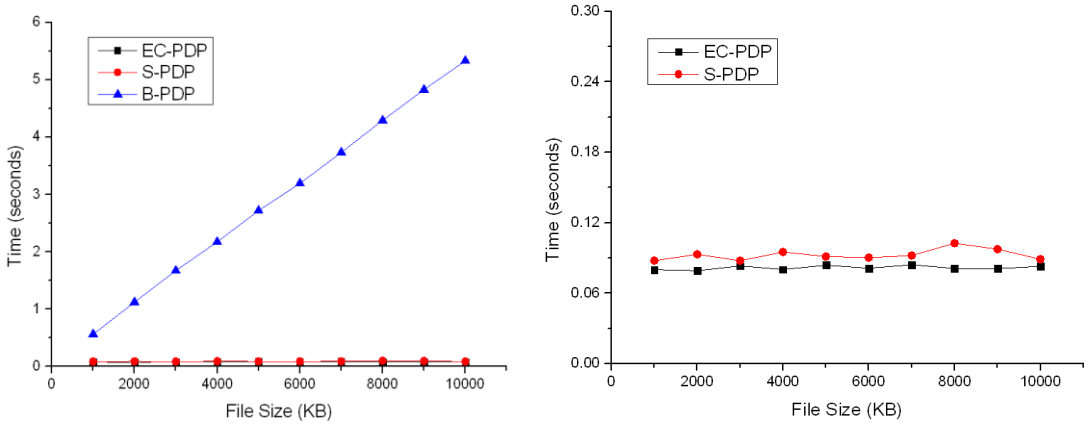


Figure 4. Performance of verification

4.4 Communication Complexity

In our schemes, the required bandwidth is also *O(1)*, because the *chal* and the proof is constant. Compared with S-PDP and E-PDP, challenge and proof of our schemes is reduced. Because of high efficiency and low bandwidth, our schemes are suitable for an infinite number of challenges.

4.5 Block Access Complexity

We utilize sampling method to reduce the block access complexity. To assess the performance of sampling method, we compare the performance of different PDP schemes for detecting 1% modified or missing data at 99% confidence.

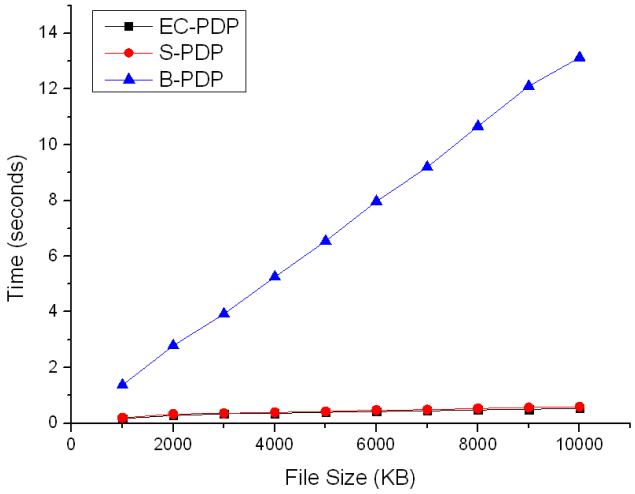


Figure 5. Performance of Sampling

As the Figure show, the time of examining all blocks is linear of the file size, while the time of our PDP scheme is not related with the size of file.

5. Conclusion

In this paper, we focus on the problem how to allow a client to verify that an untrusted

Copyright ⓒ 2014 SERSC

severs possesses the original data, with the requirement of efficiency and instantaneity, *i.e.*, to the greatest extent to reduce block access complexity, communication complexity and computational complexity on both server and client.

Our PDP scheme is based on elliptic curves and is provable secure in the random oracle model. We utilize sampling method to minimize the block access complexity and computational complexity. In our schemes, the client only maintain an *O(1)* amount of metadata to verify the proof of possession, and the required bandwidth is also *O(1)*.

We present two PDP schemes that one guarantees strong data possession, and the other is more efficient with weaker guarantee. Then we implement the schemes, and the experiments show that our schemes are practical and efficient to verity possession of large data on remote storage.

Acknowledgements

The paper is supported by DNSLAB，China Internet Network Information Center， Beijing 100190, National Science Foundation of China under Grant No. 61100172 and No. 61272512, National 863 Plans Projects No. 2013AA01A214, Program for New Century Excellent Talents in University (NCET-12-0046), and Beijing Natural Science Foundation No. 4121001.

Reference

[1] G. Ateniese, R. C. Burns, R. Curtmola, J. Herring, L. Kissner, Z. N. J. Peterson and D. X. Song, "Provable data possession at untrusted stores", in Proc. ACM Conference on Computer and Communications Security, **(2007)**, pp. 598-609.
[2] E. Brow, "Elliptic Curve Cryptography", Math 189A: Algebraic Geometry, **(2010)** December.
[3] C. Gentry, "Fully homomorphic encryption using ideal lattices", In Proc. STOC, **(2009),** pp. 169-178.
[4] C. Hanser and D. Slamanig, "Efficient Simultaneous Privately and Publicly Verifiable Robust Provable Data Possession from Elliptic Curves", presented at IACR Cryptology ePrint Archive, **(2013)**, pp. 392-392.
[5] M. Blum, W. Evans, P. Gemmell, S. Kannan and M. Naor, "Checking the correctness of memories", In Proc. of the FOCS '95, **(1995)**.
[6] M. Naor and G. N. Rothblum. "The complexity of online memory checking", In Proc. of FOCS, Full version appears as ePrint Archive Report 2006/091, **(2005).**
[7] P. Golle, S. Jarecki and I. Mironov, "Cryptographic primitives enforcing communication and storage complexity", In Financial Cryptography, **(2002),** pp. 120–135.
[8] Y. Deswarte, J.-J. Quisquater and A. Saidane, "Remote integrity checking", In Proc. of Conference on Integrity and Internal Control in Information Systems (IICIS'03), **(2003)** November.
[9] D. L. G. Filho and P. S. L. M. Baretto, "Demonstrating data possession and uncheatable data transfer", IACR ePrint archive, Report 2006/150, http://eprint.iacr.org/2006/150, **(2006).**
[10] A. Oprea, M. K. Reiter and K. Yang, "Space-efficient block storage integrity", In Proc. of NDSS'05, **(2005)**.
[11] T. S. J. Schwarz and E. L. Miller, "Store, forget, and check: Using algebraic signatures to check remotely administered storage", In Proceedings of ICDCS '06. IEEE Computer Society, **(2006)**.
[12] F. Sebe, A. Martinez-Balleste, Y. Deswarte, J. Domingo-Ferrer and J.-J. Quisquater, "Time-bounded remote file integrity checking", Technical Report 04429, LAAS, **(2004)** July.
[13] G. Yamamoto, S. Oda and K. Aoki, "Fast integrity for large data", In Proc. of SPEED '07, **(2007)**.
[14] M. N. Krohn, M. J. Freedman and D. Mazi`eres, "On-the-fly verification of rateless erasure codes for efficient content distribution", In Proceedings of the IEEE Symposium on Security and Privacy, **(2004)**.
[15] A. Juels and B. S. Kaliski, "PORs: Proofs of retrievability for large files", Cryptology ePrint archive, Report 2007/243, **(2007)** June.
[16] G. Ateniese, R. C. Burns, R. Curtmola, J. Herring, O. Khan, L. Kissner, Z. N. J. Peterson and D. Song, "Remote data checking using provable data possession", In Proceedings of ACM Trans. Inf. Syst. Secur., **(2011),** pp. 12-12.
[17] C. C. Erway, A. Küpçü, C. Papamanthou and R. Tamassia, "Dynamic provable data possession", In Proceedings of ACM Conference on Computer and Communications Security, **(2009)**, pp. 213-222.
[18] B. Chen and R. Curtmola, "Robust Dynamic Provable Data Possession", In Proceedings of ICDCS Workshops, **(2012),** pp. 515-525.
[19] Y. Zhu, H. Wang, Z. Hu, G. Ahn, H. Hu and S. S. Yau, "Efficient provable data possession for hybrid clouds", In Proceedings of ACM Conference on Computer and Communications Security, **(2010)**, pp. 756-758.

 Copyright ⓒ 2014 SERSC

[20] Y. Zhu, H. Hu, G. Ahn and M. Yu, "Cooperative Provable Data Possession for Integrity Verification in Multicloud Storage", In Proceedings of IEEE Trans. Parallel Distrib. Syst., **(2012)**, pp. 2231-2244.
[21] Y. Zhu, H. Wang, Z. Hu, G. Ahn and H. Hu, "Zero-knowledge proofs of retrievability", In Proceedings of SCIENCE CHINA Information Sciences, **(2011)**, pp. 1608-1617.
[22] K. D. Bowers, A. Juels and A. Oprea, "Proofs of retrievability: theory and implementation", In Proceedings of NS Simulator for Beginners, **(2009)**, pp. 43-54.
[23] A. Juels, B. S. Kaliski Jr., K. D. Bowers, *et al.*, "Proof of retrievability for archived files", U.S. Patent 8,381,062, pp. 013-2-19.

Authors

Hongyuan Wang, she was born in 1988, Ph.D. candidate. Her research interests include cloud computing security, provable data possession and big data privacy.

FengWang, born in 1977, Ph.D., senior engineer. His research interests include low-power networking, sensor networks and Internet of Things technology and applications.

Liehuang Zhu, born in 1976, Ph.D., professor. His research interests include the security analysis of cryptographic protocol, security of internet of things and cloud computing security.

Yijia Lilong, born in 1991, Master Degree Candidate. His research interests include security in cloud computing and provable data possession.

Yu Chen, born in 1988, Master Degree Candidate. Her research interests include security in cloud computing and provable data possession.

Copyright ⓒ 2014 SERSC

Chang Liu, born in 1989, Ph.D. candidate. His research interests include cloud computing security, privacy-preserving data outsourcing and big data privacy.

Copyright ⓒ 2014 SERSC