Indian Journal of Science and Technology, *Vol 9(29), DOI: 10.17485/ijst/2016/v9i29/90783, August 2016*

ISSN (Print) : 0974-6846
ISSN (Online) : 0974-5645

Improved Data Integrity Proofs using Additive Homomorphic Encryption for Remote Storage

Parth Shah[1*] and Amit Ganatra[2]

[1]Department of Information Technology, Chandubhai S Patel Institute of Technology, Charusat, Changa - 388421, Gujarat, India; parthshah.ce@charusat.ac.in
[2]Department of Computer Engineering, Chandubhai S Patel Institute of Technology, Charusat, Changa - 388421, Gujarat, India; amitganatra.ce@charusat.ac.in

Abstract

Background/Objectives: Remote data storage becomes the hype nowadays as various organizations provide free access of the application. Security and efficiency are the major concerns while using such kind of application. Hence solution related to data integrity verification needs to be focused to achieve reliability. **Methods/Statistical Analysis:** Despite of all the hype surrounding the storage and security solutions, clients are still hesitant to deploy their business in the cloud. As security is the major concerns which may become the hindrance as clients are more concern about data privacy and data protection. In proposed work we have provided solution of efficiency and security related to remote data storage using additive homomorphic encryption. **Findings:** Solution to this can be provided using some of the data integrity proofs techniques. The advent of an advanced model of the security to provide the solution should not negotiate with the required efficiency and reliability present in the current solutions. Additive homomorphic encryption has been proven to efficient compared to multiplicative homomorphic encryption. Additive homomorphic encryption is efficient compare to multiplicative homomorphic encryption which provides better efficiency. **Applications/Improvements:** Methodology given could be useful for remote file storage applications. Proposed algorithm has been implemented and tested. The results shows that it gives improvement over the existing solution in terms of tag generation and verification.

Keywords: Additive Homomorphism, Data Integrity Proof, Encryption, Efficiency, Remote Storage

1. Introduction

Storage outsourcing causes a number of challenges[1]. To verify that the file or data has been stored on server entrusted to it by the client should be verifiable. The server may not be trustworthy in terms of security and reliability, e.g., it may maliciously or accidentally erase the data or migrate it onto thearchives. Worsening the problems are factors such as restricted network bandwidth and limited computing power. This problem is considered as data integrity issue which can be solved using various data integrity proofs mechanism.

Verifying integrity of the data, huge amount of data need to be downloaded, which incurs network overhead. The basic problem is that the client required to access whole fileto perform verification, and the client maybe constrained for verification due to a priory bound. In addition, there are also three problems like public verifiability, data updates and privacy against third party auditors.

In the proposed work we have tried to provide solution related to remote data integrity verification using additive homomorphic tags[2], which is quite efficient compared to existing solutions.

Rest of the sections of paper are organized as follows: Section 2 gives the information of related work done so far. Section 3 describes various cryptographic assessment and overview of proposed scheme. Section 4 gives the details about the proposed algorithm. Implementation and experimental results are given in Section 5. Conclusion is given in Section 6 and which also gives the future enhancement related to proposed work.

**Author for correspondence*

2. Related Work

Initial solution of data integrity proofs provided by[3] uses hash functions which is based on RSA. Unfortunately the solution has the downside of needing the server to exponentiation over the whole file. This clearly prohibits the server whenever the file size is too large and more number of client access the systems simultaneously. As proposed in[4] improvement of Speed in Public Key Cryptography using Message Encoding Algorithm is used to improve the speed of public key cryptography but this would be overhead for remote file storage.

Additionally error recovery along with integrity checking is provided in[5]. They propose scheme using erasure-coded data that comprehends availability of data in event of error. XOR-based, parity m/n erasure codes have been used to create n different shares of a file which would bekept at multiple sites. To provide collusion resistant scheme, they blind parity and data by XORing them with a pseudo-random stream.

In[6] provided first probabilistic solution of Provable Data Possession (PDP). It provides data format independence without constraining on possible number of tries the user can challenge to the remote server. This scheme uses multiplicative homomorphic verifiable tags. However, this does not guarantee that the client can retrieve the file in the case of a failure. Also it relies on modular exponentiation over file blocks, so scheme is computationally intensive.

The theory of Proof of Retrievability (POR) is introduced by[7]. It can recover the data along with it checks the possession of data in case of disruption whereas PDP does not. Sentinels are used which disguised the blocks, hidden among file blocks such that the server cannot distinguish from encrypted blocks. This scheme can only be useful to encrypted files. Number of challenges is fixed a priory, as individual challenge consumes some sentinel blocks.

All the above mentioned techniques are built on public-key cryptosystem; consequently the calculation overheads are very severe, particularly in the case of huge file size. In[8] authors have proposed PDP scheme based which is based on symmetric key cryptosystem and a provably secure, which supports some dynamic operations, including modification, deletion and appending. They store pre-computed response in the form of metadata. In this method, Computation and communication complexities of server and client are less because of symmetric key cryptosystem however it is unsuitable for public verification. As it is not fully dynamic, it cannot do block insertions, but only append type insertions are possible. Number of challenges and updates are fixed a priori so limited. Also update requires recreation of remaining challenges.

Multiple-Replica PDP (MR-PDP)[9] is a provably secure scheme. Client is allowed to stores t replicas of a file so that it can verify t copies that held by the server. They have extended previous work on PDP for a single copy of a file.

Table 1. Comparative analysis of various PDP schemes

Scheme Matrix	[3]	[6]	[9]	[12]	[13]	[14]	[15]	[16]
Data possession	NO	YES	YES	YES	YES	YES	YES	YES
Support Sampling	NO	YES	YES	YES	YES	YES	YES	YES
Type of Guarantee	D	D	P	P	P	P	P	P
Server Block access	O(logn)	O(1)	O(1)	O(logn)	O(1)	O(logn)	O(logn)	O(logn)
Server Computation overhead	O(1)	O(1)	O(1)	O(logn)	O(1)	O(logn)	O(logn)	O(logn)
Client Computation overhead	O(1)	O(1)	O(1)	O(logn)	O(1)	O(logn)	O(1)	O(logn)
Communication overhead	O(logn)	O(1)	O(1)	O(logn)	O(1)	O(logn)	O(1)	O(logn)
Storage cost	O(1)	O(1)	O(1)	O(n)	O(1)	O(1)	O(1)	O(1)
Support dynamic integrity	NO	NO	YES	YES	YES	NO	YES	YES
Supporting public auditability	NO	YES	NO	NO	YES	YES	NO	YES
Data recovery	NO	NO	NO	NO	YES	NO	NO	NO

The scheme is computationally more efficient in comparison to use single replica PDP. This scheme is able to generate further replicas on demand with low cost, when failure occurs. Unfortunately, RSA is used to provide the solution and data update is also not considered.

In their extended work[10] they use Forward Error-Correcting codes (FEC) which resulted in trade-offs on performance, flexibility and reconfigurable rate of error correction and data format of output, distilled the security requirements and key performance for mixing FECs into PDP. A Monte-Carlo simulation is used to build and to evaluate trade-offs in space overhead, reliability and performance. A detailed analysis of the scheme is provided which quantifies the probability of the success of an attacker given different attack strategies, encodings and client checking strategies. However, the scheme needs to generate MACs for each block, resulting into large additional storage.

Previous schemes related to POR use the extra storage and is applicable only to encrypted files. In[11] designed two different schemes based on POR. One of them isbased on BLS signatures that has been proved secure in the random oracle model, also it has the shortest response and query with public verifiability. Another one isbuilt on function of pseudorandom, which has also shortest response but a longer query including private verifiability. They encode file using an erasure code, which in turn able to convert into an error-correcting code. The scheme provides the static solution without any data update. Comparative analysis of existing solution is given in Table 1.

3. Preliminaries

3.1 Cryptographic Setting and Assumptions

Our scheme uses two public parameters. The positive integer $d > 2$ and m which is a large integer which should have many small divisors and at the same time it should have many integers less than m that can be inverted modulo m.

The secret parameters is$r \in Z_m$ such that $r^{-1} \bmod m$ exists and a small divisor $m' > 1$ of m such that $s = \log_{m'} m$ is a (secret) security parameter. This parameters are based on[17].

3.2 Overview of Proposed Scheme

Proposed scheme involves the three entities. One of them is server S. Clients C that outsources their data and prompt the storage server to generate proof. Third party, denoted by Auditor denoted by AUD that allowed to check the integrity of clients' data outsourced to the server.

Our scheme consists of the following modules:

- *KEYGEN*$(1^S) \rightarrow (p_k, s_k)$: This module will generate key. Parameter s as taken as security input, and generate sp$_k$ and s$_k$ as public parameters, where p$_k$ and s$_k$ is the corresponding public key and private key of a client respectively.
- *TAG GENERATION* $(p_k, s_k, F_{id}) \rightarrow T_m$: Client initiates this module to generate the verification metadata. File is divided into the blocks like a_1, $a_2 \ldots a_n$, where n is number of blocks. It takes file block a_i, a secret key s_k, a public key p_k as an input and returns the verification metadata $E_k(a_i)$.
- *UPLOAD*(F, T_m): Over a private channel this *data uploading* module executed by the client to assure secrecy of the data.
- *AUDITINT* (GenProof$(p_k, F_{id}, chal) \rightarrow \gamma$ and Check Proof$(p_k, s_k, chal, \gamma) \rightarrow$ {"success", "failure"}): This module is executed between server S and verifier (client or third party) so that server convinces verifier that file is not maliciously tempered. The verifier provides the file identifier F_{id} as an input and the corresponding private key p_k. The server takes F as an input corresponding to F_{id}. The solution is based on challenge-response type protocol, where verifier sends a *chal* to the prover and the prover compute *sresp* and sends to verifier. If *resp* is valid with respect to *chal*, verifier outputs success, that indicates integrity of F is guaranteed, otherwise failure.

4. Proposed Algorithm

In the proposed scheme is based on the approach used in[17]. Followings are the detailed description of the each modules of the system

- **KEY GENERATION** (1^S)

Generate public key $p_k = (d, m)$ and secret key $s_k = (r, m')$, such that $s = \log_{m'} m$ is a security parameter and m is large integer which may have many divisor.

- **TAG GENERATION** (p_k, s_k, F_{id})

Let $p_k = (d, m)$ and $s_k = (r, m')$, $F = a_1, a_2 \ldots a_n$. where n is number of blocks.

Randomly split a_i into secrete $a_{i1}, a_{i2} \ldots a_{id}$ such that

$$a_i = \sum_{j=1}^{d} a_{ij} \bmod m'.$$

Generate tags $E_k(a_i) = (a_{i1} * r \bmod m, a_{i2} * r \bmod m, \ldots, a_{id} * r \bmod m)$, where $1 \leq i \leq$n.

- **Calculate**

$$Tag_{Sum} = \sum_{i=1}^{n} E_k\left(a_{i1}\right), \sum_{i=1}^{n} E_k\left(a_{i2}\right), \ldots, \sum_{i=1}^{n} E_k\left(a_{i1}\right))$$

and generate

$$\tau_{sum} = \left(\sum_{i=1}^{n} E_k(a_{i1}\right) + \sum_{i=1}^{n} E_k(a_{i2}) + \ldots + \sum_{i=1}^{n} E_k(a_{id})).$$

Output $\tau = (E_k\left(a_i\right), \tau_{sum})$.

- **GENERATE PROOF** (p_k, F_{id}, *chal*)

Let p_k= (d, m) and *chal*=(c, r', m'', k)where $m' < m''$ $and\ r' \epsilon Z_m$ such that $r'^{-1} modm$ exits.

Compute section the indices of the for which the proofs will be generated $i_j = \pi_k\left(c\right)$, for $1 \leq j \leq c$.

Compute

$$E_k'\left(a_{i_c}\right) = (a_{i_1 1} * r' modm,\ a_{i_2 2} * r' modm, \ldots, a_{i_c d} * r' modm)$$

where $a_{i_c} = \sum_{j=1}^{d} a_{i_c j} modm''$.

Calculate $Tag'_{Sum} =$

$$(\sum_{i=1}^{n} E_k\left(a_{i_1 1}\right), \sum_{i=1}^{n} E_k\left(a_{i_2 2}\right), \ldots, \sum_{i=1}^{n} E_k\left(a_{i_c k}\right)).$$

Generate

$$\tau'_{sum} = \left(\sum_{i=1}^{n} E_k(a_{i_1 1}\right) + \sum_{i=1}^{n} E_k(a_{i_2 2}) + \ldots + \sum_{i=1}^{n} E_k(a_{i_c k}))$$

where $k \epsilon [1 \ldots c]$

Output $\gamma = (E_k'\left(a_{i_c}\right), \tau'_{sum})$.

CHECK PROOF (*pk*, *F*, *chal*, γ)

Letp_k= (d, m), s_k = (r, m') and *chal*=(c, r', m'', k) and γ for $1 \leq k \leq c$.

Compute

$$E_k''\left(a_i\right) = ((E_k\left(a_{i1}\right) * r'^{-1}) * rmodm,\ (E_k\left(a_{i2}\right) * r'^{-1}) * rmodm, \ldots, (E_k\left(a_{id}\right) * r'^{-1}) * rmodm)$$

where $k \epsilon [1 \ldots c]$ where $a_i = \sum_{j=1}^{d} a_{ij} modm'$.

Calculate

$$Tag''_{Sum} = Tag'_{Sum} * r'^{-1} * r = \left(\sum_{i=e1}^{n} E_k\left(a_{i_1 1}\right) * r'^{-1}\right) * r, \left(\sum_{i=1}^{n} E_k\left(a_{i_2 2}\right) * r'^{-1}\right) * r$$

$, \ldots ,$

$$\sum_{i=1}^{n} E_k\left(a_{i_c k}\right) * r'^{-1} * r).$$

Generate

$$\tau''_{sum} = \left(\sum_{i=1}^{n} E_k(a_1\right) + \sum_{i=1}^{n} E_k(a_2) + \ldots + \sum_{i=1}^{n} E_k(a_k))$$

where $k \epsilon [1 \ldots c]$.

If $\tau'_{sum} = \tau''_{sum}$ then output success otherwise fail.

5. Implementation and Experimental Results

Experiments related to computation complexity have been conducted and compared with the conventional methodology.

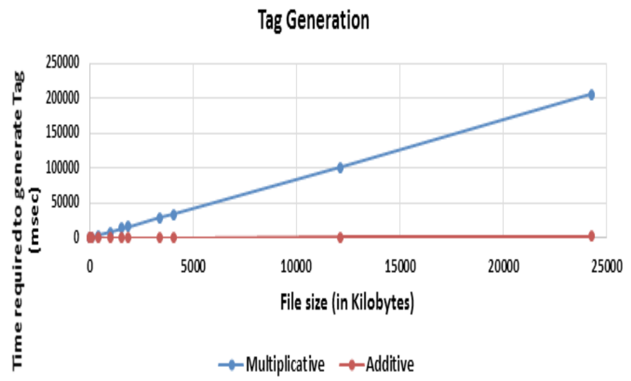

Figure 1. Time required to generate tags for files having different size.

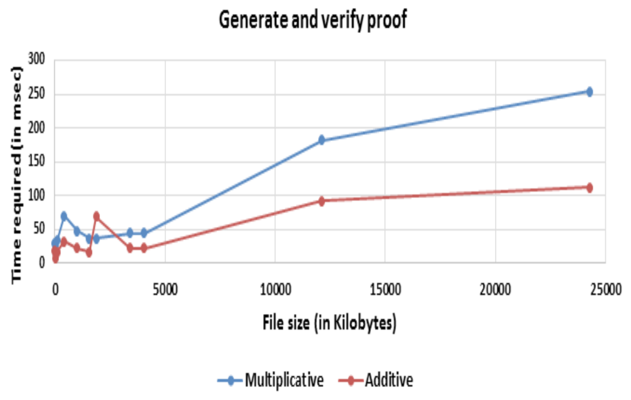

Figure 2. Time required to generate and verification of proof.

All experiments are conducted on i5-4210 U an Intel core CPU @1.7GHz 2.4GHz system with a 256 KB cache, and 8 GB of RAM. Windows 64 bit operating system is used to run the experiments. Experiments were conducted on different file size to simulate time required to generate tags (Figure 1.) and for the generation and verification of proof (Figure 2).

Table 2. Pre-processing vs. challenge trade-offs with file size

File Size (in KB)	Tag Generation		Generate and Verify Proof	
	Multi (in ms)	Add (in ms)	Multi (in ms)	Add (in ms)
1	1.67	2.19	28.93	18.03
5	55.91	11.21	29.72	15.96
10	114.78	19.05	16.44	8.26
25	246.16	27.53	27.68	15.79
100	911.45	55.45	33.12	16.59
418	3960.94	81.44	69.57	32.82
1010	9030.69	164.53	47.69	22.55
1555	14366.93	231.07	37.2	16.41
1865	16877.83	237.4	37.21	68.62
3405	29086.95	354.42	44.43	22
4052	34461.68	435.18	44.51	22.82
12124	101481.71	1062.17	182.44	91.97
24266	206038.51	3014.42	254.3	112.25

6. Conclusion and Future Work

We provided a framework for building public-key additive homomorphic solution which stratifies the certain homomorphic properties. The proposed framework is efficient compared to previous solution.

The work can be further extended to provide solution by considering the deduplication functionality along with confidentiality. Also key distribution problem may be considered for the further improvement.

7. References

1. Lee J-Y. A study on data integrity and consistency guarantee in cloud storage for collaboration. Indian Journal of Science and Technology. 2015 Apr; 8(S7). DOI: 10.17485/ijst/2015/v8iS7/70471.
2. Suveetha K, Manju T. Ensuring confidentiality of cloud data using homomorphic encryption. Indian Journal of Science and Technology. 2016 Feb; 9(8). DOI: 10.17485/ijst/2016/v9i8/87964.
3. Deswarte Y, Quisquater J-J, Saidane A. Remote integrity checking. Proc of Conference on Integrity and Internal Control in Information Systems '03; 2003 Nov.
4. Amalarethinam DIG, Geetha JS, Mani K. Analysis and enhancement of speed in public key cryptography using message encoding algorithm. Indian Journal of Science and Technology. 2015 Jul; 8(16). DOI: 10.17485/ijst/2015/v8i16/69809.
5. Schwarz TSJ, Miller EL. Store, forget, and check: Using algebraic signatures to check remotely administered storage. 2006 26th IEEE International Conference on Distributed Computing Systems, ICDCS; 2006. p. 12. DOI: 10.1109/ICDCS.2006.80.
6. Ateniese G, Burns R, Curtmola R, Herring J, Kissner L, Peterson Z, Song D. Provable data possession at untrusted stores. Proceedings of the 14th ACM Conference on Computer and Communications Security (CCS '07); New York, NY, USA. 2007. p. 598–609. DOI: http://dx.doi.org/10.1145/1315245.1315318.
7. Juels A, Kaliski BS. Pors: proofs of retrievability for large files. Proceedings of the 14th ACM Conference on Computer and Communications security (CCS '07); New York, NY, USA. 2007. p. 584–97. DOI: http://dx.doi.org/10.1145/1315245.1315317.
8. Ateniese G, Di Pietro R, Mancini LV, Tsudik G. Scalable and efficient provable data possession. Proceedings of the 4th International Conference on Security and Privacy in Communication Networks (SecureComm '08); New York, NY, USA. 2008. p. 10. DOI: http://dx.doi.org/10.1145/1460877.1460889.
9. Curtmola R, Khan O, Burns R, Ateniese G. MR-PDP: Multiple-replica provable data possession, The 28th International Conference on Distributed Computing Systems, ICDCS '08; Beijing. 2008. p. 411–20. DOI: 10.1109/ICDCS.2008.68.
10. Chen B, Curtmola R. Robust dynamic provable data possession. Proceedings of the 2012 32nd International Conference on Distributed Computing Systems Workshops (ICDCSW '12); Washington, DC, USA: IEEE Computer Society. 2012. p. 515–25. DOI: http://dx.doi.org/10.1109/ICDCSW.2012.57.
11. Shacham H, Waters B. Compact proofs of retrievability. In: Pieprzyk J, editor. Proceedings of the 14th International Conference on the Theory and Application of Cryptology and Information Security: Advances in Cryptology (ASIACRYPT '08); Berlin, Heidelberg: Springer-Verlag. 2008. p. 90–107. DOI: http://dx.doi.org/10.1007/978-3-540-89255-7_7.
12. Erway C, Kupcu A, Papamanthou C, Tamassia R. Dynamic provable data possession. In Proceedings of the 16th ACM Conference on Computer and Communications Security (CCS '09); New York, NY, USA. 2009. p. 213–22. DOI: http://dx.doi.org/10.1145/1653662.1653688.

13. Storer MW, Greenan K, Long DDE, Miller EL. Secure data deduplication. In Proceedings of the 4th ACM International Workshop on Storage Security and Survivability (StorageSS '08); New York, NY, USA. 2008. p. 1–10. DOI: http://dx.doi.org/10.1145/1456469.1456471.
14. Wang C, Wang Q, Ren K Lou W. Ensuring data storage security in cloud computing. 17th International Workshop on Quality of Service, IWQoS; Charleston, SC. 2009. p. 1–9. DOI: 10.1109/IWQoS.2009.5201385.
15. Liu F, Gu D Lu H. An improved dynamic provable data possession model. 2011 IEEE International Conference on Cloud Computing and Intelligence Systems (CCIS); Beijing. 2011. p. 290–5. DOI: 10.1109/CCIS.2011.6045077.
16. Shen S-T, Tzeng W-G. Delegable provable data possession for remote data in the clouds. Qing S, Susilo W, Wang G, Liu D, editors. Proceedings of the 13th International Conference on Information and Communications Security (ICICS'11); Berlin, Heidelberg: Springer-Verlag. 2011. p. 93–111.
17. Domingo-Ferrer. A provably secure additive and multiplicative privacy homomorphism. Information Security-ISC, Springer LNCS. 2002; 2433:471–83.