ISSN 2320 - 2602
Volume 9 No.7, July 2020
International Journal of Advances in Computer Science and Technology
Available Online at http://www.warse.org/IJACST/static/pdf/file/ijacst15972020.pdf
https://doi.org/10.30534/ijacst/2020/15972020

De duplication Implementation using AES Algorithm

[1] **Punam Shriram Wankhede , [2] Dr. A. S. Kapse**
[1] Student of Final Year M.E, Computer Science & Engineering, Sant Gadge Baba Amravati University, Chikhli, Maharashtra, India
Punamwankhede18@gmail.com
[2] Assistant Professor (Sr. Scale) & Head, Department of Computer Science & Engineering, Anuradha Engg. College, Sant Gadge Baba Amravati University, Chikhli, Maharashtra, India
askapse@gmail.com

ABSTRACT
Vulnerability of data is of utmost important now a days and to secure it became a challenging part. **T**here are number of process which are available and many strategic plans are implemented and currently introducing to make the system more impact. Our proposed system works on the same factor to authenticate the data during communication and securing the data not only at sender side but also at receiver side. The methodology we are using includes utilization of ring signatures to construct homomorphism authenticators, so that a public verifier is able to audit shared data integrity without retrieving the entire data, yet it cannot distinguish who is the signer on each block. It doesn't incurred more traffic by performing multiple auditing tasks simultaneously. It also keeps the third party secured but can be retrieved with authentication credentials. The user is logged in and can be able to access and modify the data as necessary before de duplication.

Key words: Homomorphism, authenticators, auditing, data de duplication

1. INTRODUCTION

Cloud computing greatly facilitates data providers who want to outsource their data to the cloud without disclosing their sensitive data to external parties and would like users with certain credentials to be able to access the data. This requires data to be stored in encrypted forms with access control policies such that no one except users with attributes (or credentials) of specific forms can decrypt the encrypted data. An encryption technique that meets this requirement is called attribute-based encryption (ABE), where a user's private key is associated with an attribute set, a message is encrypted under an access policy (or access structure) over a set of attributes, and a user can decrypt a cipher text with his/her private key if his/her set of attributes satisfies the access policy associated with this cipher text. However, the standard ABE system fails to achieve secure deduplication, which is a technique to save storage space and network bandwidth by eliminating redundant copies of the encrypted data stored in the cloud. On the other hand, to the best of our knowledge, existing constructions for secure deduplication are not built on attribute-based encryption. Nevertheless, since ABE and secure deduplication have been widely applied in cloud computing, it would be desirable to design a cloud storage system possessing both properties [1][2]

Table 1: Comparison of various algorithm with AES

Sr.No.		**Parameters**						
	Algorithms	**Time**	**Bit keys**	**Multiple Network Functionality**	**Sym**	**More round addition**	**Speed**	**Accuracy**
1	Data Encryption Standard (DES)	Maximum Time	It uses a 56-bit key to encrypt the 64 bit block size data.	No	No	No	Normal	60%
2	International Data Encryption Algorithm (IDEA)	Maximum Time	It uses 128 bit key length which operates on 64 bit blocks	No	No	No	Normal	70%
3	Blowfish	Minimum Time as Compared to IDEA	Provides a good encryption rate in software and no effective cryptanalysis of it has been found to date	No	Yes	No	Good	60%
4	Triple DES (TDES)	Minimum Time as Compared to Blowfish	Three keys are referred to as bundle keys with 56 bits per key.	No	No	No	Good	75%
5	Advanced Encryption Standard (AES)	Minimum Time	AES has a fixed block size of 128-bits and a key size of 128, 192, or 256-bits	Yes	Yes	yes	Good	80%

2. PROPOSED SYSTEM

The propose system is a privacy-preserving public auditing mechanism for shared data in the cloud. We utilize ring signatures to construct homomorphism authenticators, so that a public verifier is able to audit shared data integrity without retrieving the entire data, yet it cannot distinguish who is the signer on each block.

To improve the efficiency of verifying multiple auditing tasks, we further extend our mechanism to support batch auditing.

There are two interesting problems we will continue to study for our future work. One of them is traceability, which means the ability for the group manager to reveal the identity of the signer based on verification metadata in some special situations. [3]

Module:

- User Registration:

For the registration of user with identity ID the group manager randomly selects a number. Then the group manager adds into the group user list which will be used in the traceability phase. After the registration, user

Obtains a private key which will be used for group signature generation and file decryption.

Registration

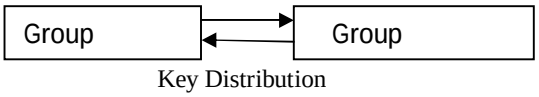


Figure 1: User Registration

- Public Auditing

Homomorphic authenticators are unforgeable verification metadata generated from individual data blocks, which can be securely aggregated in such a way to assure an auditor that a linear combination of data blocks is correctly computed by verifying only the aggregated authenticator.[4]

Overview to achieve privacy-preserving public auditing, we propose to uniquely integrate the Homomorphic authenticator with random mask technique.

In our protocol, the linear combination of sampled blocks in the server's response is masked with randomness generated by a pseudo random function (PRF). The proposed scheme is as follows:

- Setup Phase
- Audit Phase

- Sharing Data

The canonical application is data sharing. The public auditing property is especially useful when we expect the delegation to be efficient and flexible. The schemes enable a content provider to share her data in a confidential and selective way, with a fixed and small cipher text expansion, by distributing to each authorized user a single and small aggregate key.

- Integrity Checking

Hence, supporting data dynamics for privacy-preserving public risk auditing is also of paramount importance. Now we show how our main scheme can be adapted to build upon the existing work to support data dynamics, including block level operations of modification, deletion and insertion and can adopt this technique in our design to achieve privacy-preserving public risk auditing with support of data dynamics. The user download the particular file not download entire file by using the proposed algorithm and its design in efficient and clever manner. [4][5]

3. ALGORITHM EXPLANATION

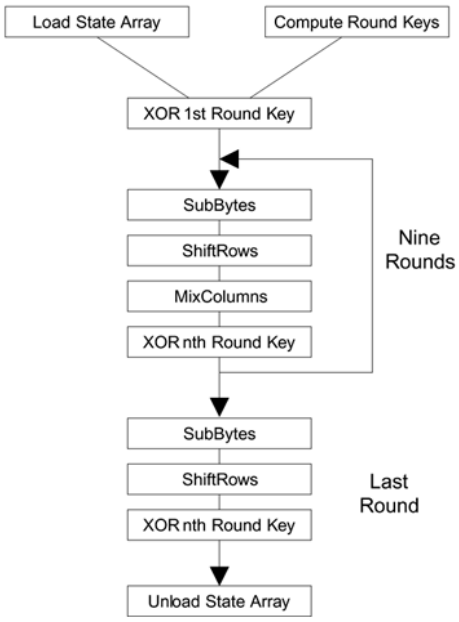


Figure 2 : Flow Chart of AES

Steps

1. The data is divided into blocks. Under this method of encryption, the first thing that happens is that your plaintext (which is the information that you want to be encrypted) is separated into blocks. The block size of AES is 128-bits, so it separates the data into a four-by-four column of sixteen bytes (there are eight bits in a byte and 16 x 8 = 128). If your message was "buy me some potato chips please" the first block looks like this:

b	m	o	p
u	e	m	o
y		e	t
	s		a

By skipping the rest of the message for this example and just focus on what happens to the first block as it is encrypted. The "…to chips please" would normally just be added to the next block.

2. Key expansion involves taking the initial key and using it to come up with a series of other keys for each round of the encryption process. These new 128-bit round keys are derived with Rijndael's key schedule, which is essentially a simple and fast way to produce new key ciphers. If the initial key was "keys are boring1":

k			i
e	a	b	n
y	r	o	g
s	e	r	1

3. Then each of the new keys might look something like this once Rijndael's key schedule has been used:

14	29	1h	s5
h9	9f	st	9f
gt	2h	hq	73
ks	dj	df	hb

Although they look like random characters (and the above example is just made up) each of these keys is derived from a structured process when AES encryption is actually applied. We'll come back to what these round keys are used for later on.

4. In this step, because it is the first round, our initial key is added to the block of our message:

b	m	o	p
u	e	m	o
y		e	t
	s		a

+

k			i
e	a	b	n
y	r	o	g
s	e	r	1

5. This is done with an XOR cipher, which is an additive encryption algorithm. While it looks like you can't actually add these things together, be aware that it is actually done in binary. The characters are just a stand-in to try and make things easier to understand. Let's say that this mathematical operation gives us a result of:

h3	jd	zu	7s
s8	7d	26	2n
dj	4b	9d	9c
74	el	2h	hg

6. In this step, each byte is substituted according to a predetermined table. This is kind of like the example from the start of the article, where the sentence was coded by changing each letter to the one that comes after it in the alphabet (hello becomes ifmmp). This system is a little bit more complicated and doesn't necessarily have any logic to it. Instead, there is an established table that can be looked up by the algorithm, which says, for example, that h3 becomes jb, s8 becomes 9f, dj becomes 62 and so on. After this step, let's say that the predetermined table gives us:

jb	n3	kf	n2
9f	jj	1h	js
74	wh	0d	18
hs	17	d6	px

Many more rounds…

After the last round key was added, it goes back to the byte substitution stage, where each value is changed according to a predetermined table. Once that's done, it's back to shift rows and moving each row to the left by one, two or three spaces. Then it goes through the mix columns equation again. After that, another round key is added. It doesn't stop there either. At the start, it was mentioned that AES has key sizes of either 128, 192 or 256-bits. When a 128-bit key is used, there are nine of these rounds. When a 192-bit key is used, there are 11. When a 256-bit key is used, there are 13. So the data goes through the byte substitution, shift rows, mix columns and round key steps up to thirteen times each, being altered at every stage. After these nine, 11 or 13 rounds, there is one additional round in which the data is only processed by the byte substitution, shift rows and add round key steps, but not the mix columns step. The mix columns step is taken out because at this stage, it would just be eating up processing power without altering the data, which would make the encryption method less efficient.[6] To make things clearer, the entire

AES encryption process goes:

- Key expansion
- Add round key
- Byte substitution
- Shift rows
- Mix columns
- Add round key
- x 9, 11 or 13 times, depending on whether the key is 128, 192 or 256-bit
- Byte substitution
- Shift rows
- Add round key

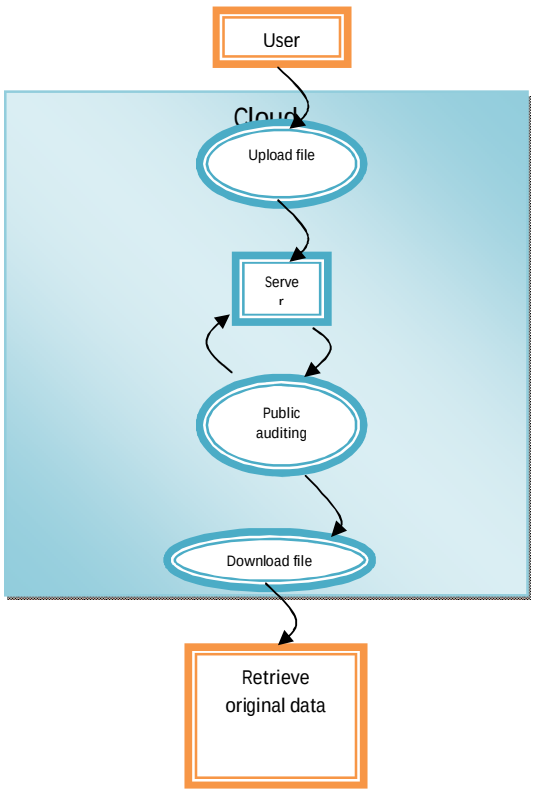
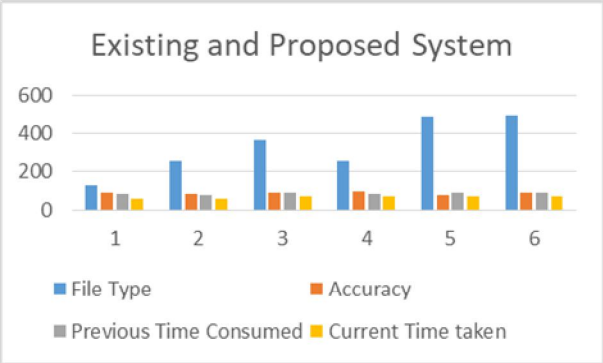


Figure 3: DFD Diagram of Public Auditing & privacy Preserving System

Graphical result comparison in accordance with accuracy and time consumed of previous and proposed system with Data Storage and Accuracy

File System Type (MB)	Accuracy (%)	Previous Time Consumed	Current Time
Word (125)	90	85	79
Excel (255)	85	80	71
Word (369)	90	92	83
Excel (256)	93	83	70
Database (485)	75	89	78
Image (495)	89	90	77



4. CONCLUSION

After finding the result on the basis of time consuming and accuracy it was concluded that by making use of current method the accuracy of almost 80% of receiving data is achieved as compared to previous and the rest of the files remain secured. Also while sharing the data, the user can generate a secret key for the sharing file so that only the file that need to share can be access. The concept of key-aggregate cryptosystem is introduced recently. In this method many secret keys could be combined to form a single secret key. The current encryption algorithm works fine with the text file, in future we intend to develop and implement an algorithm that could encrypt and decrypt other data formats like images, tables, graphs etc.[7][8]

REFERENCES

[1] Hui Cui, Robert H. Deng, Yingjiu Li, and Guowei Wu, "*Attribute-Based Storage Supporting Secure Deduplication of Encrypted Data in Cloud.*" IEEE Transactions on Cloud computing,year: 2017, Volume: PP, Issue: 99

[2] Bellare.M, Keelveedhi.S, Ristenpart.T (2013), "*Message-locked encryption and secure deduplication*", In EUROCRYPT, pp 296– 312 . Ng, C., & Lee, P. P. C. (2013). RevDedup : A ReverseDeduplication Storage System Optimized for Reads to Latest Backups.

[3] Luo, X., Yang, L., Hao, D., Liu, F., & Wang, D. (2014). "*On Data and Virtualization Security Risks and Solutions of Cloud Computing.*" Journal of Networks, 9(3), 571–581. doi:10.4304/jnw.9.3.571- 581

[4] Amoroso, E. G. (2014). "*Practical methods for securing the cloud. IEEE Cloud Computing,*"1(1),28–38. doi:10.1109/MCC.2014.1

[5] Ankita N., & Sheeja S. (2014a). "*Design and Implementation of AES Algorithm Using, 2(1)*", 419– 423.

[6] Brar, Y., Krishan, S., Mehta, A., & Talwar, V. (2014). "*An Advanced Security - A Two-Way Password Technique for Cloud Services*", 3(4), 7–15.

[7] Li, J., Li, Y. K., Chen, X., Lee, P., & Lou, W. (2014). "*A Hybrid Cloud Approach for Secure Authorized Deduplication. IEEE Transactions on Parallel and DistributedSystems*",1–1.doi:10.1109/ TPDS.2014. 2318320.

[8] G. Ateniese, K. Fu, M. Green, and S. Hohenberger, "*Improved Proxy Re-Encryption Schemes with Applications to Secure Distributed Storage,*" ACM Transactions on Information and System Security, 9(1), 2006, pp. 1-30

[9] D.T. Meyer and W.J Bolosky., "*A Study of Practical Deduplication,*"ACM Transactions on Storage, 2012, pp.1-20.

[10] J.R. Douceur, A. Adya, W.J. Bolosky, D. Simon, and M.Theimer, "*Reclaiming Space from Duplicate Files in a Serverless Distributed File System,*" Proceedings of IEEE International Conference on Distributed Computing Systems, 2002, pp. 617 -624