Journal for Research | Volume 02 | Issue 09 | November 2016
ISSN: 2395-7549

# Homomorphic Authenticable Ring Signature Mechanism for Public Auditing on Shared Data in the Cloud

**MS. Gayatri D Patwardhan**
*Department of Computer Engineering*
*BVCOE, Navi Mumbai*

**Prof.B. W. Balkhande**
*Department of Computer Engineering*
*BVCOE, Navi Mumbai*

## Abstract

With cloud storage services, it is commonplace for data to be not only stored in the cloud, but also shared across multiple users. However, public auditing for such shared data — while preserving identity privacy remains to be an open challenge. In this paper, we propose the first privacy-preserving mechanism that allows public auditing on shared data stored in the cloud. In particular, we exploit ring signatures to compute the verification information needed to audit the integrity of shared data. With our mechanism, the identity of the signer on each block in shared data is kept private from a third-party auditor (TPA), who is still able to publicly verify the integrity of shared data without retrieving the entire file. Our experimental results demonstrate the effectiveness and efficiency of our proposed mechanism when auditing shared data.

**Keywords: Homomorphic Authenticable Ring Signature Mechanism, TPA**

---

## I. INTRODUCTION

Cloud service providers manage an enterprise-class infrastructure that offers a scalable, secure and reliable environment for users, at a much lower marginal cost due to the sharing nature of resources. It is routine for users to use cloud storage services to share data with others in a team, as data sharing becomes a standard feature in most cloud storage offerings, including Drop box and Google Docs.

The integrity of data in cloud storage, however, is subject to skepticism and scrutiny, as data stored in an untrusted cloud can easily be lost or corrupted, due to hardware failures and human errors. To protect the integrity of cloud data, it is best to perform public auditing by introducing a third-party auditor (TPA), who offers its auditing service with more powerful computation and communication abilities than regular users.

The first provable data possession (PDP) mechanism to perform public auditing is designed to check the correctness of data stored in an untrusted server, without retrieving the entire data. Moving a step forward, Wang et al. (referred to as WWRL) is designed to construct a public auditing mechanism for cloud data, so that during public auditing, the content of private data belonging to a personal user is not disclosed to the third-party auditor.

We believe that sharing data among multiple users is perhaps one of the most engaging features that motivates cloud storage. A unique problem introduced during the process of public auditing for shared data in the cloud is how to preserve identity privacy from the TPA, because the identities of signers on shared data may indicate that a particular user in the group or a special block in shared data is a higher valuable target than others. For example, Alice and Bob work together as a group and share a file in the cloud. The shared file is divided into a number of small blocks, which are independently signed by users. Once a block in this shared file is modified by a user, this user needs to sign the new block using her public/private key pair. The TPA needs to know the identity of the signer on each block in this shared file, so that it is able to audit the integrity of the whole file based on requests from Alice or Bob.

We propose Oruta, a new privacy preserving public auditing mechanism for shared data in an untrusted cloud. In Oruta, we utilize ring signatures to construct homomorphic authenticators, so that the third party auditor is able to verify the integrity of shared data for a group of users without retrieving the entire data — while the identity of the signer on each block in shared data is kept private from the TPA. In addition, we further extend our mechanism to support batch auditing, which can audit multiple shared data simultaneously in a single auditing task. Meanwhile, Oruta continues to use random masking to support data privacy during public auditing, and leverage index hash tables to support fully dynamic operations on shared data. A dynamic operation indicates an insert, delete or update operation on a single block in shared data. A high-level comparison between Oruta and existing mechanisms in the literature is shown. To our best knowledge, this represents the first attempt towards designing an effective privacy preserving public auditing mechanism for shared data in the cloud.

### *A. Existing System*

Many mechanisms have been proposed to allow not only a data owner itself but also a public verifier to efficiently perform integrity checking without downloading the entire data from the cloud, which is referred to as public auditing . In these mechanisms, data is divided into many small blocks, where each block is independently signed by the owner; and a random combination of all the blocks instead of the whole data is retrieved during integrity checking. A public verifier could be a data user (e.g., researcher) who would like to utilize the owner's data via the cloud or a third-party auditor (TPA) who can provide expert integrity checking services.

Moving a step forward, Wang et al. designed an advanced auditing mechanism .so that during public auditing on cloud data, the content of private data belonging to a personal user is not disclosed to any public verifiers. Unfortunately, current public auditing solutions mentioned above only focus on personal data in the cloud .We believe that sharing data among multiple users is perhaps one of the most engaging features that motivates cloud storage. Therefore, it is also necessary to ensure the integrity of shared data in the cloud is correct.

Existing public auditing mechanisms can actually be extended to verify shared data integrity. However, a new significant privacy issue introduced in the case of shared data with the use of existing mechanisms is the leakage of identity privacy to public verifiers. However, a new significant privacy issue introduced in the case of shared data with the use of existing mechanisms is the leakage of identity privacy to public verifiers.

#### 1) Disadvantages of Existing System

- Failing to preserve identity privacy on shared data during public auditing will reveal significant confidential information to public verifiers.
- Protect these confidential information is essential and critical to preserve identity privacy from public verifiers during public auditing.

### B. Proposed system

To solve the above privacy issue on shared data, we propose Oruta, a novel privacy-preserving public auditing mechanism. More specifically, we utilize ring signatures to construct homomorphic authenticators in Oruta, so that a public verifier is able to verify the integrity of shared data without retrieving the entire data while the identity of the signer on each block in shared data is kept private from the public verifier.

In addition, we further extend our mechanism to support batch auditing, which can perform multiple auditing tasks simultaneously and improve the efficiency of verification for multiple auditing tasks. Meanwhile, Oruta is compatible with random masking, which has been utilized in WWRL and can preserve data privacy from public verifiers. Moreover, we also leverage index hash tables from a previous public auditing solution to support dynamic data. A high-level comparison among Oruta and existing mechanisms is presented.

#### 1) Advantages of Proposed System

- A public verifier is able to correctly verify shared data integrity.
- A public verifier cannot distinguish the identity of the signer on each block in shared data during the process of auditing.
- The ring signatures generated for not only able to preserve identity privacy but also able to support block less verifiability.

## II. LITERATURE SURVEY

Literature survey is the most important step in software development process. Use of privacy-preserving public auditing mechanism, and cryptographic scheme can increase the security level for the data that are stored on the cloud servers.

Following is the literature survey of some existing technique for cloud security.

### A. Q Wang, C. Wang, "Enabling Public Verifiability and Data Dynamics for Storage Security in Cloud Computing"

Cloud Computing has been envisioned as the next generation architecture of IT Enterprise. It moves the application software and databases to the centralized large data centers, where the management of the data and services may not be fully trustworthy. This unique paradigm brings about many new security challenges, which have not been well understood. This work studies the problem of ensuring the integrity of data storage in Cloud Computing. In particular, we consider the task of allowing a third party auditor (TPA), on behalf of the cloud client, to verify the integrity of the dynamic data stored in the cloud. The introduction of TPA eliminates the involvement of the client through the auditing of whether his data stored in the cloud is indeed intact, which can be important in achieving economies of scale for Cloud Computing. The support for data dynamics via the most general forms of data operation, such as block modification, insertion and deletion, is also a significant step toward practicality, since services in Cloud Computing are not limited to archive or backup data only. While prior works on ensuring remote data integrity often lacks the support of either public verifiability or dynamic data operations, this paper achieves both.

We first identify the difficulties and potential security problems of direct extensions with fully dynamic data updates from prior works and then show how to construct an elegant verification scheme for the seamless integration of these two salient features in our protocol design. In particular, to achieve efficient data dynamics, we can improve the existing proof of storage models by manipulating the classic (MHT) Merkle Hash Tree construction for block tag authentication. To support efficient handling of multiple auditing tasks, we can further explore the technique of bilinear aggregate signature to extend our main result into a multi-user setting, where TPA can perform multiple auditing tasks simultaneously. Extensive security and performance analysis show that the proposed schemes are highly efficient and provably secure.

### *B. C. Wang, Q. Wang, "Privacy-Preserving Public Auditing for Data Storage Security in Cloud Storage"*

Using Cloud Storage, users can remotely store their data and enjoy the on-demand high quality applications and services from a shared pool of configurable computing resources, without the burden of local data storage and maintenance. However, the fact that users no longer have physical possession of the outsourced data makes the data integrity protection in Cloud Computing a formidable task, especially for users with constrained computing resources. In this paper, author propose a secure cloud storage system supporting privacy-preserving public auditing .We can further extend our result to enable the TPA to perform audits for multiple users simultaneously and efficiently.

### *C. A. Juels and B. S. Kaliski, "PORs: Proofs of Retrievability for Large Files"*

We define and explore proofs of retrievability (POR). A POR scheme enables an archive or backup service (prover) to produce a concise proof that a user (verifier) can retrieve a target file F, that is, that the archive retains and reliably transmits file data sufficient for the user to recover F in its entirety. A POR may be viewed as a kind of cryptographic proof of knowledge (POK), but one specially designed to handle a large file (or bit string) F. We explore POR protocols here in which the communication costs, number of memory accesses for the prover, and storage requirements of the user (verifier) are small parameters essentially independent of the length of F. In addition to proposing new, practical POR constructions, we explore implementation considerations and optimizations that bear on previously explored, related schemes. In a POR, unlike a POK, neither the prover nor the verifier need actually have knowledge of F. PORs give rise to a new and unusual security definition whose formulation is another contribution of our work.

We view PORs as an important tool for semi-trusted online archives. Existing cryptographic techniques help users ensure the privacy and integrity of files they retrieve. It is also natural, however, for users to want to verify that archives do not delete or modify files prior to retrieval. The goal of a POR is to accomplish these checks without users having to download the files themselves. A POR can also provide quality-of service guarantees, i.e., show that a file is retrievable within a certain time bound.

### *D. G. Ateniese, R. D. Pietro, "Scalable and Efficient Provable Data Possession"*

In this paper author introduce a model for provable datapossession (PDP) that allows a client that has stored data at an untrusted server to verify that the server possesses the original data without retrieving it. The model generates probabilistic proofs of possession by sampling random sets of blocks from the server, which drastically reduces I/O costs. The client maintains a constant amount of metadata to verify the proof. The challenge/response protocol transmits a small, constant amount of data, which minimizes network communication. Thus, the PDP model for remote data checking supports large data sets in widely distributed storage systems. To support the dynamic auditing, Ateniese et al. developed a dynamic provable data possession protocol based on cryptographic hash function and symmetric key encryption. Their idea is to pre-compute a certain number of metadata during the setup period, so that the number of updates and challenges is limited and fixed beforehand.

The author construct a highly efficient and provably secure PDP technique based entirely on symmetric key cryptography, while not requiring any bulk encryption. Also, in contrast with its predecessors, this PDP technique allows outsourcing of dynamic data, i.e, it efficiently supports operations, such as block modification, deletion and append.

### *E. C. Wang, Q. Wang, "Towards Secure and Dependable Storage Services in Cloud Computing"*

In this paper, author has proposed an effective and flexible distributed storage verification scheme with explicit dynamic data support to ensure the correctness and availability of users' data in the cloud. They rely on erasure correcting code in the file distribution preparation to provide redundancies and guarantee the data dependability against Byzantine servers, where a storage server may fail in arbitrary ways. This construction drastically reduces the communication and storage overhead as compared to the traditional replication based file distribution techniques. By utilizing the homomorphic token with distributed verification of erasure coded data, their scheme achieves the storage correctness insurance as well as data error localization: whenever data corruption has been detected during the storage correctness verification, this scheme can almost guarantee the simultaneous localization of data errors, i.e., the identification of the misbehaving server(s).

In order to strike a good balance between error resilience and data dynamics, their work further explore the algebraic property of our token computation and erasure coded data, and demonstrate how to efficiently support dynamic operation on data blocks, while maintaining the same level of storage correctness assurance. In order to save the time, re- sources, and even the related online burden of users, extension of the proposed main scheme to support third party auditing, where users can safely delegate the integrity checking tasks to third party auditors and be worry free to use the cloud storage services.

## III. PROBLEM STATEMENT

### *A. System Model*

This application involves three parties: the cloud server, the third party auditor (TPA) and users. There are two types of users in a group: the original user and a number of group users.

The original user and group users are both members of the group. Group members are allowed to access and modify shared data created by the original user based on access control polices. Shared data and its verification information (i.e. signatures) are both stored in the cloud server. The third party auditor is able to verify the integrity of shared data in the cloud server on behalf of group members. Our system model includes the cloud server, the third party auditor and users. The user is responsible for deciding who is able to share her data before outsourcing data to the cloud. When a user wishes to check the integrity of shared data, she first sends an auditing request to the TPA. After receiving the auditing request, the TPA generates an auditing message to the cloud server, and retrieves an auditing proof of shared data from the cloud server. Then the TPA verifies the correctness of the auditing proof. Finally, the TPA sends an auditing report to the user based on the result of the verification.

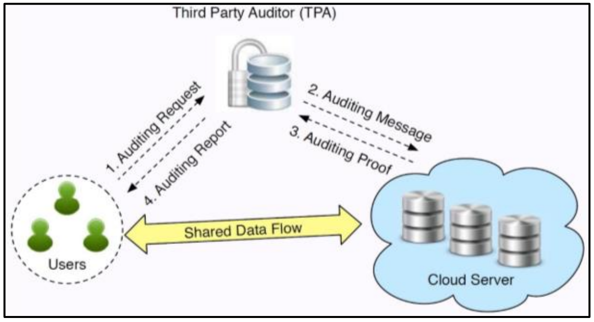

Fig. 1: System Model

### *B. Threat Model*

#### *1) Integrity Threats*

Two kinds of threats related to the integrity of shared data are possible. First, an adversary may try to corrupt the integrity of shared data and prevent users from using data correctly. Second, the cloud service provider may inadvertently corrupt (or even remove) data in its storage due to hardware failures and human errors. Making matters worse, in order to avoid this integrity threat the cloud server provider may be reluctant to inform users about such corruption of data.

#### *2) Privacy Threats*

The identity of the signer on each block in shared data is private and confidential to the group. During the process of auditing, a semi-trusted TPA, who is only responsible for auditing the integrity of shared data, may try to reveal the identity of the signer on each block in shared data based on verification information. Once the TPA reveals the identity of the signer on each block, it can easily distinguish a high-value target (a particular user in the group or a special block in shared data).

### *C. Design Objectives*

To enable the TPA efficiently and securely verify shared data for a group of users, Oruta should be designed to achieve following properties:

1) Public Auditing The third-party auditor is able to publicly verify the integrity of shared data for a group of users without retrieving the entire data.
2) Correctness The third-party auditor is able to correctly detect whether there is any corrupted block in shared data.
3) Unforgeability Only a user in the group can generate valid verification information on shared data.
4) Identity Privacy During auditing, the TPA cannot distinguish the identity of the signer on each block in shared data.

## IV. METHODOLOGY

### *A. Privacy Preserving Public Auditing Module*

The details of our public auditing mechanism in Oruta includes: Key-Gen, Sig-Gen, Modify, Proof-Gen and Proof Verify. In Key-Gen, users generate their own public/private key pairs. In Sig-Gen, a user is able to compute ring signatures on blocks in shared data. Each user is able to perform an insert, delete or update operation on a block, and compute the new ring signature on this new block in Modify. Proof Gen is operated by the TPA and the cloud server together to generate a proof of possession of shared data. In Proof-Verify, the TPA verifies the proof and sends an auditing report to the user.

The proposed scheme is as follows:

- Setup Phase
- Audit Phase

#### *1) Setup Phase*

The user initializes the public and secret parameters of the system by executing KeyGen, and pre-processes the data file F by using SigGen to generate the verification of metadata. The user then stores the data file F and the verification metadata at the cloud server. The user may alter the data file F by performing updates on the stored data in cloud.

#### *2) Audit Phase*

TPA issues an audit message to the cloud server to make sure that the cloud server has retained the data file F properly at the time of the audit. The cloud server will create a response message by executing Gen proof using F and its verification metadata as inputs. The TPA then verifies the response by cloud server via Verify Proof.

A owner is a person who can access resources from the cloud. The owner would first register to the interface to get the services with the valid username and password. In order to correctly audit the integrity of the entire data, a public verifier needs to choose the appropriate public key for each block. Then they can request for the file to the cloud service admin. There will be a third-party auditor who performs the integrity checking of the data before providing it to the owner or the users. This is done by 1st splitting the data into blocks and then performing integrity check. The owner has the option of downloading the verified file and also uploads new files.

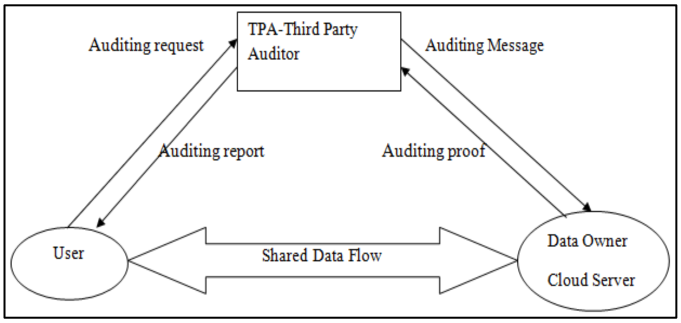

Fig. 2: System Architecture

### *B. Batch Auditing Module*

With the establishment of privacy-preserving public auditing in Cloud Computing, TPA may concurrently handle multiple auditing delegations upon different users' requests. The individual auditing of these tasks for TPA can be tedious and very inefficient. Batch auditing not only allows TPA to perform the multiple auditing tasks simultaneously, but also greatly reduces the computation cost on the TPA side. Given K auditing delegations on K distinct data files from K different users, it is more advantageous for TPA to batch these multiple tasks together and audit at one time.

The TPA registers to the application with a valid username and password. TPA logins to the application and verifies the integrity of data. TPA views all the list of files uploaded by the owner without the key. Has the privilege of encrypting the data and save it on cloud. TPA also view data which is uploaded by various owner.

### *C. Data Dynamics Module*

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. We can adopt this technique in our design to achieve privacy-preserving public risk auditing with support of data dynamics.

To enable each user in the group to easily modify data in the cloud and shared the latest version of the data with rest of the group, oruta should also support dynamic operations on shared data. A dynamic operation includes an insert, delete or update operation on a single block. However, since the computation of a ring signature includes an identifier of a block, traditional methods, which only use the index of a block as its identifier, are not suitable for supporting dynamic operations on shared data. The reason is that, when user modifies a single block in shared data by performing an insert or delete operation, the indices of blocks that after the modified block are all changed and the changes of these indices requires users to recomputed the signatures of these blocks, even though the content of these blocks are not modified. The details of our public auditing mechanism in Oruta includes: Key-Gen, Sig-Gen, Modify, Proof-Gen and Proof Verify. In Key-Gen, users generate their own public/private key pairs. In Sig-Gen, a user is able to compute ring signatures on blocks in shared data. Each user is able to perform an insert, delete or update operation on a block, and compute the new ring signature on this new block in Modify. Proof Gen is operated by the TPA and the cloud server together to generate a proof of possession of shared data. In Proof-Verify, the TPA verifies the proof and sends an auditing report to the user.

- Modify: A user in the group modifies the block in the shared data by performing one of the following three operations:
- Insert: The user inserts the new block say mj into shared data. Total number of blocks in shared data is n. He/She computes the new identifier of the inserted block mj as idj={vj,rj}where idj=identifier of jth block, vj=Virtual index. For the rest of blocks, identifiers of these blocks are not changed. This user outputs the new ring signature of the inserted block with SignGen, and uploads to the cloud server. Total number of blocks in shared data increases to n+1.
- Delete: The user deletes the block mj, its identifier idj and ring signature from the cloud server. The identifiers of other blocks in shared data are remains the same. The total number of blocks in shared data in cloud decreases to n-1.
- Update: The user updates the jth block in shared data with a new block mj. The virtual index of this block remain the same and new ring signature is computed. The user computes the new identifier of the updated block. The identifiers of other blocks in shred data are not changed. The user outputs the new ring signature of new block with SignGen, and uploads to the cloud server. The total number of blocks in shared data is still n.

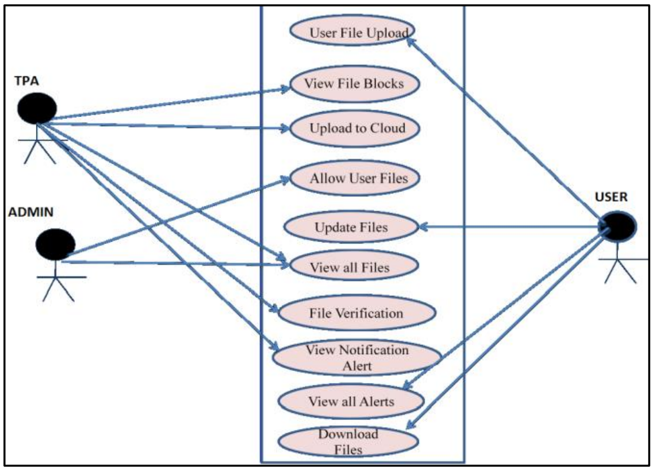

Fig. 3: Use case diagram

### *1) Dynamic operation for group users*

The dynamicity can be achieved for number of users in the group.

The group member can be added dynamically as well as any user can be revoked from the group

#### a) Algorithm

The RSA algorithm is used for key generation. RSA is one of the first practical public-key cryptosystems and is widely used for secure data transmission. In such a cryptosystem, the encryption key is public and differs from the decryption key which is kept secret. In RSA, this asymmetry is based on the practical difficulty of factoring the product of two large prime numbers, the factoring problem. RSA is made of the initial letters of the surnames of Ron Rivest, Adi Shamir, and Leonard Adleman, who first publicly described the algorithm in 1977.

RSA involves a *public key* and a *private key*. The public key can be known by everyone and is used for encrypting messages. Messages encrypted with the public key can only be decrypted in a reasonable amount of time using the private key. The keys for the RSA algorithm are generated the following way:

1) Choose two distinct prime numbers $p$ and $q$.
- For security purposes, the integers $p$ and $q$ should be chosen at random, and should be of similar bit-length. Prime integers can be efficiently found using a primality test.

2) Compute $n = pq$.
- $n$ is used as the modulus for both the public and private keys. Its length, usually expressed in bits, is the key length.

3) Compute $\varphi(n) = \varphi(p)\varphi(q) = (p - 1)(q - 1) = n - (p + q - 1)$, where $\varphi$ is Euler's totient function. This value is kept private.
4) Choose an integer e such that $1 < e < \varphi(n)$ and $\gcd(e, \varphi(n)) = 1$; i.e., e and $\varphi(n)$ are coprime.
- $e$ is released as the public key exponent.
- $e$ having a short bit-length and small Hamming weight results in more efficient encryption – most commonly $2^{16} + 1 = 65{,}537$. However, much smaller values of $e$ (such as 3) have been shown to be less secure in some settings.[8]

5) Determine d as $d \equiv e^{-1} (\text{mod } \varphi(n))$; i.e., d is the modular multiplicative inverse of e (modulo $\varphi(n)$).
- This is more clearly stated as: solve for $d$ given $d \cdot e \equiv 1 \pmod{\varphi(n)}$
- This is often computed using the extended Euclidean algorithm. Using the pseudocode in the *Modular integers* section, inputs $a$ and $n$ correspond to $e$ and $\varphi(n)$, respectively.
- $d$ is kept as the private key exponent.

The public key consists of the modulus n and the public (or encryption) exponent e. The private key consists of the modulus n and the private (or decryption) exponent d, which must be kept secret. p, q, and $\varphi(n)$ must also be kept secret because they can be used to calculate d.

- An alternative, used by PKCS#1, is to choose $d$ matching $de \equiv 1 \pmod{\lambda}$ with $\lambda = \text{lcm}(p - 1, q - 1)$, where lcm is the least common multiple. Using $\lambda$ instead of $\varphi(n)$ allows more choices for $d$. $\lambda$ can also be defined using the Carmichael function, $\lambda(n)$.

Since any common factors of (p-1) and (q-1) are present in the factorisation of p*q-1,[9] it is recommended that (p-1) and (q-1) have only very small common factors, if any besides the necessary 2.[10]

#### b) Encryption

Alice transmits her public key $(n, e)$ to Bob and keeps the private key $d$ secret. Bob then wishes to send message $M$ to Alice.

He first turns $M$ into an integer $m$, such that $0 \le m < n$ and $\gcd(m, n) = 1$ by using an agreed-upon reversible protocol known as a padding scheme. He then computes the ciphertext $c$ corresponding to

$$c \equiv m^e \pmod{n}$$

This can be done efficiently, even for 500-bit numbers, using Modular exponentiation. Bob then transmits $c$ to Alice.

Note that at least nine values of $m$ will yield a ciphertext $c$ equal to $m$,[note 1]

c) Decryption

Alice can recover *m* from *c* by using her private key exponent *d* via computing

$$m \equiv c^d \pmod{n}$$

The Advanced Encryption Standard (AES), also referenced as Rijndael (its original name), is a specification for the encryption of electronic data established by the U.S. National Institute of Standards and Technology (NIST) in 2001.The key size used for an AES cipher specifies the number of repetitions of transformation rounds that convert the input, called the plaintext, into the final output, called the cipher text. The number of cycles of repetition is as follows:

- 10 cycles of repetition for 128-bit keys. 12 cycles of repetition for 192-bit keys.
- 14 cycles of repetition for 256-bit keys.

Each round consists of several processing steps, each containing four similar but different stages, including one that depends on the encryption key itself. A set of reverse rounds are applied to transform cipher text back into the original plaintext using the same encryption key.

d) High-level description of the algorithm

1) KeyExpansions—round keys are derived from the cipher key using Rijndael's key schedule. AES requires a separate 128-bit round key block for each round plus one more.
2) Initial Round
    - AddRoundKey each byte of the state is combined with a block of the round key using bitwise xor.
3) Rounds
    - Sub Bytes—a non-linear substitution step where each byte is replaced with another according to a lookup table.
    - Shift Rows—a transposition step where the last three rows of the state are shifted cyclically a certain number of steps.
    - Mix Columns—a mixing operation which operates on the columns of the state, combining the four bytes in each column.
    - AddRoundKey
    - Final Round (no Mix Columns)
    - Sub Bytes
    - Shift Rows
    - AddRoundKey

## V. CONCLUSION

We propose Oruta, the first privacy preserving public auditing mechanism for shared data in the cloud. We utilize ring signatures to construct homomorphic authenticators, so the TPA is able to audit the integrity of shared data, yet cannot distinguish who is the signer on each block, which can achieve identity privacy. To improve the efficiency of verification for multiple auditing tasks, we further extend our mechanism to support batch auditing. An interesting problem in our future work is how to efficiently audit the integrity of shared data with dynamic groups while still preserving the identity of the signer on each block from the third-party auditor.

## REFERENCES

[1] M. Armbrust, A. Fox, R. Griffith, A. D.Joseph, R. H.Katz, A. Konwinski, G. Lee, D. A. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, ―A View of Cloud Computing,‖ Communications of the ACM, vol. 53, no. 4, pp. 50–58, Apirl 2010.
[2] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and D. Song, ―Provable Data Possession at Untrusted Stores,‖ in Proc. ACM Conference on Computer and Communications Security (CCS), 2007, pp. 598–610.
[3] C. Wang, Q. Wang, K. Ren, and W. Lou, ―Privacy-Preserving Public Auditing for Data Storage Security in Cloud Computing,‖ in Proc. IEEE International Conference on Computer Communications (INFOCOM), 2010, pp. 525–533.
[4] R. L. Rivest, A. Shamir, and Y. Tauman, ―How to Leak a Secret,‖ in Proc. International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT). SpringerVerlag, 2001, pp. 552–565.
[5] D. Boneh, C. Gentry, B. Lynn, and H. Shacham, ―Aggregate and Verifiably Encrypted Signatures from Bilinear Maps,‖ in Proc. International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT). Springer-Verlag, 2003, pp. 416–432.
[6] H. Shacham and B. Waters, ―Compact Proofs of Retrievability,‖ in Proc. International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT). SpringerVerlag, 2008, pp. 90–107.
[7] Y. Zhu, H. Wang, Z. Hu, G.-J. Ahn, H. Hu, and S. S.Yau, ―Dynamic Audit Services for Integrity Verification of Outsourced Storage in Clouds,‖ in Proc. ACM Symposium on Applied Computing (SAC), 2011, pp. 1550–1557.
[8] S. Yu, C. Wang, K. Ren, and W. Lou, ―Achieving Secure, Scalable, and Fine-grained Data Access Control in Cloud Computing,‖ in Proc. IEEE International Conference on Computer Communications (INFOCOM), 2010, pp. 534–542.
[9] D. Boneh, B. Lynn, and H. Shacham, ―Short Signature from the Weil Pairing,‖ in Proc. International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT). Springer-Verlag, 2001, pp. 514–532.