Journal of Computer Science 10 (10): 2077-2087, 2014
ISSN: 1549-3636
© 2014 Science Publications
doi:10.3844/jcssp.2014.2077.2087 Published Online 10 (10) 2014 (http://www.thescipub.com/jcs.toc)

A MODIFIED ELLIPTIC CURVE DIGITAL SIGNATURE ALGORITHM FOR PUBLIC VERIFIABILITY WITH DATA DYNAMICS IN CLOUD COMPUTING

[1]Sumathi, D. and [2]Ramachandra V. Pujeri

[1]Department of Computer Science and Engineering,
PPG Institute of Technology, Coimbatore, India
[2]Department of Computer Science and Engineering,
KGiSL Institute of Technology and Engineering, Coimbatore, India

Received 2014-02-27; Revised 2014-03-05; Accepted 2014-06-28

ABSTRACT

Cloud storage is considered to be the most critical factor in decision making for users as it largely scales down the infrastructure in terms of size, cost and design. Considering factors such as local storage cost, maintenance a single server model can support multiple users on a needed basis. This raises concerns for integrity verification i.e., assuring the correctness of the data stored available in cloud. The proposed auditing algorithm suggests and investigates digital signature for integrity verification. A Modified Version of Elliptic curve digital Signature Algorithm is proposed for auditing the task. The main focus of this study is to address problems such as privacy preserving, public auditing. In addition, the performance of the auditing task is optimized. Data dynamics have been modeled through various data operations such as block insertion, deletion and block modification. Extensive theoretical and experimental analysis presented in the paper shows that security, performance of the proposed algorithm are improved in terms of verification time of the auditing process.

Keywords: Cryptography, Data Security, Digital Signatures, Information Security, Privacy

1. INTRODUCTION

Data Storage has been widely considered the prominent snag among the services of Cloud Computing. Nowadays, the number of clients for storing their data has been on the increase. Moving data into the cloud helps the end users by freeing up hard disk space and it also reduces the cost of maintenance because the data center handles it. One of the biggest concerns in the cloud data storage is data integrity verification. Examples of data loss incidents of cloud storage services emerge from time to time (Armbrust *et al*., 2009). As users do not maintain the local copy of the outsourced data, there are chances for the cloud service providers to deceit the user about the status of the outsourced data. Cloud service providers have the chance to hide the loss occurred to the outsourced data from the users. Various methods have been proposed for data correctness verification on behalf of cloud users (Buyya *et al*., 2009). The verification of the outsourced data must be done periodically by the user. The cost of verification of the outsourced data is considered to be formidable and expensive for cloud users (Ateniese *et al*., 2007; Buyya *et al*., 2009; Wang *et al*., 2009b). Hence, the cloud users may possibly offer the auditing service to the Third Party Auditor (TPA). The TPA acts as a representative of cloud users who can often check the integrity of the data stored in the cloud. The result of the auditing task will be in favor of the cloud users to improve their cloud based service platform.

The idea of public auditability has been discussed in various papers (Wang *et al*., 2011; 2009a; 2010b; 2009c). Public auditability allows the third party auditor to do the auditing task. Privacy of the data is

Corresponding Author: Sumathi, D., Department of Computer Science and Engineering, PPG Institute of Technology, Coimbatore, India

considered to be an important factor because the auditing task is done by a third party auditor (Ateniese *et al*., 2007; Marshal, 2013; Erway *et al*., 2009; Hao et al., 2011) and also the data may be exposed to the TPA. This kind of complication affects the security of the protocols in cloud computing. Public auditability can be done with the help of the TPA without retrieving the whole copy of the data outsourced. Remote integrity protocols have been implemented in (Armbrust *et al*., 2009; Ateniese *et al*., 2007; Buyya *et al*., 2009) which concentrates not only on public auditability but also on privacy against verifier, data dynamics and public verifiability. The advanced features of Remote data integrity checking protocols are as follows:

- Privacy against verifier: The representative of the cloud user audits the outsourced data and reports to the cloud user without acquiring any knowledge of the content that has been verified
- Data dynamics: Data stored in the cloud can be accessed by the users. In addition to the access feature, users might do other operations such as modification, deletion and insertion on the outsourced data
- Public verifiability: This feature allows any of the clients to perform the auditing task and report to the data owner if any discrepancies are found

The features of remote data integrity checking protocol are considered in all real time applications. A document uploaded by the user can be accessed by other clients based on the mutual trust developed between them. In addition to the access facility, data owner may further modify, delete or append a portion of the data uploaded earlier. Hence during the design of remote data integrity checking protocol certain factors such as data dynamics, public audit and privacy must be taken into consideration. Advanced features of remote data integrity checking protocols have been focused in protocols (Xiangtao and Yifa, 2012). Ateniese *et al*. (2007) has devised a protocol to support data dynamics at the block level including operations such as modification, insertion and deletion. The protocol also in turn supports the block appending operation. Wang *et al*. (2011) has concentrated on the special feature “data dynamics”. The protocols in (Wang *et al*., 2009a; 2011) support privacy against third party verifiers. The key factors of cloud data storage are public auditability and data dynamics. An efficient algorithm has been proposed by considering the factors like data dynamics, public auditability and privacy against verifier.

Our contribution can be summarized into three main points:

- An efficient modified version of digital signature algorithm is proposed to implement the public verifiability scheme This algorithm efficiency is proved through concrete implementation and comparisons with the state of the art
- Our scheme is extended to support full dynamic data operations performed in block level
- Theoretical and experimental analysis has been done to test the efficiency of the algorithm

2. RELATED WORK

Traditional cryptographic methods for data integrity based on hash functions cannot work on the outsourced data. Downloading the entire content of the file is not a practical solution for data validation, because it leads to time consumption and expensive transaction. Various traditional approaches such as challenge response protocol have been used to check for data validation in cloud storage. Certain researchers have focused their attention on remote data possession checking schemes to prove the data integrity through public auditability. Remote data possession schemes can be categorized in two types namely Provable Data Possession (PDP) (Wang and Ren, 2010a) and Proof Of Retrievability (POR) (Ateniese *et al*., 2007; Shacham and Waters, 2013). The difference between PDP and POR is that POR can check the possession of data and recover data in case of failure and also detect the integrity of the data if it is tampered with even below a threshold level. Ateniese et al. (2007) developed a scheme for Provable Data Possession (PDP) model which utilizes the RSA-based homomorphic authenticators for auditing outsourced data and the scheme involves sampling a few random blocks of the file (Ateniese *et al*., 2007). This scheme does not support public auditability and the number of audits is limited to a bound. Xiangtao and Yifa (2012) proposed a new remote data integrity checking scheme for cloud storage which uses a “Proof Of Retrievability” (POR) model to give a more meticulous proof of their scheme (Xiangtao and Yifa, 2012). To ensure both possession and retrievability of remote data files they have used spot-Checking and (ECC) error correcting codes. This scheme lacks the support of dynamic updates and public auditability. Proposed the first method in the dynamic PDP scheme (Erway *et al*., 2009). Skip data structure has been implemented to facilitate data possession with dynamic support. Its efficiency is questionable because the search time to insert a block and finding a particular block is longer than is in trees. Wang *et al*. (2009a) implements a dynamic architecture for public checking (Wang *et al*., 2011). The challenge-response protocol is used for determining the data correctness, hence possible errors are located. However, the performance of this protocol greatly affects the practical application of their scheme. Zhuo Hao, Sheng Zhong and Nenghai Yu recommend privacy-preserving remote data integrity checking protocol with data dynamics and public verifiability (Hao *et al*., 2011). The drawback with this scheme is that there is no clear mapping relationship between the data and the tags. Data dynamics is supported only at the block level. Data integrity must be verified frequently by the data owner through the auditing task. Frequent auditing task leads to time consumption and it is expensive also. Various researchers suggest solutions by introducing the Third Party Auditor. As a result of this, the cloud storage providers and data owners can choose a Third Party Auditor (TPA) for the periodic auditing of the data outsourced by the data owner as proposed by (Zhu *et al*., 2012). Consider a storage system which consists of cloud service provider that operates cloud server, a client who uploads a file onto the cloud and a Third Party Auditor (TPA) who computes and verifies for data integrity. The clients store their data in the cloud server without taking a copy of it. For any client to check for the data integrity, it is of critical importance (Yang *et al*., 2012) that the server must ensure for data integrity. If the cloud server modifies any piece of data, the client must be able to discern it. Data must be kept private against the third party verifier. Juels and Kaliski (2007) proposed a scheme by introducing special scheme blocks called sentinels among the data blocks for proof of retrievability. To ensure data integrity verification the sentinels have to be verified by the verifier. Maintaining the sentinels at the data owner side leads to the storage overhead when the thin clients are used. Error correcting codes along with the sentinels are stored on the server side. The drawback in this scheme is that the cost of storage is high. Sravan and Saxena (2011) encrypt only a few bits of data per data blocks thus reducing the computational cost.

They introduced a metadata verification scheme for integrity verification. This may not be suitable for all applications and files of large size. Hao *et al*. (2011) devised a new remote data integrity checking protocol which involves data dynamics at block level and it supports public verifiability without the help of a third party auditor. An efficient modified version of Elliptic curve digital signature algorithm is implemented to reduce the data integrity verification time. We have tested the efficiency of the algorithm through theoretical and experimental analysis.

2.1. Data Integrity Checking

The paper proposes a remote data integrity checking protocol for cloud storage that addresses the following features:

- Confidentiality: The confidentiality of the outsourced data is protected from CSP and TPA
- Authentication: An authenticated user can access the document by the mutual trust maintained between the parties
- Data dynamics: Data can be remotely updated by the corresponding data owner through operations such as modification, deletion, append and insertion
- Privacy against verifier: Verifier can perform the task without the knowledge of the data

Small key size: Keys generated and used for encryption are considered to be comparatively smaller in size. The size of a DSA public key is at least 1024 bits, whereas the size of an ECDSA public key would be 160 bits. At a security level of 80 bits, an attacker needs the equivalent of about 2^{80} signature generations to find the private key which is very difficult.

3. SYSTEM MODEL

A cloud storage system consists of three entities namely Data Owner (DO) and a Cloud Server (CS) under the control of a Cloud Service Provider (CSP) (as illustrated in **Fig. 1** In this cloud computing paradigm, the DO stores the data in the CS without retaining a local copy of it. The critical importance in outsourcing data is that either the data owner or any public verifier has to verify the data for integrity verification. When the server modifies the data, the DO must be able to detect it. Data integrity verification task is done by computing digital signatures (Lindell, 2010). The verifier queries the server randomly to provide the hash value of the message. Keeping the hash value of the message, the verifier computes the digital signature and sends it to the DO. The DO then verifies the digital signature and identifies the modifications found if any. To ensure the correctness of the user's data, an efficient algorithm based on digital signatures is used. Digital signatures are used for revealing the identity of the sender of the document and also ensuring that the original content of the document sent by the cloud service provider is not modified.

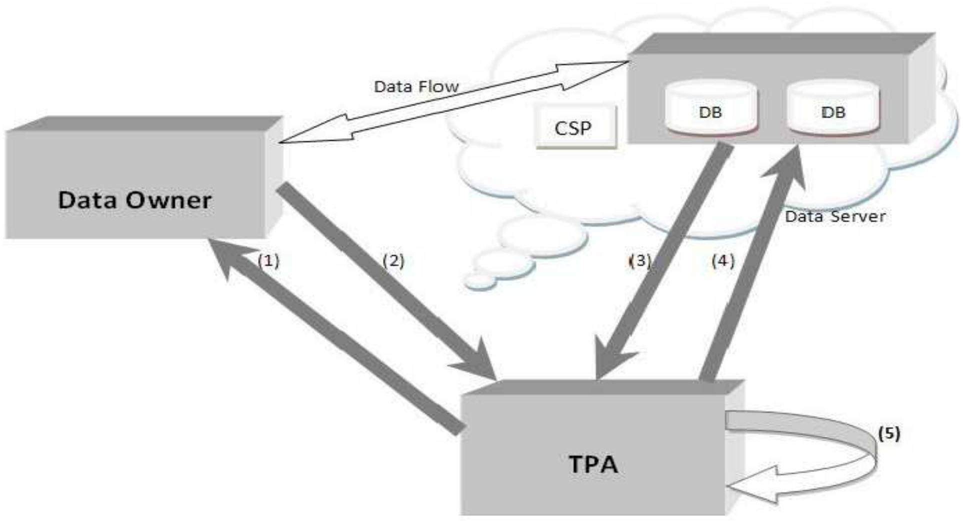


Fig. 1. Architecture of the proposed protocol

Among the various digital signatures, the Elliptic Curve Digital Signature Algorithm (ECDSA) is the fastest signature algorithm generating the short signature. But it is known widely that the verification process of ECDSA is slower than that of the signature generation process. Hence, an algorithm to prove the data integrity by using a modified Elliptic Curve Digital Signature Algorithm (ECDSA) is proposed. The objective of the proposed algorithm is to improve the performance of the integrity verification and optimize the audit schedule by reducing the time complexity of the signature verification. And, the improvement in the digital signature is proved through mathematical relationships.

4. ARCHITECTURE OF THE PROPOSED MODEL

The Data Owner (DO) is an entity who uploads a file or an archive onto the cloud. DO's rely on the cloud service provider for the data maintenance and computation. Cloud Server (CS) is an entity who receives the data sent by the data owner. The cloud server stores the data and the CSP controls the cloud server. Signature generation and verification is done with the help of elliptic curve digital signatures. DO preprocess the file and generates signatures for all the blocks:

- Post request for data integrity
- Sends feedback
- Hash code
- Request for hash code
- Evaluates Signature Scheme & Compares

5. ELLIPTIC CURVE CRYPTOGRAPHY

Elliptic curve can be defined over finite fields and real numbers. An elliptic curve (Amara and Siad, 2011) over the real numbers is defined in the form of the equation as $y^2 = x^3+ax+b \pmod p$, where x, y, a and b are real numbers. An elliptic curve group over the real numbers and finite fields (Shanmugalakshmi and Prabu, 2009; Khalique *et al*., 2010) consists of the points on the curve, along with a special point ∞, called the point at infinity which will be the identity element.

The order of the elliptic curve can be defined as P € E(Fq), where q = p or q = 2 m where m is the smallest integer r such that rP = α. The order of the curve, is the number of points of E(F), donated by #E(F). In our paper, the order of the curve is determined by the block size of the data that the user uploads onto the cloud. Data integrity is verified by downloading the entire content of the file or metadata of the file. Accessing the entire file and checking for integrity verification leads to I/O cost and time constraints. Hence, to overcome these drawbacks, a scheme is devised to verify data integrity by computing the digest value (M_d) for the whole data. The digest value sent by the server to the verifier is used for computing digital signatures and it is also verified. If both the values are same, then a report is generated to show that the data has not been modified by the server or any intruder.

In this study a remote data integrity protocol based on Elliptic curve digital signature is implemented. We consider a file 'F' of size 'n' is divided into blocks 'b' of equal lengths 'n' where f = $b_1, b_2, b_3,\ldots. b_n$ and b = |f|/n. Data owner preprocesses the file before uploading it onto the cloud. Data owner computes the signatures for the blocks since the files are divided into blocks. The data owner maintains a table to store the information about the uploaded files as shown in **Table 1**. The table contains the following fields.

TPA maintains the same information maintained by the DO. **Table 2** is maintained by the CSP. CSP receives the information regarding the operation that has to be performed on the block. This information is stored in a lookup table along with the location of the block. A table is maintained in order to reduce the searching time.

DO might randomly chose the blocks for integrity verification. When the DO needs to verify the integrity of the data, it posts a request to the TPA which in turn sends the request to CSP. TPA posts the request to CSP to compute the hash code for the corresponding block. CSP computes the hash code and sends to the TPA. TPA refers to the table and computes the signature with the received hash code and compares for variation. If there is no difference, it sends the message to the data owner. If the data owner needs to do any operation on the block, it has to run the function update (blk no, Key, Sign, 'M') to the TPA. TPA updates the table with the received information. It sends the information to the CSP and runs the function as opt2 (blkno, 'M'). When the CSP gets the information about the operation on the block, it does the corresponding operation on that block. The speed for identifying the location of the block is fast since its address is maintained in the table.

5.1. Key Generation

The data owner selects a random number x <------ Z_p and computes the private and public key. The domain parameters D = (q, FR, a, b, P, n, h) and the associated public key, private key pair pp -------> (d, P_u) is used.

Table 1. Table maintained by the DO

S. No	Block No	Key	Signature	Operation
1	1	K1	S1	Either of this (m, i, d) (m-modification, I-insertion, d- deletion)
2	2	K2	S2	Either of this (m, i, d)
3	3	K3	S3	Either of this (m, i, d)

Table 2. Table maintained by cloud service provider

S. No	Block No	Opt status	Starting address	Ending address
1	1	Either of this (m,i,d)	FC0000	FCFFFF
2	2	Either of this (m,i,d)	1A000	1ABBB
3	3	Either of this (m,i,d)	5B000	5BFFFF

5.2. Signature Generation

Data owner runs the Signgen() to generate the signatures for each block. The files are divided into blocks and signatures are generated for all blocks F = $\{b_1, b_2, b_3\ldots\ldots b_n\}$. Functions that are carried over by the three entities DO, TTP, CSP have been discussed further.

5.3. Correctness and Security Definitions

Definition 1: Correctness

Our protocol proves to be correct if whatever algorithms are executed honestly, the output will accept correct signature (Lindell and Pinkas, 2009). The output of the function Verify (α, (U_n, V_n)) will be {TRUE} even after the blocks modifications.

Thus our protocol is proved to be correctness under computation scheme of two party protocol.

Definition 2: Security

Our protocol is said to secure against adversary since it cannot convince the verifier to accept the wrong result.

Consider a game between the antagonist and the challenger. The challenger makes a query to the verifier for the verification of certain blocks integrity by initiating the Keygen () and Signgen ().The verifier in turn posts the query to the prover for the hash code of the corresponding block for verification. The antagonist can play the part of the server. The antagonist runs the function and the hash value is generated. Hash value is calculated as mentioned in the Equation 1 below:

$$HComp(b_i, P_{uk}) \rightarrow \sum_{i=0}^{n} \beta_i \tag{1}$$

The goal of the antagonist is to win in the game by cheating the verifier in trying to generate valid responses (β_i) that are acceptable and pass the integrity verification.

6. ALGORITHM FOR MODIFICATION OF A BLOCK

Suppose the data owner wants to modify a block in a file, DO has to post the request to the TPA and CSP. DO sends the corresponding block number b_i^1 to replace with the block number b_i.

DO construct the function F (update, delete, insert) and sends it to the TPA and the server upon receiving the request, the TPA runs the function update-TPA (blkno, key, sign, 'M').The TPA replaces the new block number, new key and signature with the old block number and other details.

When the server receives the message as an update, it searches the block table for the corresponding block number and the location of the block where it has been stored. After identification, the server replaces the block b_i^1 with the block b_i and the server runs the function Update_CSP (blkno, Op_status, Beg_addr, End_addr) in order to modify the status and address of the block if any changes are done.

6.1. Algorithm for Insertion of a New Block

Data modification refers to modifying an existing block, whereas insertion refers to the addition of a new block. Suppose the data owner needs to insert a new block in the file, DO has to send a request to the TPA and CSP. DO execute the function F (update, delete, insert) and sends it to the TPA and the server.

When the TPA receives a request, the TPA constructs the function insert (b_i, b_i*, Key, Sign,' I'). The new block number b_i* will be inserted after the block b_i. As a result of this function, the block table gets updated with the new key.

6.2. Algorithm for Block Deletion

If the data owner wants to delete a block, DO make the function F (update, delete, insert) and sends the message to the TPA and CSP.

Upon receiving the request, the TPA executes the function delete (blkno, key, sign, 'D') in order to remove the details of the block. Deletion of a block will move the latter blocks forward.

When the server receives a request from the DO, it checks in the block table for the location of the block. It deletes the block from the corresponding location and the block table gets updated as a result of this.

Signgen (β_I, K_i): The data owner possesses the files that are to be uploaded onto the cloud. DO has to execute this function to compute signatures for the blocks. Before computing the signatures, the file F is divided into blocks {b_1, b_2, b_3,……,b_n}. For each block the data owner has to generate signatures by computing the function given below Equation 2 and 3:

$$Signgen(\beta_i, K_i) \rightarrow (U_n, V_n) \rightarrow \sum_{i=1}^{n} (\beta \times K_i + U_i) \bmod n \tag{2}$$

Let:

$$S_n \leftarrow (U_n, V_n) \tag{3}$$

Proof (β_i): When the DO wants to verify the integrity, it raises the query as a function chal (b_i) for random checking of the data. When the TPA receives the request, it passes on the request to the CSP. Therefore, the CSP in turn runs the function to compute the hash value of a block as calculated in Equation 1 and the hash value for the corresponding block is generated and sends (β_i) to the TPA. Upon receiving the hash value of the block (β_i), the TPA generates the signatures for the corresponding block for further verification:

$$proof(\beta_i) \rightarrow \alpha_i \tag{4}$$

Let:

$$\alpha_i = (V_i - U_i) G (\bmod f) \beta^{-1} \tag{5}$$

Verify (): This function is to check for the variations in signatures. When the TPA computes the signatures for the corresponding block based on the hash value sent by the CSP, TPA has to prove for the data integrity. It has to validate the response by executing this function.

Verify (α,(U_n, V_n))→{TRUE, FALSE}: If the result is true it shows that the signatures are the same, then TPA sends a message to the data owner that integrity is maintained. If the value is false then it proves for data integrity violation.

Considering Equation 4:

$$proof(\beta_i) \rightarrow \alpha_i$$

From Equation 5:

$$\begin{aligned} \alpha_i &= (V_i - U_i) G (\bmod f) \beta^{-1} \\ &= (\beta K_i + U_i - U_i) G (\bmod f) \beta^{-1} \end{aligned} \tag{6}$$

Cancelling U_i Equation 6 becomes:

$$= K_i G$$

Hence it is proved that from the Equation 5 and 6, the signatures computed are the same.

7. SECURITY AND MATHEMATICAL ANALYSIS OF THE PROPOSED PROTOCOL

This section deals with the security analysis of the proposed protocol in two ways. The proposed protocol is proved to be secure in two ways:

- If client and server are considered to be honest, the server can pass the verification phase effectively
- The server must be proved as a trusted server

Theorem 1

The server can pass the verification phase successfully if the client and server are considered to be honest.

7.1. Proving by theorem of contradiction

Assumption

Assume the negation that the server fails in the verification phase.

On receiving the request from the server, the server runs the function HComp (b_i, P_{uk}) and generates the hash value for the corresponding block. The server sends the hash code (β_i) to the TPA for further comparison. The TPA generates the signatures α_i by running the function proof (β_i). Then the TPA runs the function verify() to compare the signatures. If the function returns false, it is a proof that our assumption is false.

Conclusion

Thus it is proved to be true because it has an opposite truth value to the assumption.

Analysis

$$\text{To prove } Verify\left(\alpha,\left(U_n,V_n\right)\right)\rightarrow\left\{TRUE,FALSE\right\} \quad (7)$$

To prove Equation 7, consider (1):

$$Signgen\left(\beta_i,K_i\right)\rightarrow\left(U_n,V_n\right)\rightarrow\sum_{i=1}^{n}\left(\beta\times K_i+U_i\right)\bmod n$$

$$\sum_{i=1}^{n}\left(U_n,V_n\right)\Rightarrow\left(U_1,V_1\right)+\left(U_2,V_2\right)+\ldots..\left(U_n,V_n\right) \quad (8)$$

Equation 8 implies as

$$\begin{aligned}&(\beta_1\times K_1+U_1)\bmod n+(\beta_2\times K_2+U_2)\\&\bmod n+..+(\beta_n\times K_n+U_n)\bmod n\end{aligned} \quad (9)$$

Considering $S_i = (u_i, v_i)$

To verify the signature for a sample block,
From Equation 5:

$$\alpha_i=\left(V_i-U_i\right)G\left(\bmod f\right)\beta^{-1}$$

From Equation 6:

$$\left(\beta K_i+U_i-U_i\right)G\left(\bmod f\right)\beta^{-1}$$

Canceling u_i in Equation 5:

$$\beta_i.k_i.G\bmod f\,\beta_i^{-1}$$

By following a, $a^{-1} = 1$:

$$\therefore\left(k_i.G\right)\bmod f$$

Hence it is proved that the server passes the verification phase if the client and the server are honest.

Theorem 2

The server must be proved as a trusted server.

We can prove this theorem by the method of proof by contradiction.

Assumption

Assume the negation that the server is untrusted.

When the client or the data owner is ready to verify for the data integrity, the DO sends the request to the TPA which in turn sends the request to the CSP. The CSP must compute the hash code for the randomly chosen block (μ) and sends it to the TPA for further integrity verification. The TPA runs the function verify() to compare the signatures. If the function returns false, then it is proved that our assumption is false.

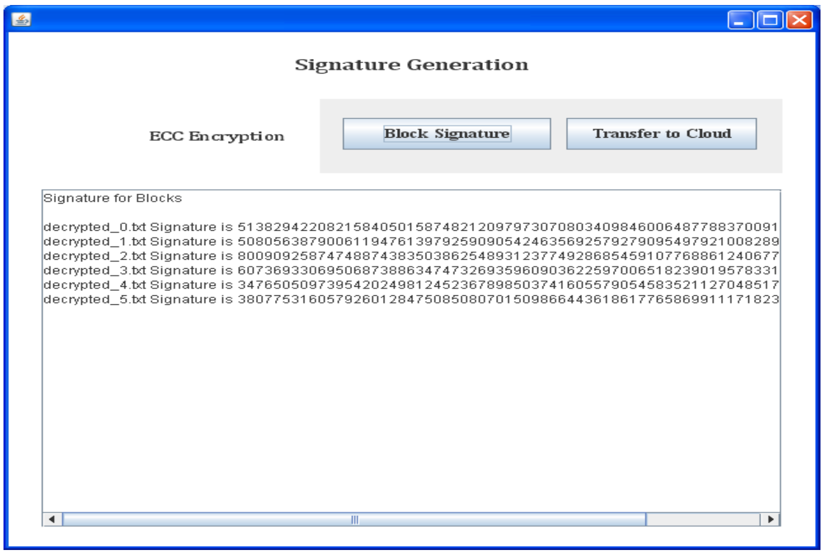
Conclusion

Thus it is proved to be true because it has an opposite truth value and hence contradiction.

8. EXPERIMENTAL RESULTS AND PERFORMANCE ANALYSIS OF OUR PROPOSED ALGORITHM

The auditing mechanism is carried over between the TPA and data owner. Our work is implemented in an Intel Core processor running at 1.86, GHz, 2048 MB of RAM, 250 GB Serial ATA drive. Our algorithms are implemented in the Java Language. The elliptic curve used in this algorithm is curved over prime fields GF (p) and it can be among P-192, P-224, P-256, P-384 and P-521. Among the curves we have used P-256 for showing the performance. The sample file size taken in our paper is 1 GB Using Elliptic curve digital signature algorithm, the signatures for the blocks has been calculated as shown in **Fig. 2**. A block table maintained by the DO is shown in **Fig. 2**.

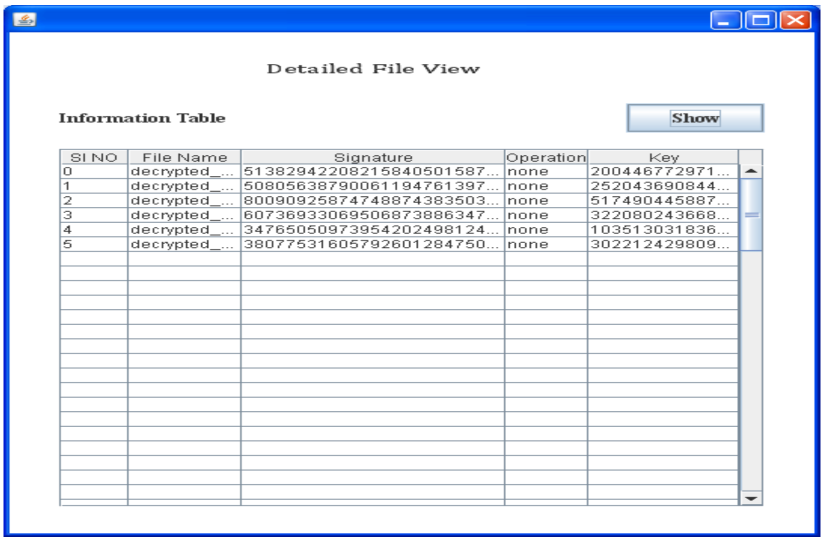
DO and TPA maintains a block table as shown below in **Fig. 3**. The operation field in the block table denotes the type of block operation that is to be performed by CSP. The block table contains signatures and keys for all the blocks. The field “Operation” shows the status of the block operation.



```
Signature for Blocks

decrypted_0.txt Signature is 5138294220821584050158748212097973070803409846006487788370091
decrypted_1.txt Signature is 5080563879006119476139792590905424635692579279095497921008289
decrypted_2.txt Signature is 8009092587474887438350386254893123774928685459107768861240677
decrypted_3.txt Signature is 6073693306950687388634747326935960903622597006518239019578331
decrypted_4.txt Signature is 3476505097395420249812452367898503741605579054583521127048517
decrypted_5.txt Signature is 3807753160579260128475085080701509866443618617765869911171823
```

Fig. 2. Signature generation for blocks



SI NO	File Name	Signature	Operation	Key
0	decrypted_...	51382942208215840501587...	none	200446772971...
1	decrypted_...	50805638790061194761397...	none	252043690844...
2	decrypted_...	80090925874748874383503...	none	517490445887...
3	decrypted_...	60736933069506873886347...	none	322080243668...
4	decrypted_...	34765050973954202498124...	none	103513031836...
5	decrypted_...	38077531605792601284750...	none	302212429809...

Fig. 3. Block table maintained by DO and TPA

CSP maintains the block table which stores the information about the location of the block stored in the space provided by the server in the provider. The field "Start Addr" and "End Addr" refers to the location of the block. The details are given in **Table 2**.

8.1. Comparison of Traditional and Proposed Modified Version of Ecdsa Algorithm

When the DO needs to check for data integrity, DO sends the request to Third Party Auditor (TPA). On receiving the request, the TPA verifies the signatures. The improved version of ECDSA is used by the TPA for signature verification. Traditional ECDSA takes much time for signature verification. But our proposed algorithm takes lesser time when compared with the time taken by the traditional ECDSA. The below **Fig. 4** shows the comparison of time taken for signature verification.

8.2. Analysis of Key Size and Computation Time

The evaluation of our proposed algorithm is done by measuring the storage size and key size and computational time. The simulation of DO, TPA and CSP has been done on a Windows system with an Intel core processor at 3.16 GHz CPU and with 16 GB RAM. The elliptic curve we used is P-256 with the file size as the order of the curve. Coding was written in Java and was tested on Windows platform. The file size starts varying from 10 KB. The proposed algorithm evaluation results are shown in terms of showing computation time and memory usage for various storage sizes. The Computation time increases when the storage size is increased. The computation time is increased for various storage sizes like 64 kbps, 128 kbps and 256 kbps. **Fig. 5** shows the illustrations of computation time for different no of users at storage 64 kbps.

The Computation time increases since the key size is increased. Increasing the key size will lead to increase in memory usage and it is shown in **Fig. 6**.

A case study has been done on considering the various storage size and different number of users for various session size. From the various graphs it has been interpreted that the computation time and memory size increases when the session size increases.

9. DISCUSSION

We compare our proposed scheme with other existing verification schemes as shown in **Table 3**. Hao et al. (2011) RSA has been used for verification. The key length for RSA has increased over recent years and thus it generates processing burden on applications using RSA. To avoid this burden, we proposed Elliptic Curve Digital Signature algorithm for verification scheme.

Our proposed scheme shows that it is private against the TPA since the data is encrypted before it is outsourced. Through our theoretical and experimental analysis our proposed protocol is proved to be efficient.

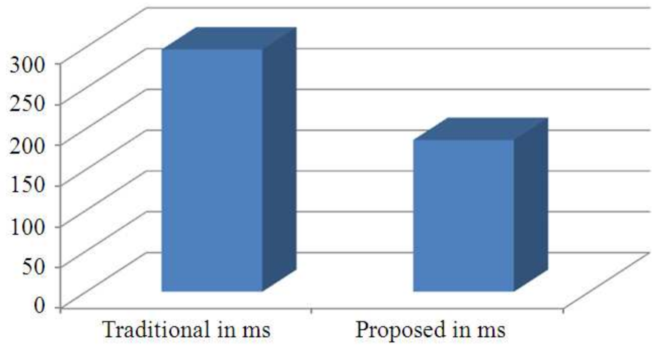


Fig. 4. Comparison of Traditional and Proposed ECDSA

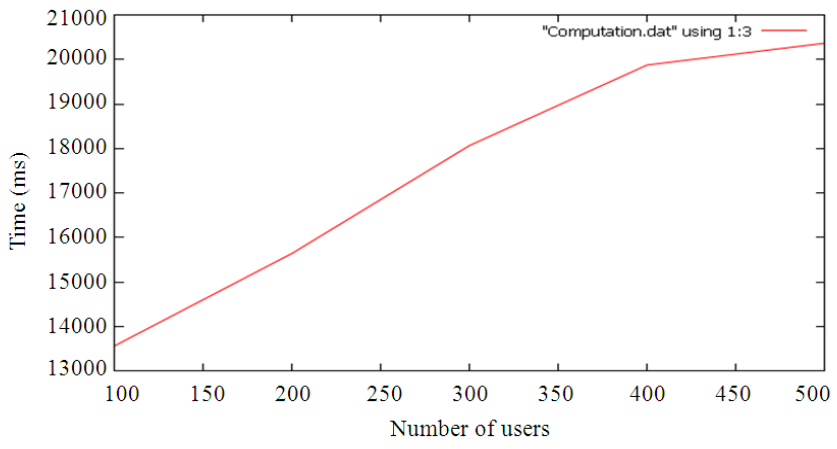


Fig. 5. Shows computation time for Different no of users at storage 64 kbps

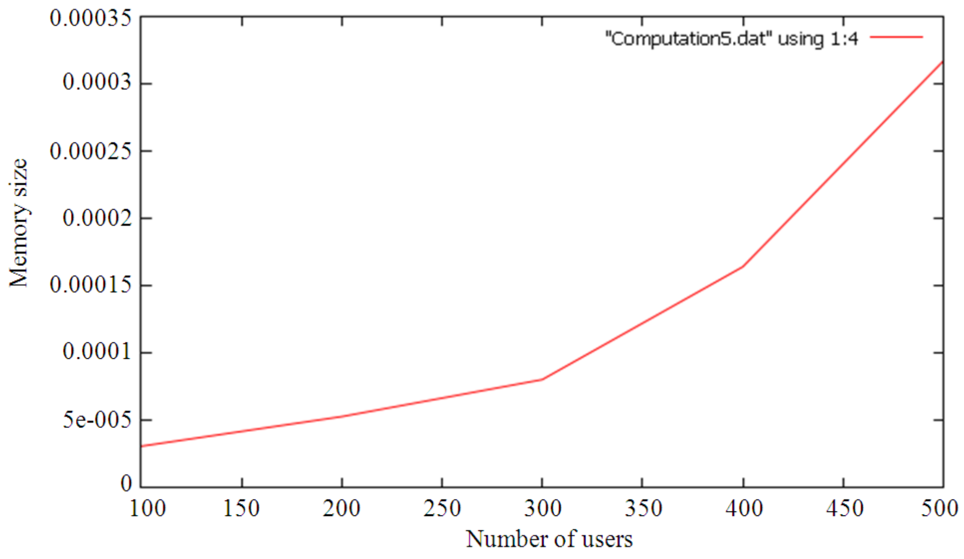


Fig. 6. Shows Memory usages for Different no of users at storage 128 kbps

Table 3. Comparisons of proposed algorithm with existing protocols

Type of guarantee	S-PDP(2) probabilistic/ deterministic	9 deterministic	Probabilistic 22	DPDP (Erway *et al*., 2009)	23, 24	25	Zhuo Hao deterministic	Proposed algorithm deterministic
With the help of TPA	No	No	Yes	No	Yes	Yes	No	Yes
Public verifiability	yes	no	Yes	No	Yes	Yes	Yes	Yes
Data dynamics	Append Only	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Server computation	O(c)	O(n)	O(c log n)	O(c log n)	O(c)	O(s)	O(n)	O(c)
Verifier computation	O(c)	O(n)	O(c log n)	O(c log n)	O(c)	O(c + s)	O(n)	O(c)

n is the block number, c is the sampling block number and s is the number of sectors in a block

10. CONCLUSION

In this study we propose an efficient modified version of Elliptic Curve Digital Signature Algorithm for data integrity verification task. The proposed algorithm supports data dynamics, public verifiability and proved to be secure against the auditing schedule. Extensive experimental and theoretical analysis shows that the proposed algorithm is provably secure and efficient. Accessing time of a particular block for deletion or modification is very fast since their location is been indexed.

The future work aims to extend the focus in one of the most important factor in security i.e. authentication that can be bound with integrity. The most critical characteristics of security namely CIA can be incorporated with the Service Level Agreements that acts as a legal bonding between the DO and CSP. Integration of CIA with SLA could be given more attention in our future work.

11. REFERENCES

Amara, M. and A. Siad, 2011. Elliptic Curve Cryptography and its applications. Proceedings of the 7th International Workshop on Systems, Signal Processing and their Applications, May 9-11, IEEE Xplore Press, Tipaza, pp: 247-250. DOI: 10.1109/WOSSPA.2011.5931464

Armbrust, M., A. Fox, R. Griffith, A.D. Joseph and R.H. Katz *et al*., 2009. Above the clouds: A berkeley view of cloud computing. University of California, Berkeley.

Ateniese, G., R. Burns, R. Curtmola, J. Herring and L. Kissner *et al*., 2007. Provable data possession at untrusted stores. Proceedings of the 14th ACM Conference on Computer and Communications Security, Oct. 28-31, ACM New York, pp: 598-609. DOI: 10.1145/1315245.1315318

Buyya, R., C.S. Yeo, S. Venugopal, J. Broberg and I. Brandic, 2009. Cloud computing and emerging IT platforms: Vision, hype and reality for delivering computing as the 5th utility. Future Generat. Comput. Syst., 25: 599-616. DOI: 10.1016/j.future.2008.12.001

Erway, C., A. Kupcu, C. Papamanthou and R. Tamassia, 2009. Dynamic provable data possession. Proceedings of the 16th ACM Conference on Computer and Communications Security, Nov. 09-13, ACM New York, pp: 213-222. DOI: 10.1145/1653662.1653688

Hao, Z., S. Zhong and N. Yu, 2011. A privacy-preserving remote data integrity checking protocol with data dynamics and public verifiability. IEEE Trans. Knowl. Data Eng., 23: 1432-1437. DOI: 10.1109/TKDE.2011.62

Juels, A. and B.S. Kaliski, 2007. PORs: Proofs of retrievability for large files. Proceedings of the 14th ACM Conference on Computer and Communications Security, Oct. 28-31, ACM, New York, pp: 584-597. DOI: 10.1145/1315245.1315317

Khalique, A., K. Singh and S. Sood, 2010. Implementation of elliptic curve digital signature algorithm. Int. J. Comput. Applic. 2: 21-27.

Lindell, Y. and B. Pinkas, 2009. Secure multiparty computation for privacy preserving data mining. J. Privacy Confidentiality, 1: 59-98.

Lindell, Y., 2010. Foundations of cryptography. Bar-Ilan University, Israel.

Marshal, S.V., 2013. Secure audit service by using TPA for data integrity in cloud system. Int. J. Innovat. Technol. Exp. Eng., 3: 2278-3075.

Shacham, H. and B. Waters, 2013. Compact proofs of retrievability. J. Cryptol., 26: 442-83. DOI: 10.1007/s00145-012-9129-2

Shanmugalakshmi, R. and M. Prabu, 2009. Research issues on elliptic curve cryptography and its applications. IJCSNS, 9: 19-22.

Sravan, K.R and A. Saxena, 2011. Data Integrity Proofs in Cloud Storage. Proceedings of the 3rd International Conference on Communication Systems and Networks, Jan. 4-8, IEEE Xplore Press, Bangalore, pp: 1-4. DOI: 10.1109/COMSNETS.2011.5716422

Wang, C., Q. Wang, K. Ren and W. Lou, 2009a. Ensuring data storage security in cloud computing. Proceedings of the17th International Workshop on Quality of Service, Jul. 13-15, IEEE Xplore Press, Charleston, SC., pp: 1-9. DOI: 10.1109/IWQoS.2009.5201385

Wang, Q., C. Wang, J. Li, K. Ren and W. Lou, 2009b. Enabling public verifiability and data dynamics for storage security in cloud computing. Proceedings of the 14th European Conference on Research in Computer Security, Sept. 21-23, Springer Berlin Heidelberg, Saint-Malo, France, pp: 355-370. DOI: 10.1007/978-3-642-04444-1_22

Wang, C., S.S.M. Chow, Q. Wang, K. Ren and W. Lou *et al*., 2009c. Privacy preserving public auditing for secure cloud storage.

Wang, C., K. Ren, W. Lou and J. Li, 2010a. Toward publicly auditable secure cloud data storage services. IEEE Netw., 24: 19-24. DOI: 10.1109/MNET.2010.5510914

Wang, C., Q. Wang, K. Ren and W. Lou, 2010b. Privacy-preserving public auditing for data storage security in cloud computing. Proceedings of the IEEE INFOCOM, Mar. 14-19, IEEE Xplore Press, San Diego, CA., pp: 1-9. DOI: 10.1109/INFCOM.2010.5462173

Wang, Q., C. Wang, K. Ren, W. Lou and J. Li, 2011. Enabling public auditability and data dynamics for storage security in cloud computing. IEEE Trans. Parallel Distributed Syst., 22: 847-859. DOI: 10.1109/TPDS.2010.183

Xiangtao, Y. and L. Yifa, 2012. A new remote data integrity checking scheme for cloud storage with privacy preserving. IEEE 14th International Conference on Communication Technology, Nov. 9-11, IEEE Xplore Press, Chengdu, pp: 704-708. DOI: 10.1109/ICCT.2012.6511296

Yang, Y., L. Wu, Y. Yan and C. Xu, 2012. Fine-grained data integrity check for power cloud computing. Proceedings of the 5th International Conference on Biomedical Engineering and Informatics, Oct. 12-16, IEEE Xplore Press, Chongqing, pp: 1346-1350. DOI: 10.1109/BMEI.2012.6513215

Zhu, Y., H. Hu, G.J. Ahn and S.S. Yau, 2012. Efficient audit service outsourcing for data integrity in clouds. J. Syst. Software, 85: 1083-1095. DOI: 10.1016/j.jss.2011.12.024