IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 5, Issue 7, July 2016

An Efficient and Secure Protocol for Ensuring Data Storage Security in Cloud Computing Using ECC

Santosh Kumar Singh[1], Dr. P.K. Manjhi[2], Dr. R.K. Tiwari[3]

Research Scholar, Department of Computer Applications, Vinoba Bhave University, Hazaribag, India [1]

Assistant Professor, University Department of Mathematics, Vinoba Bhave University, Hazaribag, India [2]

Professor, H.O.D CSE, R.V.S College of Engg & Tech., Jamshedpur, India [3]

Abstract: Computing applications and data are growing so rapidly that increasingly larger servers and data centre are needed for fast processing within the required time. A fundamental shift in the way Information Technology (IT) and computing services are being delivered and purchased, results in the development of cloud computing. Currently, there has been an increasing trend in outsourcing data to remote cloud, where the people outsource their data at Cloud Service Provider(CSP) who offers huge storage space with low cost. Thus users can reduce the maintenance and burden of local data storage. Meanwhile, once data goes into cloud they lose control of their data, which inevitably brings new security risks toward integrity and confidentiality. Hence, efficient and effective methods are needed to ensure the data integrity and confidentiality of outsource data on entrusted cloud servers. However, Cloud computing requires that organizations trust that a service provider's platforms are secured and provide a sufficient level of integrity for the client's data. In this paper, we propose an efficient and secure protocol to address these issues. Our design is based on Elliptic Curve Cryptography and Sobol Sequence (random sampling). Our method allows third party auditor (TPA) to periodically verify the data integrity stored at CSP without retrieving original data. The challenge-response protocol transmits a small, constant amount of data, which minimizes network communication. Most importantly, our protocol is confidential: it never reveals the data contents to the malicious parties. The proposed scheme also considers the dynamic data operations at block level while maintaining the same security assurance. To compare with existing schemes, our scheme is more secure and efficient.

Keywords: Data storage, Integrity, Confidentiality, Elliptic Curve Cryptography (ECC), Sobol Sequence, Cloud Computing, TPA, CSP.

I. INTRODUCTION

Several trends are opening up the era of Cloud Computing, which is an Internet-based development and use of computer technology. The ever cheaper and more powerful processors, together with the software as a service (SaaS) computing architecture, are transforming data centers into pools of computing service on a huge scale. The increasing network bandwidth and reliable yet flexible network connections make it even possible that users can now subscribe high quality services from data and software that reside solely on remote data centers. Moving data into the cloud offers great convenience to users since they don't have to care about the complexities of direct hardware management. The pioneer of Cloud Computing vendors, Amazon Simple Storage Service (S3) and Amazon Elastic Compute Cloud (EC2) [1] are both well known examples. While these internet-based online services do provide huge amounts of storage space and customizable computing resources, this computing platform shift, however, is eliminating the responsibility of local machines for data maintenance at the same time. As a result, users are at the mercy of their cloud service providers for the availability and integrity of their data. Recent downtime of Amazon's S3 [2] and Apple ICloud [3] are well known examples of cloud data storage. However, once data goes into cloud, the users lose the control over the data. This lack of control raises new formidable and challenging issues related to confidentiality and integrity of data stored in cloud [4]. The confidentiality and integrity of the outsourced data in clouds are of paramount importance for their functionality. The reasons are listed as follows [5]:

- The CSP, whose purpose is mainly to make a profit and maintains a reputation, has intentionally hide data loss an incident which is rarely accessed by the user's
- The malicious CSP might delete some of data or is able to easily obtain all the information and sell it to the biggest rival of Company.
- An attacker who intercepts and captures the communications is able to know the user's sensitive information as well as some important business secrets.
- Cloud infrastructures are subject to wide range of internal and external threats.

Remote data integrity checking is a protocol that focuses on how frequently and efficiently we verify whether cloud server can faithfully store the user's data without retrieving it. In this protocol, the user generates some metadata. Later, he can challenge the server for integrity of certain file blocks through challenge-response protocol. Then the server generates responses that the server still possesses the data in its original form to corresponding challenge sent by the verifier who may be original user or trusted third party entity. Recently, several researchers have proposed different variations of remote data integrity checking protocols under different cryptography schemes [6]. However, all these protocols focus on static data verification. One of the design principles of cloud storage is to provide dynamic scalability of data for various applications. This means, the data stored in cloud are not only accessed by the users but also frequently updated through block operations such as modification, insert and delete operations. Hence, it is crucial to develop more secure and efficient mechanism to support dynamic audit services. The protocols to verify dynamic data in cloud are proposed in [7]. Although the existing schemes aim at providing integrity verification for different data storage systems, but problem of confidentiality of data has not been fully addressed. The protocols [8] have been proposed to ensure the confidentiality and integrity of remote data. But, all these schemes are unable to provide strong security assurance to the users, because these schemes verifying integrity of outsourced data based on pseudorandom sequence, which does not cover the whole data while computing the integrity proof. Therefore, probabilistic verification schemes based on pseudorandom sequence does not give guarantee to the users about security of their data. Syam et al. [9] proposed a distributed verification protocol using Sobol sequence to ensure availability and integrity of data, but it is also not addressed the data confidentiality issue.

How to achieve a secure and efficient design to seamlessly integrate these two important components for data storage service remains an open challenging task in Cloud Computing. In this paper, we propose an efficient and secure protocol to ensure the confidentiality and integrity of data storage in cloud computing using Elliptic Curve Cryptography (ECC) [10, 11, 12] and Sobol Sequence [13]. The ECC can offer same levels of security with small keys comparable to RSA and other PKC methods. It is designed for devices with limited computing power and/or memory, such as smartcards, mobile devices and PDAs. An important factor is the key strength, i.e. the difficulty in breaking the key and retrieving the plain text. In our design, first the user encrypts data to ensure the confidentiality, then, compute metadata over encrypted data. Later, the verifier can use remote data integrity checking protocol to verify the integrity. The verifier should able to detect any changes on data stored in cloud. The security of our scheme relies on the hardness of specific problems in Elliptic Curve Cryptography. Compared to existing schemes, our scheme has several advantages:

• It should detect all data corruption if anybody deletes or modifies the data in cloud storage, since we are using Sobol sequence instead of pseudorandom sequence for challenging the server for the integrity verification.

• Our scheme achieves the confidentiality of data

• It is efficient in terms of computation, storage, because its key size is low compared to RSA based solutions.

The rest of paper is organized as follows: In Section II we are introducing the concept of ECC, Sobol sequence and the necessity to adopt ECC and Sobol sequence to secure data (Integrity and Confidentiality) at CSP. Sections III introduce the system model: including cloud storage model, security threats, design goals, notations and permutations. In Section IV, we provide the detailed description of our scheme and We presented proposed scheme implementation and comparison with existing schemes in Section V. finally section VI gives the concluding remark of the whole paper.

II. ECC, SOBOL SEQUENCE

A. Elliptic curve cryptography (ECC)

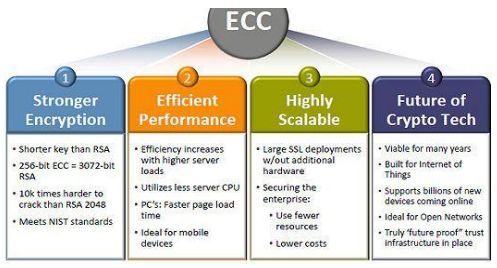

ECC
1
Stronger Encryption
• Shorter key than RSA
• 256-bit ECC = 3072-bit RSA
• 10k times harder to crack than RSA 2048
• Meets NIST standards
2
Efficient Performance
• Efficiency increases with higher server loads
• Utilizes less server CPU
• PC's: Faster page load time
• Ideal for mobile devices
3
Highly Scalable
• Large SSL deployments w/out additional hardware
• Securing the enterprise:
• Use fewer resources
• Lower costs
4
Future of Crypto Tech
• Viable for many years
• Built for Internet of Things
• Supports billions of new devices coming online
• Ideal for Open Networks
• Truly 'future proof" trust infrastructure in place

Fig. 1.Characteristics of ECC

It is an approach to public-key cryptography based on the algebraic structure of elliptic curves over finite fields. ECC requires smaller keys compared to non-ECC cryptography (based on plain Galois fields) to provide equivalent security. Elliptic curves are applicable for encryption, digital signatures, pseudo-random generators and other tasks. Its main characteristics are as shown in Fig. 1. Stronger encryption, efficient performance, highly scalable and it is future of cryptography technology. They are also used in several integer factorization algorithms that have applications in cryptography, such as Lenstra elliptic curve factorization. The primary benefit promised by ECC is a smaller key size, reducing storage and transmission requirements, as shown in fig. 2, i.e. that an elliptic curve group could provide the same level of security afforded by an RSA-based system with a large modulus and correspondingly larger key: for example, a 256-bit ECC public key should provide comparable security to a 3072-bit RSA public key.

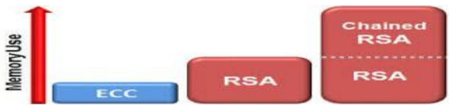


Fig. 2.Memory used by ECC, RSA

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940
Vol. 5, Issue 7, July 2016

For current cryptographic purposes, an elliptic curve is a plane curve over a finite field (rather than the real numbers) as shown in Fig. 3 which consists of the points satisfying the equation $y^2 = x^3 + ax + b$, along with a distinguished point at infinity, denoted ∞.

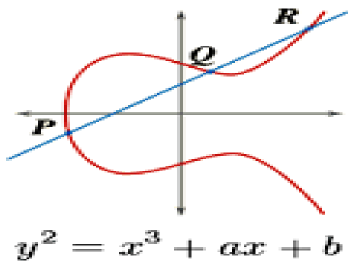
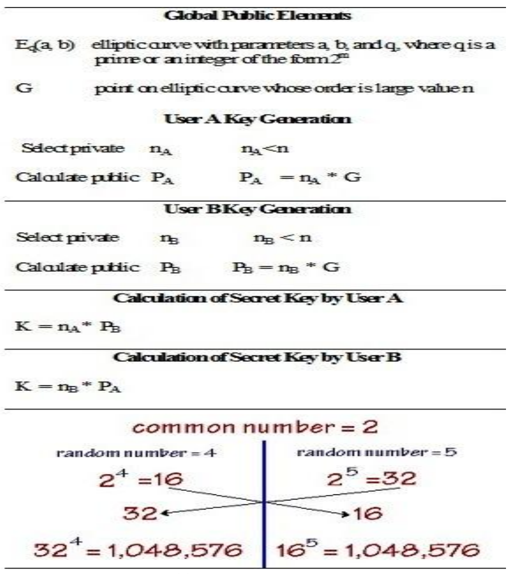


Fig. 3.Elliptic Curve

This set together with the group operation of elliptic curves is an Abelian group, with the point at infinity as identity element. The structure of the group is inherited from the divisor group of the underlying algebraic variety.
$\mathrm{Div}^0(E) \to \mathrm{Pic}^0(E) \simeq E,$
Cryptographic schemes: Several discrete logarithm-based protocols have been adapted to elliptic curves, replacing the group $(\mathbb{Z}_p)^\times$with an elliptic curve:

- The elliptic curve Diffie–Hellman (ECDH) key agreement scheme is based on the Diffie–Hellman scheme,
- The Elliptic Curve Integrated Encryption Scheme (ECIES), also known as Elliptic Curve Augmented Encryption Scheme or simply the Elliptic Curve Encryption Scheme,
- The Elliptic Curve Digital Signature Algorithm (ECDSA) is based on the Digital Signature Algorithm,
- The Edwards-curve Digital Signature Algorithm (EdDSA) is based on Schnorr signature and uses twisted Edwards curves,
- The ECMQV key agreement scheme is based on the MQV key agreement scheme,
- The ECQV implicit certificate scheme.

Diffie-Hellman: A cryptographic key exchange method developed by Whitfield Diffie and Martin Hellman in 1976. Also known as the "Diffie-Hellman-Merkle" method and "exponential key agreement," it enables parties at both ends to derive a shared, secret key without ever sending it to each other. Using a common number, both sides use a different random number as a power to raise the common number. The results are then sent to each other. The receiving party raises the received number to the same random power they used before, and the results are the same on both sides. Elliptic curve cryptography and key management shown in Fig. 4
Implementation: Some common implementation considerations include:
Domain parameters: To use ECC, all parties must agree on all the elements defining the elliptic curve, that is, the domain parameters of the scheme. The field is defined by



Global Public Elements

$E_q(a, b)$	elliptic curve with parameters a, b, and q, where q is a prime or an integer of the form 2^m	
G	point on elliptic curve whose order is large value n	
User A Key Generation		
Select private	n_A	$n_A < n$
Calculate public	P_A	$P_A = n_A * G$
User B Key Generation		
Select private	n_B	$n_B < n$
Calculate public	P_B	$P_B = n_B * G$
Calculation of Secret Key by User A		
$K = n_A * P_B$		
Calculation of Secret Key by User B		
$K = n_B * P_A$		

Fig. 4. ECC Diffie-Hellman Key Exchange

p in the prime case and the pair of m and f in the binary case. The elliptic curve is defined by the constants a and b used in its defining equation. Finally, the cyclic subgroup is defined by its generator (base point) G. For cryptographic application the order of G, that is the smallest positive number n such that$nG = \infty$, is normally prime. Since n is the size of a subgroup of $E(\mathbb{F}_p)$it follows from Lagrange's theorem that the number $h = \frac{1}{n}|E(\mathbb{F}_p)|$ is an integer. In cryptographic applications this number h, called the cofactor, must be small ($h \leq 4$) and preferably$h = 1$. To summarize: in the prime case, the domain parameters are(p, a, b, G, n, h); in the binary case, they are (m, f, a, b, G, n, h). Unless there is an assurance that domain parameters were generated by a party trusted with respect to their use, the domain parameters must be validated before use.
The generation of domain parameters is not usually done by each participant because this involves computing the number of points on a curve which is time-consuming and troublesome to implement. As a result, several standard bodies published domain parameters of elliptic curves for several common field sizes. Such domain parameters are commonly known as "standard curves" or "named curves"; a named curve can be referenced either by name or by the unique object identifier defined in the standard documents:
- NIST, Recommended Elliptic Curves for Government Use
- SECG, SEC 2: Recommended Elliptic Curve Domain Parameters
- ECC Brainpool (RFC 5639), ECC Brainpool Standard Curves and Curve Generation

Table 1, from NIST SP800-57 (Recommendation for Key Management), compares various algorithms by showing comparable key sizes in terms of computational effort for cryptanalysis. As can be seen, a considerably smaller key size can be used for ECC compared to RSA. Furthermore, for equal key lengths, the computational effort required for

IJARCCE
ISSN (Online) 2278-1021
ISSN (Print) 2319 5940
International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 5, Issue 7, July 2016

ECC and RSA is comparable. Thus, there is a computational advantage to using ECC with a shorter key length than a comparably secure RSA [18].

Table 1 Comparable Key Size in Terms of Computational Effort for Cryptanalysis (NIST SP-800-57)

Symmetric Key Algorithms	Diffie-Hellman, Digital Signature Algorithm	RSA (size of n in bits)	ECC (modulus size in bits)
80	L=1024, N=160	1024	160-223
112	L=2048, N=224	2048	224-255
128	L=3072, N=256	3072	256-383
192	L=7680, N=384	7680	384-511
256	L=15360, N=512	15,360	512+

L = size of public key, N = size of private key

B. Sobol sequence:

Sobol sequences (also called LP$_\tau$ sequences or (t, s) sequences in base 2) are an example of quasi-random low-discrepancy sequences. These sequences use a base of two to form successively finer uniform partitions of the unit interval and then reorder the coordinates in each dimension. This is because low discrepancy sequences tend to sample space "more uniformly" than random numbers. Algorithms that use such sequences may have superior convergence.

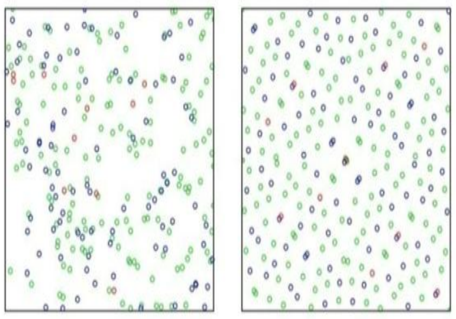


Fig. 5.Pseudorandom sequence VS. Sobol sequence

As shown in Fig. 5, 256 points from a pseudorandom number compared with the first 256 points from the 2, 3 Sobol sequence. The Sobol sequence covers the space more evenly. (Red=1... 10, blue=11... 100, green=101... 256) Implementation and availability of Sobol sequences: Good initialisation numbers for different numbers of dimensions are provided by several authors. For example, Sobol provides initialisation numbers for dimensions up to 51. The same set of initialisation numbers is used by Bratley and Fox. Initialisation numbers for high dimensions are available on Joe and Kuo. Peter Jäckel provides initialisation numbers up to dimension 32 in his book "Monte Carlo methods in finance". Other implementations are available as C, Fortran 77, or Fortran 90 routines in the Numerical Recipes collection of software. A free/open-source implementation in up to 1111 dimensions, based on the Joe and Kuo initialisation numbers, is available in C and Julia. A different free/open-source implementation is available for C++, Fortran 90, Matlab, and Python.

All computer-based random-number generation algorithms are "quasi-random" in that they are limited (ultimately by the number of bits of the operating system, but generally by algorithm assumptions) to have a period over which the random number sequence repeats. Hence no implemented random number generator is "truly" random. In this article, we will use Zemax' Sobol Sequence generator.

C. Related Work:

The security of remote storage applications has been increasingly addressed in the recent years, which has resulted in various approaches to the design of storage verification primitives. The literature distinguishes two main categories of verification schemes [10]: Deterministic verification schemes check the conservation of a remote data in a single, although potentially more expensive operation and probabilistic verification schemes rely on the random checking of portions of outsourced data. Wang et al. [14] discussed the problem of ensuring the availability and integrity of data storage in cloud computing. They utilized the homomorphic token and error correcting codes to achieve the integration of storage correctness insurance and data error localization, but their scheme does not support an efficient insert operation due to the index positions of data blocks. Existing schemes are unable to provide strong security assurance to the users because all these schemes are verifying integrity of data using pseudorandom sequence. It does not cover the whole data while computing integrity proof. Therefore, probabilistic verification schemes based on pseudorandom sequence does not give strong guarantee to the users about security of their data. To overcome this problem, Syam et al. [9] proposed a homomorpic distributed verification protocol to ensure data storage security in cloud computing using Sobol Sequence instead of pseudorandom sequence, which is more uniform than pseudorandom sequence. Their scheme achieves the availability and integrity of outsourced data in cloud but similar [14], it is also not addressing data confidentiality issue. To achieve all these security and performance requirements of cloud storage, we propose an efficient and secure protocol using ECC, Sobol sequence in section IV.

III.SYSTEM MODEL

Cloud Data Storage Model: The cloud storage model considering here is consists of three main components as illustrated in Fig. 6. (1) Cloud User: the user, who can be an individual or an organization originally storing their data in cloud and accessing the data. (2) Cloud Service Provider (CSP): the CSP, who manages cloud servers (CSs) and provides a paid storage space on its infrastructure to users as a service. (3) Third Party Auditor (TPA) or Verifier: the TPA or Verifier, who has expertise and capabilities that users may not have and verifies the integrity of outsourced data in cloud on behalf of users.

Based on the audit result, the TPA could release an audit report to user.

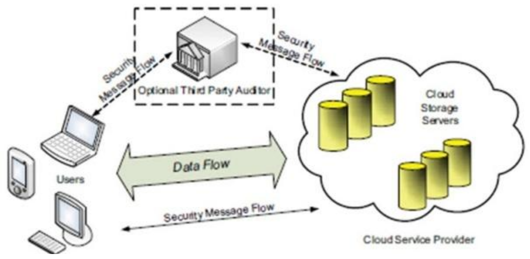


Fig.6. Cloud Data Storage Model

In cloud data storage model, the user stores his data in cloud through cloud service provider and if he wants to access the data back, sends a request to the CSP and receives the original data. If data is in encrypted form that can be decrypted using his secrete key. However, the data is stored in cloud is vulnerable to malicious attacks; it would bring irretrievable losses to the users, since their data is stored at an untrusted storage servers. It doesn't matter that whether data is encrypted or not before storing in cloud and no matter what trust relations the client and the server may have a priori share. The existing security mechanisms need to re-evaluate. Thus, it is always desirable to need an efficient and secure method for users to verify that whether data is intact? If user does not have the time, he assigns this task to third party auditor. The auditor verifies the integrity of data on behalf of users.

In this paper, we are considering two types of attacks for cloud data storage those are: Internal Attacks and External Attacks. Internal Attacks: These are initiated by malicious Cloud Service Provider (CSP) or malicious users. Those are intentionally corrupting the user's data inside the cloud by modifying or deleting. They are also able obtain all the information and may leaked it to outsiders. External Attacks: these are initiated by unauthorized parties from outside the cloud. The external attacker, who is capable of comprising cloud servers and can access the user's data as long as they are internally consistent i.e. he may delete or modify the customer's data and may leaked the user private information. We have designed an efficient and secure storage protocol to ensure the following goals. These goals are classified into two categories: Efficiency (Low computation overhead and less communication overhead) and Security Goals (Confidentiality, Integrity).

Notations and Permutations:

- F - the data file to be stored in cloud, the file F is divide into n blocks of equal length: $m_1, m_2, \ldots, m_n$, where $n=[|m|/l]$.
- $f_{key}(.)$- Sobol Random Function (SRF) indexed on some key, which is defined as f: $\{0, 1\}^* \times key\text{-}\{0, 1\} \log_2 n$.
- $\pi_{key}(.)$–Sobol Random Permutation (SRP) which is defined as $\pi : \{0,1\}^{\log 2(1)} \times key\text{–} \{0,1\}^{\log 2(1)}$.

Elliptic Curve Cryptography over ring Z_n: Let n be an integer and let a, b be two integers in Z_n such that gcd $(4a^3+27b^2, n) =1$. An elliptic curve E_n (a, b) over the ring Z_n is the set of points(x, y) $\in Z_n \times Z_n$ satisfying the equation: y^2+ax+b, together with the point at infinity denoted as O_n.

IV. PROPOSED SCHEME

To ensure the confidentiality and integrity of data stored in cloud, we propose an Efficient and Secure protocol. Our scheme is designed under the Elliptic Curve Cryptography [10, 12] construction and use of Sobol sequence to verify the integrity of storage data randomly. This protocol consists of three phases, namely Setup, Verification and Dynamic Data Operations and Verification. The three process model is depicted in fig.7. The construction of these phases is presented briefly as follows: Setup In this phase, the user pre-processes the file before storing in cloud. The Setup phase consists of three algorithms, those are: (1) KeyGen (2) Encryption (3) MetadataGen.

KeyGen:

In this algorithm, the user generates private key and public key pair using algorithm 1, it takes k as input and generates private key and public key pair as output as follows: the given security parameter k (k>512), user chooses two large primes p and q of size k such that p≡ q≡ 2 (mod 3). Then compute

$$n=pq \qquad (1)$$

and

$$N_n= \text{lcm}(p+1, q+1). \qquad (2)$$

Where N_n is a order of elliptic curve over the ring Z_n denoted by E_n (0, b), and b is a randomly chosen integer such that gcd(b, n)=1 and compute P is a generator of $E_n(0, b)$. It outputs public key PK= {b, n, p} and private key PR= $\{N_n)\}$.

Algorithm 1: KeyGen
1. Procedure: KeyGen(k) ⟵{ PK,PR}
2. Take security parameter k (k>512)
3. Choose two random primes p and q of size k: p≡q≡ 2 (mod 3)
4. Compute n=pq
5. Compute N_n = lcm(p+1, q+1)
6. Generate random integer b<n, gcd (b, n) =1
7. Compute P, is a generator of $E_n(0,b)$
8. Private key PR= { N_n }
9. Public key PK= {n, b, P}
10. end procedure

Encryption: To ensure the confidentiality of data, the user encrypts the each data block m_i in the file F using algorithm 2, it takes m_i keyed Sobol Ranodom Function (SRF) and secrete random parameter s as inputs and produce m'_i as output as follows:

$$F = \{m_1, m_2 \ldots m_n\} = \{m_i\}_{1 \le i \le n} \qquad (3)$$

$$F' = m'_i = m_i + f_k(s) \qquad (4)$$

where s is random of size l.

```
Algorithm 2: Encryption
1. Procedure: Encryption ($m_i$, s) $\leftarrow m'_i$
2. for 1 to n
3. Compute  $m_i' = m_i + f_k(s)$
4. end for
5. end procedure
```

Meta data Gen: After encrypting the data, the user computes a metadata over encrypted data to verify the integrity of data using algorithm 3, which takes m'_i, public key and private key as inputs and produce metadata T_i as output: $T_i \leftarrow m'_i P(\mathrm{mod}\ N_n))$ (5)

where $P\varepsilon E_n(0, b)$

```
Algorithm 3:MetadataGen
1. Procedure: MetadataGen($m'_i$ ,n, b, P) $\leftarrow T_i$
2. for 1 to n
3. Compute $T_i \leftarrow m'_i P(\mathrm{mod}\ N_n))$
4. end for
5. end procedure
```

After computation of metadata, the user sends metadata, public key to the TPA for later verification and sends file F' to cloud servers for storage. Verification Phase: Once data has stored in cloud, in order to ensure the integrity of data, our scheme entirely relies on verification phase. To verify the integrity of data, the verifier first creates a challenge and sends to the server. Upon receiving a challenge from the verifier, the server computes a response as integrity proof and return to the verifier. It consists of three algorithms: (1) Challenge (2) Proof Gen (3) Check Proof. Challenge: The verifier creates a challenge by running algorithm 4, it takes k_{SRF}, j and Q as input and return chal as output as follows: the verifier chooses a random keys k_{SRF} and k_{SRP} using Sobol sequence and computes random indices $1 \le i_j \le n$ (j= 1,…., c) of the set[1,n], where c = $\pi_{k_{SRP}}$ (c) (6)which prevents the server from anticipating which blocks will be queried in each challenge. The verifier also generates a fresh random value r to guarantee that the server does not reuse any values from the previous challenge and computes Q = rP. (7) Then, verifier creates the challenge chal = {k_{SRF}, j, Q}, and sends to the server.

```
Algorithm 4: Challenge
1. Procedure: Challenge($k_{SRF}$,j,Q) $\leftarrow$ chal
2. Generates a random keys $k_{SRF}$, $k_{SRP}$ and fresh
   random value using Sobol Sequence.
3. Compute c = $\boldsymbol{\pi_{k_{SRP}}}$ (c)
4. Compute Q=rP$\varepsilon$ $E_n(0, b)$
5. Create challenge chal={ $k_{SRF}$, j, Q}
6. end procedure
```

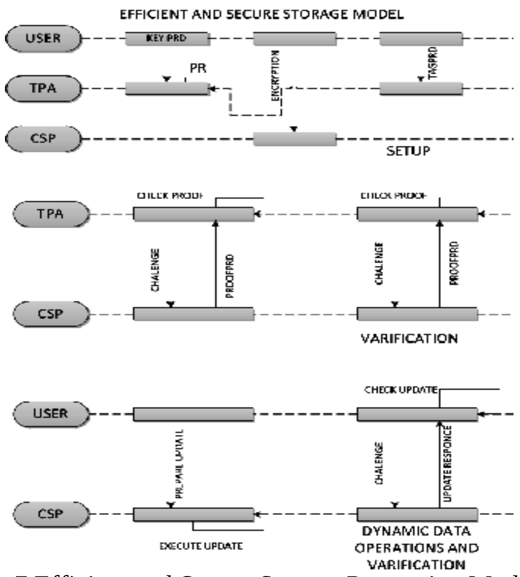


Fig. 7 Efficient and Secure Storage Processing Model.

Proof Gen: Upon receiving a challenge from the verifier, each server computes a response as integrity proof using algorithm 5, it takes encrypted data m'_i, challenge chal as inputs and produce response R as output as follows: first, it generates random numbers using Sobol random Function (SRF) i.e.

$$a_j = f_{k_{SRF}}(j) \quad (8)$$

Then compute $b = \sum_{j=1}^{c} a_j m'_{i_j}$ (9)

where $1 \le i_j \le n$

Later, computes a response R = bQ mod n (10)

$$= \sum_{j=1}^{c} a_j m'_{i_j} Q \bmod n$$
$$= \sum_{j=1}^{c} a_j m'_{i_j} rP \bmod n$$
$$= r\left(\sum_{j=1}^{c} a_j m'_{i_j} P \bmod n\right)$$

```
Algorithm 5: ProofGen
1. Procedure: ProofGen($m'_i$ , $k_{SRF}$, Q)$\leftarrow$R
2. Generates a n random numbers using $k_{SRF}$
3. for 1 to n
4. Generate $a_j = \mathbf{f_{k_{SRF}}}$ (j)
5. end for
6. compute   $b = \boldsymbol{\sum_{j=1}^{c} a_j m'_{i_j}}$
7. compute  R= bQ mod n
8. end procedure
```

Check Proof: After receiving a response from the server, the verifier checks the integrity using algorithm 6, it takes public key pk, challenge query chal, and proof R as inputs and return output as 1 if the integrity of file is verified as successfully or 0 as follows: the verifier re-generates random numbers using Sobol Random function i.e.

$a_j = f_{k_{SRF}}(j)$

then compute $S = \prod_{j=1}^{c} a_j T'_{i_j} \bmod n$ (11)

$R' = r\, S \bmod n$ (12)

Now, verifier checks whether R'=R, (13)

if response is valid, then it returns 1 otherwise 0.

Algorithm 6: CheckProof

```
1. Procedure: CheckProof($T'_i$ , r, $k_{SRF}$, n)$\leftarrow$R'
2. Generates a n random numbers using key $k_{SRF}$
3. for 1 to n
    4. Generate $a_j = \mathbf{f}_{\mathbf{k}_{\mathbf{SRF}}}$ ( j )
5. end for
6. compute  $S= \prod_{j=1}^{c} \mathbf{a}_j \mathbf{T'}_{\mathbf{i}\,\mathbf{j}}$ mod n
7. compute  R' = r S mod n
8. verify if (R'=R)
9. return true
10. else
11. return false
12. end if
13. end procedure
```

Dynamic Data Operations   The proposed scheme also supports dynamic data operations at block level [15] while maintaining same security assurance, such as Block Modification (BM), Block Insertion (BI) and Block Deletion (BD). These operations are performed by the server based on the user request in the general form (Block OP, j, m'_i), where Block Op indicates the block operation such as BM, BI and BD. The parameter j indicates the particular block to be updated and m^*_i is the new block. In order to update data in cloud, the user creates a request and sends to the server. Upon receiving an update request from the user, the server performs the particular update operation (modification/insert/delete). Here, we show that how our scheme supports dynamic data operations efficiently:

Algorithm 7: PrepareUpdate

```
1. Procedure:PrepareUpdate$\leftarrow$(BM/BI/BD,j, $m'_i$)
2. Select a update block $m_j$
3. if (update==modification/insert)
    4. Encrypt $m'_j \leftarrow m_j + f_k(s)$
    5. Compute Tj $\leftarrow$ $m'_j$ P mod $N_n$
    6. Update= (BM/BI, j, $m'_i$)
7. else if(update==deletion)
    8. Update = ((BD, j)
9. Send update request to the server
10. end if
11. end procedure
```

Block Modification (BM): Data modification is one of the frequently used operations in cloud data storage. Suppose, the user wants to modify the block m_j with m'_i, then the user runs the algorithm 7 to do the following:

1) Create a new block m_j
2) Encrypt the new block using equation (2)

$$m'_j \leftarrow m_j + f_k(s) \tag{14}$$

3) Compute new metadata using equation

$$T_j \leftarrow m'_j P \bmod N_n \tag{15}$$

4) Create update request (BM, j, m_i) and sends to the server.
5) The Metadata sends to TPA for later verification

Upon receiving an update request, the server replace the block m'_i with m'_j and construct update version of the file F" by running algorithm 8.

Algorithm 8: ExecuteUpdate

```
1. Procedure: ExecuteUpdate$\leftarrow$ {F"}
2. if(update==modification)
    3. replace $m_i$ with m'j  in the file F'
    4. update file F"
5. else if(update==insert)
    6. insert $m^*_x$ before $m_i$ or append
7. else if(update==deletion)
    8. delete $m_i$ from file F'
    9. update the file F"
    10. move all blocks backward after $i^{th}$ block
11. end if
12. end procedure
```

Block Insertion (BI): In this operation, the user wants to insert a new block m* after position j in the file F'= {m'_1 ,…, m'_n}. The block insertion operation changes the logical structure of the file; the proposed scheme can perform the block insertion operation without re-computing metadata of all blocks that have been shifted after inserting a block, because block index is not included in the metadata. To perform an insertion of a new block m* after position j in a file, the user runs algorithm 7 to do the following:
1. Create a new block m^*_j
2. Encrypt the new block

$$m'_j \leftarrow m^*_j + f_k(s) \tag{16}$$

3. Compute new metadata

$$T^*_j \leftarrow m'_j P \bmod N_n \tag{17}$$

4. Create update request (BI, j, m'_i) and sends to the server.
5. The Metadata sends to TPA for later verification

Upon receiving the update request, the server replace the block m'_j with m'_j and construct update version of the file F" by run the algorithm 8.

Block Deletion (BD): The Block deletion operation is the opposite of insertion operation. When one block is deleted, all subsequent blocks are moved one step forward. Suppose, the user wants to delete a specific data block at position j from the file F', creates a delete request (BD, j), sends to the server and also sends request to the TPA to delete corresponding block metadata. Upon receiving a delete request from the user, the server deletes the block m'j from the file and constructs update version of the file F". Similarly, the TPA deletes corresponding metadata.

IJARCCE
ISSN (Online) 2278-1021
ISSN (Print) 2319 5940
International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 5, Issue 7, July 2016

Here, deletion of metadata do not depends on other block metadata. The detail of delete operation is given in algorithm 8.

Verification: To ensure the security of dynamic data operations, the user verifies the integrity of updated block immediately after updating as follows:
(1) The user challenges the server immediately for the proof of update operation i.e. $Q = rP$ (18)

(2) Upon receiving a request from the user, the server computes a response for updated block and returns to the user: $R_j \leftarrow m'_j P \bmod n$ (19)

(3) After receiving an update response from the server, the user verifies whether response is matched with metadata of particular block by running algorithm 9, if it returns true, server has been updated data successfully otherwise not.

```
Algorithm 9: VerifyUpdate
 1. Procedure: VerifyUpdate(pk, Q, R')→{1,0 }
 2.    if(update==modification/insert)
 3.        if($T_j$=$R_j$)
 4.            return 1
 5.            else
 6.            return 0
 7.            end if
 8.    else if(update==deletion)
 9.     verification directly starts from static case
10.    end if
11. end procedure
```

V. PROPOSED SCHEME IMPLEMENTATION

In this section, we present the Performance analysis, experimental results and Security analysis of our protocol as well as Comparison with Existing scheme.

A. Performance Analysis:

We analyze the performance of our scheme in terms of storage, communication and computation complexity.
Storage cost: Here, we detail the storage cost required by the client, TPA and server. User Side: The user needs to store the only secrete parameter. The storage cost for that is O (1). Server Side: the server needs to be store the complete file, the cost for storage file is O (n) bits. TPA or Verifier: the verifier needs to store metadata and public key. The metadata is a relatively smaller than original file, so storage cost for metadata is O (1). Communication Cost: Here, we consider the communication cost between the server and verifier during verification phase. The challenge sent by the verifier to the server, which consists of O (1) and the response (it is a small size compare to original file) sent by server to the verifier, which consists of O (1). Thus, total communication cost is O (1).

Computation Cost: We analyze the computation cost of the user, verifier and server as follows: User: during the setup phase, the user generates a private key and public key whose cost is O (1). Then, to encrypt a file, the user needs to perform integer addition and its cost is O (n). Finally, computes the metadata by performing n-bit point multiplications whose cost is O (1). Hence, total computation cost of the user is: O (1).

Verifier: During the verification phase, the TPA or verifier needs to generate three random numbers $\langle k_{SRF}, j, r\rangle$, then compute $c = \pi_{k_{SRP}}(c)$ and $Q = rP$, whose cost is O(1). Again, after receiving the response, the verifier re-generates $\{a_j\}$ j= [l, c], the computation cost of each $a_j m'_{i_j}$ corresponds to the sum of point multiplication of two bits. Finally, the verifier computes R', the cost of R' is a two point multiplications plus sum of 2 bit integer plus generating random numbers cost, which is O(1) respectively. Hence, the total computation cost at verifier side is O(1).

Server Side: During the verification phase, the server needs to generate n-Sobolrandom b-bit integers a_i , then it computes $b = \sum_{j=1}^{c} a_j m'_{i_j}$ and $R = r\sum_{j=1}^{c} a_j m'_{i_j} P \bmod n$, The computation of each $a_j m'_{i_j}$ corresponds to the sum of point multiplication of two bits. The computation cost of $a_j m'_{i_j}$ is O(1). Next, the server computes a proof, which consists of point multiplications in ProofGen algorithm, its cost is O(1). The total computation cost of server for generating integrity proof (response) is O(1). In table 2, we summarized the storage, communication and computation costs.

Table 2: Summary of Storage, Communication and Computation cost of Proposed Protocol

Storage Cost			Communication Cost		Computation Cost	
Verifier	Server	Verifier	Server	User	Verifier	Server
O(1)	O(n)	O(1)	O(1)	O(1)	O(1)	O(1)

B. Experimental Result

All experiments conducted using C++ on system with dual core 2-GHZ processor and 4GB RAM running Windows 2007. In our implementation, we use MIRACL library version 5.4.2 to achieve better security work on elliptic curve with 160-bit group order instead of RSA on 1024 bits. Here, we are measuring total time for computation cost of the verifier and server using ECC and RSA respectively.

$$\text{Speedup} = \frac{\text{RSA} - \text{ECC}}{\text{RSA}} * 100$$

Then, we compare computation cost of our protocol with RSA-based remote data checking protocols, which includes the verifier, server and user computation costs and presented results in table 3,4 & 5.

IJARCCE
ISSN (Online) 2278-1021
ISSN (Print) 2319 5940
International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 5, Issue 7, July 2016

Table 3: Computation Cost at Verifier using RSA [16] and ECC based schemes.

File Size	Verifier side using RSA[33]	Verifier Side using ECC	Speedup
10MB	424.37 ms	316.26ms	25%
20MB	482.81ms	342.43ms	29%
30MB	561.62ms	376.03ms	32%
40MB	641.46ms	415.09ms	35%
50MB	743.64ms	465.13ms	38%

Table 3 shows that the total computation cost of verifier for our proposed scheme is faster than existing RSA based scheme [15]

Table 4: Computation Cost at Server with RSA based scheme and ECC scheme

l(bits)	Server Side with RSA[33]	Server Side with ECC	Speedup (%)
10MB	388.01 ms	275.11 ms	29%
20MB	447.62 ms	312.43 ms	30%
30MB	508.39 ms	348.21 ms	31%
40MB	562.67 ms	381.21 ms	32%
50MB	625.16 ms	418.76 ms	33%

Table 4 shows that the total computation cost of the server for proposed scheme is faster than existing RSA based scheme [15].

Table 5: Metadata Computation Cost at user with RSA and ECC based schemes

l(bits)	Server Side with RSA[33]	Server Side with ECC	Speedup (%)
10MB	244.11 ms	183.06 ms	25%
20MB	296.41 ms	218.32 ms	26%
30MB	352.53 ms	253.38 ms	28%
40MB	403.17 ms	289.63 ms	29%
50MB	467.26 ms	323.92 ms	30%

Table 5 shows that the total computation cost of metadata at user side in our scheme is faster than existing RSA based scheme [15]

C. Security Analysis:

In this section, we present the formal security analysis of the proposed scheme. That means integrity and confidentiality of data stored in cloud.
In our integrity analysis, we have depended on the Finding order of elliptic curve and Elliptic curve discrete logarithm problem denoted by ELDL problems.

(1) Finding the order of elliptic curves: The order of elliptic curve over the ring Z_n is: let n=pq is defined in [38,] as $N_n = \text{lcm}(\#E_p(a, b), \#E_q(a, b))$. N_n is the order of the curve, i.e. for any $P\varepsilon E_n(a, b)$ and any integer k, such that $(k N_n+1) P=P$. (20)
If ($a=0$ and $p\equiv q\equiv 2 \bmod 3$) or ($b=0$ and $p\equiv q\equiv 3 \bmod 4$), the order of $E_n(a, b)$ is equal to N_n. The given $N_n = \text{lcm}(\#E_p(a, b), \#E_q(a, b)) = \text{lcm}(p+1, q+1)$ (21)

Solving N_n is computationally equitant to factoring the corresponding number n.

(2) Elliptic Curve Discrete Logarithm Problem (ECDLP) Consider the equation Q=rp where Q, $P\varepsilon E_n(a, b)$ and $r<n$. it is relatively hard to determine r given Q and P.
Theorem1. The proposed protocol is complete
Proof: Here, we are proving this theorem according to the definition of sound and commutative property of point multiplication in an elliptic curve [10].

We have R' = R

$R' = rS \bmod n$

$S = \prod_{j=1}^{c} a_j T_{i_j} \bmod n$ where $a_j = f_k(j)$

$= \prod_{j=1}^{c} (a_j m'_{i_j} P \bmod N_n) \bmod n$

$= \sum_{j=1}^{c} a_j m'_{i_j} P \bmod n$

$R' = r S \bmod n$

$= r(\prod_{j=1}^{c} (a_j m'_{i_j} P \bmod n)$

$= r(\sum_{j=1}^{c} a_j m'_{i_j} P \bmod n)$

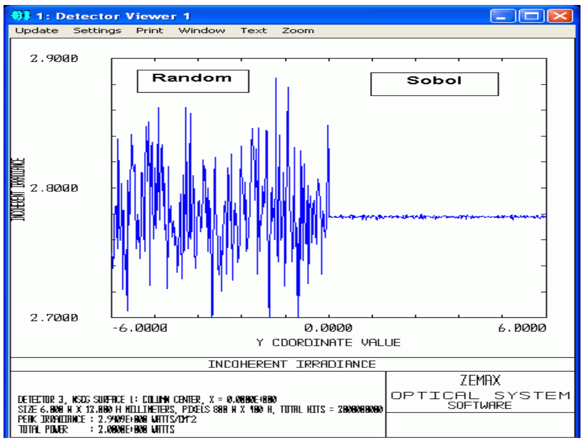
$= R$

From the equation (13), the protocol is complete or valid. Then the verifier is "probabilistically" assured that server still holds data safely. In reality, verifier only verifies that server holds the j [1, c] selective blocks where j is chosen randomly.

Monte Carlo Results

Monte Carlo methods (or Monte Carlo experiments) are a broad class of computational algorithms that rely on repeated random sampling to obtain numerical results. They are often used in physical and mathematical problems and are most useful when it is difficult or impossible to use other mathematical methods. Monte Carlo methods are mainly used in three distinct problem classes: optimization, numerical integration, and generating draws from a probability distribution. Zemax has the capability to do tolerance in different modes: sensitivity mode, inverse sensitivity and inverse increment.

It can also perform a Monte Carlo simulation. After the sensitivity analysis or inverse sensitivity analysis is performed, Zemax will perform a Monte Carlo analysis. When there are very few rays (10,000) and many many rays (1 billion) the results of the random ray-trace and Sobol ray-trace are similar, Hence Sobol sampling is most useful. The signal to noise ratio (SNR) of the random ray-trace is SQRT (N) where N is the average number of rays hitting a pixel. For the Sobol sampling scheme, the SNR goes linearly as N. This can be seen by taking a cross-section through the distribution with 1 billion rays per source:



```
DETECTOR 3, NSCG SURFACE 1: COLUMN CENTER, X = 0.0000E+000
SIZE 6.000 W X 12.000 H MILLIMETERS, PIXELS 800 W X 100 H, TOTAL HITS = 200000000
PEAK IRRADIANCE : 2.9409E+000 WATTS/CM^2
TOTAL POWER     : 2.0000E+000 WATTS
```

Fig.8. simulation using random numbers

Fig.8 shows that the random sequence gives worst performance, while Sobol Sequence gives rapid convergence to the solution. To conclude that it has been shown that Sobol sequence can evaluate more efficiently than pseudorandom sequences.

Sobol sampling can make a significant reduction in the time taken to undertake a Monte Carlo ray-tracing simulation, and will generally produce faster convergence than truly random rays. However, in any Monte-Carlo simulation, there is ultimately no more accurate method that making many samples with a truly random number generator. For this reason, Zemax allows you to select either a Sobol sampling scheme, or to use Zemax's long-period random number generator, in the source tab of the object properties.

Confidentiality: Now, we analyze the confidentiality of our scheme: The stored data in cloud cannot be leaked to a malicious attacker (servers and TPA).

In this analysis, we depend on the hardness of the Elliptive Curve Diffie- Hellman (ECHP) and Elliptive Curve Discrete Logarithm (ECDL) problems.

Theorem 2: The proposed protocol is confidential against data leakage to attacker. We prove this theorem under different attacks:

(1) The secret parameter s cannot be derived by a malicious user eavesdropping on the communication link between the user and server because of Elliptive Curve Diffie-Hellman (ECDH) problem. The public parameter {b,n,P} cannot help the adversary to infer or calculate any useful information that can reveal the shared key between the user and server.

(2) Suppose, If the malicious server wants to access the data from the encrypted file F'=mi'. But it is not possible, because in order to access the encrypted data, he should need a secrete parameter, this secrete key chosen by user randomly. If server tries to get the secret key by using different combinations of public parameters but fail to do so due to the ECDL problem. Hence, the server cannot learn anything from F'.

(3) The TPA has Ti $\leftarrow$ m'_i P (mod N_n). If he tries to access data content from metadata, the user computes metadata over encrypted the data using secrete key. However, it is not possible because the secrete parameter chosen by the user from random. So there is no chance to TPA get secrete parameter using public key and metadata. Hence, The TPA cannot learn anything from metadata T_i.

Therefore on the basis of ECDH and ECDL problems, our protocol is confidential against data leakage.

D. Comparison with Existing Schemes

Table 6: Comparisons between Proposed Protocol and selective Existing Protocols

	Q.wang [14]	Hao [17]	Syam[9]	Proposed protocol
Type of Guaranty	Prob	Deter	Prob	Prob
Integrity	Partial	Yes	Yes	Yes
Confidentiality	No	Partial	No	Yes
Public Verifiability	Yes	Yes	No	Yes
Data Dynamics	Yes	Yes	Partial	O(1)
Communication complexity	O(logn)	O(1)	O(1)	O(1)
Server Computation	O(logn)	O(n)	O(clogn)	O(1)
Verifier computation	O(logn)	O(n)	O(clogn)	O(1)
Probability Detection	O($N^{-1/2}$)	O($N^{-1/2}$)	O(N^{-1})	O(N^{-1})

Prob: Probabilistic Deter: Deterministic We proposed an ECC based verification scheme. The principal of ECC compared to RSA is that it appear to offer equal security for a far smaller key size, thereby it reduced the computation overhead. pseudorandom sequence is not uniform (uncorrelated random numbers), and it will take more time to detect data corruption, so its time consuming whereas proposed protocol verifies the integrity of the data using Sobol sequence. Our scheme should detect all data corruptions with less number of blocks since sobol sequence covers the entire data in the file more uniformly than pseudorandom sequence. Finally, the proposed protocol is private against unauthorized data leakage because, we are encrypting the data before storing in cloud. In Table 6, we summarize the comparison between the selective existing protocols and proposed protocol.

VI. CONCLUSION

Elliptic curves cryptography (ECC) is one of the public-key cryptographic algorithms. Though RSA is the most commonly applicable cryptosystem scheme nowadays for the web security, ECC may overtake it due to the proliferation of smaller devices and increasing security needs. Although, several attempts had been made at providing a secured environment for activities in the Cloud, Elliptic Curve Cryptography (ECC) provides solutions for a secured Cloud environment with improved performance in computing power and battery resource usage. This makes it attractive for mobile applications. ECC provided a robust and secured model for the development and deployment of secured application in the Cloud. In this paper, we have studied the problem of Integrity and Confidentiality of data storage in cloud computing and proposed an efficient and secure protocol using ECC and Sobol sequence. The proposed method is mainly suitable for thin users who have less resources and limited computing capability. It satisfies the all security and performance requirements of cloud data storage. Our method also supports public verifiability that enables TPA to verify the integrity of data without retrieving original data from the server and probability detects data corruptions. Moreover, our scheme also supports dynamic data operations, which performed by the user on data stored in cloud while maintaining same security assurance. We have proved that proposed scheme is secure in terms of integrity and confidentiality through security analysis. Through, performance analysis and experimental results proved that proposed scheme is efficient. Compared with previously proposed protocols, we have also proved that proposed scheme is more secure and efficient. The research may further expanded into the comparison of ECC with the quantum cryptography (assuming the experimental tools are easily accessible).

REFERENCES

[1] Amazon.com, “Amazon Web Services (AWS),” Online at http://aws. amazon.com, 2008.
[2] N. Gohring, “Amazon’s S3 down for several hours” Online at http://www.pcworld.com/businesscenter/article/142549/amazons s3 down for several hours.html, 2008.
[3] Apple “ICloud” Online at http://www.apple.com/icloud/what-is.html 2010 .
[4] T Mather, S Kumaraswamy, and S Latif “Cloud Security and Privacy”, O’REILLY Publication, first edition, sep- 2009.
[5] H. Takabi, J.B.D. Joshi, and G. Ahn, “Security and Privacy Challenges in Cloud Computing Environments”, Article in IEEE Security and Privacy, vol. 8, no.6, Nov- Dec. 10, pp. 24-31, 2010.
[6] Y. Deswarte, J.-J. Quisquater, and A. Saidane. “Remote integrity checking”. In Proc. of Conference on Integrity and Internal Control in Information Systems (IICIS’03), Nov03, lausanne, Switzerland, 2003.
[7] G. Ateniese, R. D. Pietro, L. V. Mancini, and G. Tsudik, “Scalable and Efficient Provable Data Possession,” Proc. of SecureComm ’08, pp. 1–10, Istanbul, Turkey, 2008.
[8] M. A. Shah, M. Baker, J. C. Mogul, and R. Swaminathan, “Auditing to Keep Online Storage Services Honest,” Proc. 11th USENIX Workshop on Hot Topics in Operating Systems (HOTOS ’07), pp. 1–6, CA, USA, 2007.
[9] P. Syam Kumar, R. Subramanian, “Homomorpic Distributed Verification Ptorotocol for Ensuring Data Storage in Cloud Computing”. International Journal of Information, VOL. 14, NO.10, OCT-11, pp.3465-3476, 2011.
[10] N. Oualha, M. Onen, Y. Roudier, “A Security Protocol for Self-Organizing Data Storage”. Tech. Rep. EURECOM+2399, Institut Eurecom, France, 2008.
[11] V. Miller, “Uses of elliptic curves in cryptography”, advances in Cryptology, Proceedings of Crypto’85, Lecture Notes in Computer Science, 218 Springer-Verlag, pp.417-426. 1986.
[12] K. Koyama, U. Maurer, T. Okamoto, and S. Vanstone, “New Public-Key Schemes Based on Elliptic Curves over the Ring Zn”, Advances in Cryptology - CRYPTO '91, Lecture Notes in Computer Science, Springer-Verlag, vol. 576, Aug 91 pp. 252-266, 1991.
[13] Brately P and Fox B L “Algorithm 659: Implementing Sobol’s Quasi-random Sequence Generator” ACM Trans. Math. Software 14 (1), pp. 88–100, 1988.
[14] C. Wang, Q. Wang, K. Ren, N. cao and W. Lou , “Towards Secure and Dependable Storage Services in Cloud Computing”, Accepted for publication in future issue of IEEE Trans. Service Computing. DOI:10.1109/TSC.2011.24.
[15] Z. Hao, S. Zhong, and N. Yu, “A Privacy-Preserving Remote Data Integrity Checking Protocol with Data Dynamics and Public Verifiability”, Accepted for publication in future issue of IEEE Trans. Knowledge and Data Engineering, DOI: 10.1109/TKDE.2011.62
[16] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson,and D. Song, “Remote Data Checking using Provable Data Possession,” ACM Trans. ACM Transactions on Information and System Security, Vol. 14, No. 1, Article 12, may, pp. 12.1–12.34, 2011.
[17] Z. Hao and N. Yu, “A multiple-replica remote data possession checking protocol with public verifiability,” in Second International Symposium on Data, Privacy, and E- Commerce ,.Buffalo, Niagara Falls, 2010.
[18] William Stallings, Cryptography and Network Security Principles and Practice Sixth Edition, Pearsons, ISBN10:0-13-335469-5.

BIOGRAPHY

Santosh Kumar Singh is a Research Scholar in the Department of Computer Applications, Vinoba Bhave University, Hazaribag, Jharkhand, India. He received M. Phil (Computer Science) degree in 2011 and Master of Philosophy Dissertation entitled “study on the network security & network topology”. He Qualified Doctoral (Ph.D) Eligibility Test 2014 of Vinoba Bhave University, Hazaribag. His research interests are Cloud Computing, Parallel and Distributed Computing etc. He is currently working on Cloud Computing.