IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661, p- ISSN: 2278-8727Volume 11, Issue 1 (May. - Jun. 2013), PP 39-45
www.iosrjournals.org

Public Verifiability in Cloud Computing Using Signcryption Based on Elliptic Curves

Jahnvi S. Kapadia[1], Prof. Mehul P. Barot[2]
[1](Computer engineering, LDRP ITR, Gandhinagar/GTU university, India)
[2](Computer engineering, LDRP ITR, Gandhinagar/GTU university, India)

Abstract : *Cloud computing is a computing paradigm that involves outsourcing of computing resources with the capabilities of expendable resource scalability, on-demand provisioning with little or no up-front IT infrastructure investment costs. It has recently emerged as a promising hosting platform that performs an intelligent usage of a collection of services, applications, information and infrastructure comprised of pools of computer, network, information and storage resources. However along with these advantages, storing a large amount of data including critical information on the cloud motivates highly skilled hackers thus creating a need for the security to be considered as one of the top issues while considering Cloud Computing. In the cloud storage model, data is stored on multiple virtualized servers. Physically the resources will span multiple servers and can even span storage sites. We have proposed an effective scheme to ensure the correctness of user's data on cloud data storage and public verifiability without demanding user's time, feasibility or resources. Whenever data corruption is detected during the storage correctness verification, this scheme can almost guarantee the simultaneous localization of data errors and the identification of the misbehaving server(s).*

Keywords - *Elliptic Curves, Homomorphic, Signcryption, Unsigncryption.*

I. INTRODUCTION

Let's say you're an executive at a large corporation. Your particular responsibilities include making sure that all of your employees have the right hardware and software they need to do their jobs. Buying computers for everyone isn't enough; you also have to purchase software or software licenses to give employees the tools they require. Whenever you have a new hire, you have to buy more software or make sure your current software license allows another user. It's a stressful work.

Soon, there may be an alternative for executives like you. Instead of installing a suite of software for each computer, you'd only have to load one application. That application would allow workers to log into a Web-based service which hosts all the programs the user would need for his or her job. Remote machines owned by another company would run everything from e-mail to word processing to complex data analysis programs. It's called cloud computing, and it could change the entire computer industry.

Cloud computing is a computing paradigm that involves outsourcing of computing resources with the capabilities of expendable resource scalability, on-demand provisioning with little or no up-front IT infrastructure investment costs[1]. It has recently emerged as a promising hosting platform that performs an intelligent usage of a collection of services, applications, information and infrastructure comprised of pools of computer, network, information and storage resources. 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) [web-ensuring] 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.

However along with this advantage of storing a large amount of data including critical information on the cloud motivates highly skilled hackers and creates a need for the security to be considered as one of the top issues while considering Cloud Computing. Data security for such a cloud service encompasses several aspects including secure channels, access controls, and encryption. And, when we consider the security of data in a cloud, we must consider the security triad: confidentiality, integrity, and availability [2]. In the cloud storage model, data is stored on multiple virtualized servers. Physically the resources will span multiple servers and can even span storage sites. Thus an effective scheme to ensure the correctness of user's data on cloud must be utilized.

The scheme proposed here relies on erasure-correcting code in the file distribution preparation to provide redundancies and guarantee the data dependability. By utilizing the homomorphic token with distributed verification of erasure-coded data, this scheme achieves the storage correctness insurance as well as data error localization. Whenever data corruption has been detected during the storage correctness verification, this scheme can almost guarantee the simultaneous localization of data errors and the identification of the misbehaving server(s).The key feature of this scheme is that it uses signcryption/unsigncryption schemes based on elliptic curves to enforce public verifiability which is an enhancement to a previously described cloud system model in [3]

II. PROPOSED SYSTEM

1. SYSTEM'S COMPONENTS

Representative network architecture for cloud data storage for the system we have proposed is illustrated in figure 1.

Four different network entities of this system can be identified as follows:

- **User:** users, who have data to be stored in the cloud and rely on the cloud for data computation, consist of both individual consumers and organizations.
- **Cloud Service Provider (CSP):** a CSP, who has significant resources and expertise in building and managing distributed cloud storage servers, owns and operates live Cloud Computing systems.
- **Authentication Server (AS):** an authentication server, who has expertise and capabilities that users may not have, is trusted to assess and expose risk of cloud storage services on behalf of the users upon request.
- **Certificate Authority (CA):** a certification authority provides certificates for authentication and identification to User and AS.

In cloud data storage, a user stores his data through a CSP into a set of cloud servers, which are running in a simultaneous, cooperated and distributed manner. Data redundancy can be employed with technique of erasure-correcting code to further tolerate faults or server crash as user's data grows in size and importance. Thereafter, for application purposes, the user interacts with the cloud servers via CSP to access or retrieve his data. In some cases, the user may need to perform block level operations on his data. The most general forms of these operations considered are block update, delete, insert and append.

As users no longer possess their data locally, it is of critical importance to assure users that their data are being correctly stored and maintained. That is, users should be equipped with security means so that they can make continuous correctness assurance of their stored data even without the existence of local copies. As the users do not necessarily have the time, feasibility or resources to monitor their data, this task is delegated the Authentication Server which uses signcryption scheme based on Elliptic Curves has been used for this purpose. One of the key issues is to effectively detect any unauthorized data modification and corruption, possibly due to server compromise and/or random Byzantine failures [3]. Besides, in the distributed case when such inconsistencies are successfully detected, to find which server the data error lies in is also of great significance, since it can be the first step to fast recover the storage errors. To address these problems, the main scheme for ensuring cloud data storage is presented in next section.

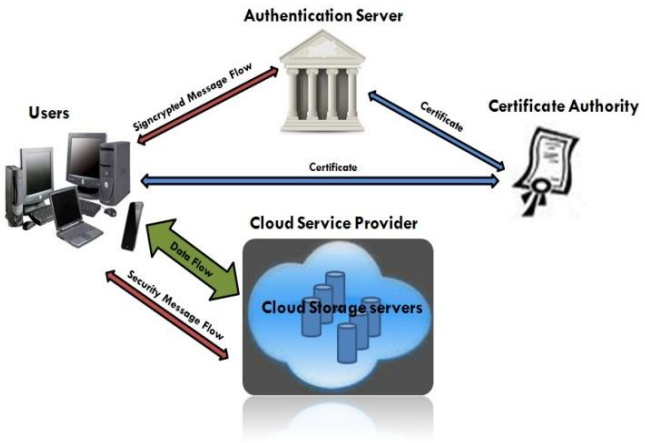

Fig 1: proposed system model

1.1 NOTATION AND PRELIMINARIES

Domain parameters of the proposed scheme consist of a suitably selected elliptic curve *E* defined over a finite field *Fq* with the Weierstrass equation of the form $y^2 = x^3 + ax + b$ and a *base point* $G \in E(Fq)$ in which *q* is a large prime number. In order to make the elliptic curve non-singular, $a,b \in Fq$ should satisfy $4a^3 + 27b^2 \neq 0 \pmod q$. To guard against *small subgroup attacks*, the point *G* should be of a prime order *n* or equivalently, $nG = O$ where *O* denotes the point of elliptic curve at infinity, and we should have $n > 4\sqrt{q}$. To protect against other known attacks on special classes of elliptic curves, *n* should not divide $q^i - 1$ for all $1 \leq i \leq V$ ($V = 20$ suffices in practice), $n \neq q$ should be satisfied, and the curve should be non-super singular [4].To retain the intractability of ECDLP, *n* should at least satisfy $n > 2^{160}$ for the common applications.

W_U**:** A randomly select integer which is a Private Key of User ($W_U \in_R [1, n-1]$)

$\mathbf{W_U}$**:** Public Key of User calculates as $W_U = w_U G$

$\mathbf{ID_U}$**:** A unique User identifier.

W_A**:** A randomly select integer which is a Private Key of AS ($W_A \in_R [1, n-1]$).

$\mathbf{W_A}$**:** Public Key of User calculates as $W_A = w_A G$

$\mathbf{ID_A}$**:** A unique AS identifier.

$Cert_U$**:** Digital certificate for public key of user from CA

$Cert_A$**:** Digital certificate for public key of AS from CA

If CA is not involved in the public key generation, it is necessary for CA to verify that each entity really possesses the corresponding private key of its claimed public key. This can be accomplished by a zero-knowledge technique. It should also be verified that the public keys belong to the main group.

F: The data file to be stored. It is assumed that F can be denoted as a matrix of m equal-sized data vectors, each consisting of l blocks. Data blocks are all well represented as elements in Galois Field $GF(2p)$ for $p = 8 \text{ or } 16$.

A: The dispersal matrix used for Reed-Solomon coding.

G: The encoded file matrix, which includes a set of $n = m + k$ vectors, each consisting of l blocks.

$\boldsymbol{fkey}(\cdot)$: Pseudorandom function (PRF), which is defined as $f : \{0,1\}^* \times key \rightarrow GF(2p)$.

$\boldsymbol{\Phi key}(\cdot)$: Pseudorandom permutation (PRP), which is defined as $\Phi: \{0,1\}^{log2(l)} \times key \rightarrow \{0,1\}^{log2(l)}$.

$\boldsymbol{Ver}$**:** A version number bound with the index for individual blocks, which records the times the block has been modified. Initially it is assumed that ver is 0 for all data blocks.

$\boldsymbol{S\ ij\ ver}$: The seed for PRF, which depends on the file name, block index$i$, the server position j as well as the optional block version numberver.

1.2 PHASES AND FLOW OF THE PROPOSED SCHEME

Different phases of the scheme where different techniques applied and the flow of data distribution, encryption and authentication is listed as follows:

Phase 1: file distribution preparation.
- Done by user
- Using erasure-correcting code

Phase 2: Token Pre-computation.
- Done by User
- Homomorpic tokens are generated using token pre-computation algorithm

Phase 3: Signcryption of pre-computed Tokens.
- Done by User
- Using Signcrption Scheme based on elliptic curves

Phase 4: Unsigncryption of Tokens.
- Done by AS
- Using Unsigncryption Scheme

Phase 5: Correctness Verification and Error Localization.
- Using Challenge Token Pre-computation algorithm

Phase 6: File Retrieval and Error Recovery.
- Using Error Recovery algorithm

III. FILE DISTRIBUTION PREPARATION

It is well known that erasure-correcting code may be used to tolerate multiple failures in distributed storage systems. In cloud data storage, we relay on this technique to disperse the data file *F* redundantly across a set of $n = m + k$ distributed servers. A $(m + k, k)$ Reed-Solomon erasure-correcting code is used to create k

redundancy parity vectors from m data vectors in such a way that the original m data vectors can be reconstructed from any m out of the m + k data and parity vectors. By placing each of the m + k vectors on a different server, the original data file can survive the failure of any k of the $m+k$ servers without any data loss, with a space overhead of k/m. For support of efficient sequential I/O to the original file, our file layout is systematic, i.e., the unmodified m data file vectors together with k parity vectors are distributed across $m+k$ different servers [3].

Let $\boldsymbol{F} = (F1, F2, \ldots, Fm)$ and $Fi = (f1i, f2i, \ldots, fli)T$ $(i \in \{1, \ldots, m\})$, where $l \leq 2p - 1$.

Note all these blocks are elements of $GF(2p)$. The systematic layout with parity vectors is achieved with the information dispersal matrix$A$, derived from $m \times (m + k)$ Vandermonde matrix.

$$\begin{pmatrix} 1 & 1 & \ldots & 1 & 1 & \ldots & 1 \\ \beta_1 & \beta_2 & \ldots & \beta_m & \beta_{m+1} & \ldots & \beta_n \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \beta_1^{m-1} & \beta_2^{m-1} & \ldots & \beta_m^{m-1} & \beta_{m+1}^{m-1} & \ldots & \beta_n^{m-1} \end{pmatrix}$$

, where βj $(j \in \{1, \ldots, n\})$ are distinct elements randomly picked from$GF(2p)$. After a sequence of elementary row transformations, the desired matrix A can be written as

$$\mathbf{A} = (\mathbf{I}|\mathbf{P}) = \begin{pmatrix} 1 & 0 & \ldots & 0 & p_{11} & p_{12} & \ldots & p_{1k} \\ 0 & 1 & \ldots & 0 & p_{21} & p_{22} & \ldots & p_{2k} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & 1 & p_{m1} & p_{m2} & \ldots & p_{mk} \end{pmatrix}$$

Where I is a $m \times m$ identity matrix and P is the secret parity generation matrix with size $m \times k$. Note that A is derived from a Vandermonde matrix, thus it has the property that any m out of the $m + k$ columns form an invertible matrix. By multiplying $\boldsymbol{F}$ by $\boldsymbol{A}$, the user obtains the encoded file:

$G = F \cdot A = (G(1), G(2), \ldots, G(m), G(m+1), \ldots, G(n)) = (F1, F2, \ldots, Fm, G(m+1), \ldots, G(n))$,

where $G(j) = (g(j)1, g(j)2, \ldots, g(j)\, l)T$ $(j \in \{1, \ldots, n\})$.

As noticed, the multiplication reproduces the original data file vectors of $\boldsymbol{F}$ and the remaining part ($G(m+1,\ldots,G(n))$ are k parity vectors generated based on $\boldsymbol{F}$.

IV. CHALLENGE TOKEN PRECOMPUTATION

In order to achieve assurance of data storage correctness and data error localization simultaneously, this scheme entirely relies on the pre-computed verification tokens. The main idea is as follows: before file distribution the user pre-computes a certain number of short verification tokens on individual vector G(j) (j ∈ {1, . . . , n}), each token covering a random subset of data blocks. Later, when the user wants to make sure the storage correctness for the data in the cloud, he challenges the cloud servers with a set of randomly generated block indices. Upon receiving challenge, each cloud server computes a short "signature" over the specified blocks and returns them to the user. The values of these signatures should match the corresponding tokens pre-computed by the user. Meanwhile, as all servers operate over the same subset of the indices, the requested response values for integrity check must also be a valid codeword determined by secret matrix P.

Suppose the user wants to challenge the cloud servers t times to ensure the correctness of data storage. Then, he must pre-compute t verification tokens for each $G(j)$ $(j \in \{1, \ldots, n\})$, using a PRF $f(\cdot)$, a PRP $\Phi(\cdot)$, a challenge key $kchal$ and a master permutation key K_{PRP}. To generate the i^{th} token for server j, the user acts as follows:

1. Derive a random challenge value α_i of GF(2^p) by $\alpha_i = f_{kchal}$ (i) and a permutation key $k^{(i)}_{prp}$ based on K_{PRP} .
2. Compute the set of r randomly-chosen indices: $\{Iq \in [1, \ldots, l] | 1 \leq q \leq r\}$,where Iq = $\Phi k_{prp}^{(i)}$ (q).
3. Calculate the token as:

$$v_i^{(j)} = \sum_{q=1}^{r} \alpha_i^q * G^{(j)}[I_q], \text{ where } G^{(j)}[I_q] = g_{I_q}^{(j)}.$$

Note that v(j)i , which is an element of GF(2p) with small size, is the response the user expects to receive from server j when he challenges it on the specified data blocks.

The details of token generation are shown in Algorithm 1.

procedure
  Choose parameters l, n and function f, ϕ;
  Choose the number t of tokens;
  Choose the number r of indices per verification;
  Generate master key K_{prp} and challenge k_{chal};
  for vector $G^{(j)}$, $j \leftarrow 1, n$ **do**
    for round $i \leftarrow 1, t$ **do**
      Derive $\alpha_i = f_{k_{chal}}(i)$ and $k_{prp}^{(i)}$ from K_{PRP}.
      Compute $v_i^{(j)} = \sum_{q=1}^{r} \alpha_i^q * G^{(j)}[\phi_{k_{prp}^{(i)}}(q)]$
    end for
  end for
  Store all the v_is locally.
end procedure

Algorithm 3.1: Token Pre-computation

V. SIGNCRYPTION OF PRE-COMPUTED TOKENS

After token generation, the user generates the signcrypted text (R,C, s) by following the below steps [4][5]:

1) Checks the validity of $Cert_A$ and uses it for verifying W_A.

The process of certificate validation includes:

a. Verifying the integrity and authenticity of the certificate by verifying the CA's signature on the certificate.
b. Verifying that the certificate is not expired.
c. Verifying that the certificate is not revoked.

2) Randomly selects an integer $r \in_R [1, n-1]$.
3) Computes $R = rG$ where $R = (x_R , y_R)$ in which x_R / y_R denotes the x/y-coordinate of the point R.
4) Computes $K = (r + \ddot{x}_R w_U) W_A$ where $K = (x_K , y_K)$,and $\tilde{x}_R = 2^{\lceil f/2 \rceil} + (x_R \bmod 2^{\lceil f/2 \rceil})$ in which $f = \lfloor \log_2 n \rfloor + 1$ is the bit length of n, $\lfloor . \rfloor$ denotes the floor, and $\lceil . \rceil$ indicates the ceiling. If $K = O$ user goes back to step 2 otherwise, it drives the session key of encryption as k = H(x_K || ID_U || y_K || IDc) in which H is a oneway hash function that generates the required number of bits as the secret key of deployed symmetric encryption algorithm, and || denotes the concatenation.
5) Computes the ciphertext as $C = Ek(M)$ in which E_k(.) denotes a strong symmetric encryption algorithm (e.g. AES) that uses session key k for the encryption.
6) Computes the digital signature as $s = tw_A - r(mod\ n)$ in which t = H(C || x_R || ID_U || y_R || IDc) .
7) Sends the signcrypted text i.e token (R,C, s) to Authentication Server.

VI. UNSIGNCRYPTION OF TOKENS.

AS who received the signcrypted text (R,C, s) , follows the below steps to extract the plaintext and verify the signature[4][5]:

1) Checks the validity of $Cert_U$ and uses it for verifying W_U.
2) Computes K = w_A (R + $\ddot{x}_R W_U$) = (x_K , y_K) and derives the session key as k = H(x_K || ID_U || y_K || ID_A) .
3) Decrypts the ciphertext as $M = D_k(C)$.
4) Computes t = H(C || x_R || ID_U || yR || ID_A).
5) Accepts M as the correct plaintext of User if and only if $sG + R = tWA$ .Otherwise, he rejects M.

Once all tokens are computed, sent encrypted and success acknowledgement message is received from AS to User, the final step before file distribution is to blind each parity block $g_i^{(j)}$ in $(G(m+1), \ldots, G(n))$ by $g(j)i \leftarrow g(j)\, i + fkj\,(sij), i \in \{1, \ldots, l\}$, where k_j is the secret key for parity vector $G(j)$ $(j \in \{m +$

$1, \ldots, n\}$). This is for protection of the secret matrix P. After blinding the parity information, the user disperses all the n encoded vectors $G(j)$ ($j \in \{1, \ldots, n\}$) across the cloud servers $S1, S2, \ldots, Sn$.

VII. CORRECTNESS VERIFICATION AND ERROR LOCALIZATION

Error localization is a key prerequisite for eliminating errors in storage systems. However, many previous schemes do not explicitly consider the problem of data error localization, thus only provide binary results for the storage verification. Proposed scheme outperforms those by integrating the correctness verification and error localization in the challenge-response protocol: the response values from servers for each challenge not only determine the correctness of the distributed storage, but also contain information to locate potential data error(s). Specifically, the procedure of the i-th challenge-response for a cross-check over the n servers is described as follows:

- The user reveals the α_i as well as the i-th permutation key $k^{(i)}_{prp}$ to each servers.
- The server storing vector $G^{(j)}$ aggregates those r rows specified by index $k^{(i)}_{prp}$ into a linear combination:

$$R_i^{(j)} = \sum_{q=1}^{r} \alpha_i^q * G^{(j)}[\phi_{k_{prp}^{(i)}}(q)].$$

- Upon receiving $R^{(j)}_i$s from all the servers, the user takes away blind values in $R^{(j)}$ ($j \in \{m+1, \ldots, n\}$) by:

$$R_i^{(j)} \leftarrow R_i^{(j)} - \sum_{q=1}^{r} f_{k_j}(s_{I_q,j}) \cdot \alpha_i^q, \text{where } I_q = \phi_{k_{prp}^{(i)}}(q).$$

- Then the user verifies whether the received values remain a valid codeword determined by secret matrix P:

$$(R_i^{(1)}, \ldots, R_i^{(m)}) \cdot \mathbf{P} \stackrel{?}{=} (R_i^{(m+1)}, \ldots, R_i^{(n)}).$$

Because all the servers operate over the same subset of indices, the linear aggregation of these r specified rows (R(1)i , . . . ,R(n)i) has to be a codeword in the encoded file matrix. If the above equation holds, the challenge is passed. Otherwise, it indicates that among those specified rows, there exist file block corruptions. Once the inconsistency among the storage has been successfully detected, one can rely on the pre-computed verification tokens to further determine where the potential data error(s) lies in. Note that each response R(j) i is computed exactly in the same way as token v(j) i , thus the user can simply find which server is misbehaving by verifying the following n equations:

$$R_i^{(j)} \stackrel{?}{=} v_i^{(j)}, j \in \{1, \ldots, n\}.$$

Algorithm 2 gives the details of correctness verification and error localization.

1: **procedure** CHALLENGE(i)
2: Recompute $\alpha_i = f_{k_{chal}}(i)$ and $k_{prp}^{(i)}$ from K_{PRP};
3: Send $\{\alpha_i, k_{prp}^{(i)}\}$ to all the cloud servers;
4: Receive from servers:
$\{R_i^{(j)} = \sum_{q=1}^{r} \alpha_i^q * G^{(j)}[\phi_{k_{prp}^{(i)}}(q)] | 1 \le j \le n\}$
5: **for** ($j \leftarrow m+1, n$) **do**
6: $R^{(j)} \leftarrow R^{(j)} - \sum_{q=1}^{r} f_{k_j}(s_{I_q,j}) \cdot \alpha_i^q$, $I_q = \phi_{k_{prp}^{(i)}}(q)$
7: **end for**
8: **if** $((R_i^{(1)}, \ldots, R_i^{(m)}) \cdot \mathbf{P} == (R_i^{(m+1)}, \ldots, R_i^{(n)}))$ **then**
9: Accept and ready for the next challenge.
10: **else**
11: **for** ($j \leftarrow 1, n$) **do**
12: **if** ($R_i^{(j)} != v_i^{(j)}$) **then**
13: **return** server j is misbehaving.
14: **end if**
15: **end for**
16: **end if**
17: **end procedure**

Algorithm 2

VIII. FILE RETRIEVAL AND ERROR RECOVERY

Since the layout of file matrix is systematic, the user can reconstruct the original file by downloading the data vectors from the first m servers, assuming that they return the correct response values. Notice that the verification scheme is based on random spot-checking, so the storage correctness assurance is a probabilistic one. However, by choosing system parameters (e.g., r, l, t) appropriately and conducting enough times of verification, this scheme can guarantee the successful file retrieval with high probability. On the other hand, whenever the data corruption is detected, the comparison of pre-computed tokens and received response values can guarantee the identification of misbehaving server(s), again with high probability, which will be discussed shortly. Therefore, the user can always ask servers to send back blocks of the r rows specified in the challenge and regenerate the correct blocks by erasure correction, shown in Algorithm 3, as long as there are at most k misbehaving servers are identified. The newly recovered blocks can then be redistributed to the misbehaving servers to maintain the correctness of storage.

1: **procedure**
% Assume the block corruptions have been detected among
% the specified r rows;
% Assume $s \leq k$ servers have been identified misbehaving
2: Download r rows of blocks from servers;
3: Treat s servers as erasures and recover the blocks.
4: Resend the recovered blocks to corresponding servers.
5: **end procedure**

Algorithm 3

IX. CONCLUSION

Cloud Computing is gaining remarkable popularity in the recent years for its benefits in terms of flexibility, scalability, reliability and cost effectiveness. Despite all the promises however, Cloud Computing has one problem, Security so in this thesis, we have studied the problem of data security in cloud data storage, various schemes proposed to ensure the correctness of users's data in the cloud server and have proposed a scheme which includes Public Verifiability using signcryption.

In this scheme signature generation algorithm and key pair generation algorithm needs a random number to be generated. Using this random number as seed private keys is generated. Similarly secret integer 'K' generated during signature verification algorithm should also be random in nature. An attacker can exploit this vulnerability if the algorithm used to generate the random number is not cryptographically secure i.e. it should be unpredictable. So probability of any given value being selected should be very small. As a future scope of this work cryptographically secure random number should be included while generating private keys.

REFERENCES

[1] Keane an NTT DATA Company, "Cloud Computing".
[2] K. Kajendran, J.James Jeyaseelan J. Jakkulin Joshi, "An Approach For Secured Data Storage Using Cloud Computing", International Journal of Computer Trends and Technology, 2011.
[3] Cong Wang, Qian Wang, Kui Ren , Wenjing Lou, "Ensuring Data Storage Security in Cloud Computing".
[4] Yuliang Zheng, "Signcryption or How to Achieve Cost(Signature & Encryption) << Cost(Signature) +Cost(Encryption)",1999.
[5] Yuliang Zheng, Hideki Imai, "How to construct efficient signcryption schemes on elliptic curves", Information Processing Letters 68 (1998) 227–233,1988.