4th INTERNATIONAL CONFERENCE ON
RESEARCH IN ENGINEERING, SCIENCE AND TECHNOLOGY
چهارمین کنفرانس بین المللی پژوهش در مهندسی، علوم و تکنولوژی
6th. September 2016 Athens GREECE
Athens
آتن یونان ۱۶ شهریورماه ۱۳۹۵

A Time Optimized Secure Method for Digital Signatures

Mehdi Goodarzi
Department of Computer Engineering, Sepidan Branch, Islamic Azad University Sepidan, Iran
Email: goodarzi689@gmail.com

Ahmad Mosallanejad
Department of Computer Engineering, Sepidan Branch, Islamic Azad University Sepidan, Iran
Email: ahmad.upm@gmail.com

Abstract
Cloud computing is now an important part of IT enterprise. In recent years, organizations such as banks, hospitals, medical centers etc. have started to rely on cloud services. But there are challenges in the use of cloud computing, such as ensuring the reliability, data integrity and so on. Ensuring data integrity and security, while setup cost is optimal, is one of the most important challenges. In this paper, we propose a fast and secure method to create and evaluate digital signatures. Our method (E-ECDSA) is based on Elliptic Curve Digital Signature Algorithm and is a complement to it. In particular, we consider methods for reducing the burden of generating a constant amount of metadata at the client side. Whereas many prior studies on ensuring remote data integrity did not consider the burden of generating verification metadata at the client side, the objective of this study is to resolve this issue.

Keywords: Cloud Computing, Digital Signatures, Elliptic Curve, Data Integrity, Setup Time

1. Introduction

Cloud computing is the utilization of Internet for the assignments performed on the local machine, hardware and software demand maintained somewhere else. Cloud is widely used everywhere owing to its convenience, be it in a simple data analytic program or composite web and mobile applications. Local computers no longer have to do all the heavy lifting when it comes to running applications. The network of computers that make up the cloud handles them instead. The only thing the user's computer needs to run is the cloud computing systems interface software, which can be as simple as a Web browser and the cloud's network takes care of the rest (Wang et al, 2009).
Some most important benefits of cloud computing listed below:

- Flexibility
- Disaster recovery
- Work from anywhere (portability)

The aim of cloud computing is to provide reliable, customized and guaranteed computing dynamic environment on Pay-per-use or Pay-as-you-go basis for the end users. Depending on the services provided by the cloud, it is divided into main three categories:

- **IAAS**: this is a cloud service which gives an access to the hardware resources and assets, for example, computing hardware as pay per use basis. Case of this kind of service is assume you pay month to month or yearly membership to hostingcompany which in turn stores your files on their server (Wang et al, 2009) (Ruiter and Warnier, 2011).
- **SAAS**: it provides a software services to the end user. Examples of this kind of service are Web-based email and Google documents. End user gets access to this software service but he or she cannot modify this software utility. Software is configured on cloud utility do not need to be installed on end user computer (Wang et al, 2009) (Ruiter and Warnier, 2011).
- **PAAS**: this service provides a platform or an environment in order to end user can develop his own application. for example Google App (Wang et al, 2009) (Ruiter and Warnier, 2011).

While it have lots of benefits, cloud computing has many challenging design issues which demand great knowledge affecting on the security and performance of the overall system. One of the biggest concerns with cloud data storage is that integrity verification of cloud data at untrusted server.
One of the most important issues in cloud computing is setup time optimization without loss of security and data integrity. In this study, a model based on elliptic curves algorithms is provided in which exponential complexity reduced to multiplicative for Preparations blocks. In this model, data security has been increased.
Cloud computing has many challenging design issues which demand great knowledge affecting on the security and performance of the overall system. One of the biggest concerns with cloud data storage is that integrity verification of cloud data at untrusted server.
One of the most important issues in cloud computing is setup time optimization without loss of security and data integrity. In this study, a model based on elliptic curves algorithms is provided in which exponential complexity reduced to multiplicative for Preparations blocks. In this model, data security has been increased.

2. Mathematical Background

In this section we give a quick and brief introduction to elliptic curves (Cheng, 2004) as mathematical background of our work. More detailed information is available in (Erway et al, 2012).

2.1. Elliptic Curves

Let $a_i \in F$, where F is a nite eld. $\bar{F}$ is the algebraic closure of F. Let E be a cubic curve defined by the general Weierstrass equation $y^2 + a_1xy + a_3y = x^3 + a_2x^2 + x_4 + a_6$ over F, that is $E = \{(x, y) \in \bar{F}^2 : y^2 + a_1xy + a_2y = x^3 + a_2x^2 + x_4 + a_6 \} \cup \{\infty\}$, denoted by $E(\bar{F})$.
In this definition, E is a set of points, whose both coordinates are in $\bar{F}$, which means E is the solution set for the equation on $\bar{F}$, but not F. Rational Points: assume that L be the algebraic extension of $F \subseteq L$. If both the coordinates of $P \in E$ lie in L, or $P = \infty$, we say P is L-rational. The set of L-rational points of E is denoted by E(L).

2.2. Elliptic Curves Over F_p

Let $p \geq 3$ be a prime, $a, b \in Fp$ satisfy that the discriminant $\Delta = 4a^3 + 27b^2 \neq 0$ ($a_1 = a_2 = a_3 = 0$, $a_4 = a$ and $a_6 = b$ corresponding to the general Weierstrass equation), which denes the elliptic curves without singularity. The curve is of a simple form: $y^2 = x^3+ax+b$ with $a, b \in F_p$. Then an elliptic curve $E(F_p)$ over F_p consists of a set of points$\{P = (x, y)|y^2 = x^3 + ax + b, x, y, a, b \in F_p\}$together with ∞.

3. Related works

In this section, first we describe some data integrity techniques in cloud. And then we will described some very important algorithms for digital signatures briefly. Digital signature schemes are designed to provide the digital counterpart to handwritten signatures (and more). A digital signature is a number dependent on some secret is known only to the signer (the signers private key), and, additionally, on the contents of the message being signed (Johnson et al, 2001).

3.1. Data integrity techniques incloud

Proof of retrievability: Proof of retrievability (Juels, 2007) means Verify the data stored by user at remote storage in the cloud is not modified by the cloud. POR for huge size of files named as sentinels. The main role of sentinels is cloud needs to access only a small portion of the file (F) instead of accessing entire file. This technique uses the auditing protocol when solving the problem of integrity. Here any client who wants to check the integrity of the outsourced data then there is no need to retrieve full content. Here user stores only a key, which is used to encode a file F which gives the encrypted file F'. This procedure leaves the set of sentinel values at the end of the file F'. Server only stores F'. Server does not know that where the sentinel value are stored because they indistinguishable from regular and it is randomly stored in the file F'.

Consider user or cloud client wants to store a file (F) in the cloud server or archive. Before storing the file to the cloud, owner needs to encrypt the file in order to prevent from the unauthorized access. When client send the challenge to the server to check for integrity, at that time in challenge response protocol server will ask to return a subset of sentinels in F. If the data is tampered or deleted the sentinels may get corrupted or lost and so the server is unable to generate the proof of the original file. In this way client can prove that server has modified or corrupted the file. POR is designed to be lightweight; it attempts to provide minimum storage in client and server side, number of data blocks accessed, the communication complexity of an audit. POR also provides the error-correcting codes to recover file having small fraction being corrupted. POR can be applied to the static files only. File needs to be encrypted before uploading to the server. It requires additional space to hide sentinels (Kumar and Saxena, 2011) (Shokrollahi, 2006) (Pandya and Sutaria, 2012).

Provable data possession (PDP): In this technique, the data is pre-processed before sending it to the cloud server. Here the data is filled with some tag value or say meta-data for the verification at the client side. Now entire data is sent over to the server and at the client side meta-data is stored. This meta-data is used for the verification when user need for it. When user wants to check for integrity it will sends the challenge to the server at that time server will respond with the data. Now the client will compare the reply data with the local meta-data. In this way client will say that the data is modified or not. Original PDP has low computation and storage overhead. It supports both encrypted data and plain data. It offers public verifiability. It is efficient because small portion of the file needs to be accessed to generate proof on the server. This technique is only applicable to the static files (i.e. append-files only). Probabilistic guarantees may result in false positive. Homomorphic hashing technique is used to compose multiple block inputs into a single value to reduce the size of proof (Shokrollahi, 2006) (Pandya and Sutaria, 2012).

PDP has various sub-techniques. For example Scalable PDP is an improved version of the original PDP. The difference is that Scalable PDP uses the symmetric encryption while original PDP uses public key to reduce computation overhead. Scalable PDP can have dynamic operation on remote data. Scalable PDP has all the challenges and answers are pre-computed and limited number of updates ().

And dynamic PDP that supports full dynamic operations like insert, update, modify, delete etc. Here in this technique the dynamic operation enables the authenticated insert and delete functions with rank-based authenticated directories and with a skip list. Although DPDP has some computational complexity, it is still efficient. For example, to generate the proof for 1GB file, DPDP only produces 415KB proof data and 30ms computational overhead (Galindo and Garcia, 2009).

3.2. RSA

This algorithm was developed at MIT in 1997 for the first time (Rivest et al, 1978). The RSA idea is based on the factorization of big numbers, which says the larger sequence of numbers you have got, the more you are protected. The RSA provides a robust security; so attackers cannot break RSA by factoring, due to its complexity and large keys. RSA is employed to encrypt/decrypt data and also has the ability to sign and/or verify the data packets. RSA doesn't mandate the utilization of a particular

hash function, therefore the security of the signature and encryption are partially dependent on the hash function selected to use for computing the signature.

3.3. ECDSA

ECDSA is based on elliptic curves (Koblitz, 1991). ECDSA schemes provide the same functionality as RSA schemes including sign and/or verify signed packets. There are some environments where 1024-bit RSA cannot be implemented, while 192-bit ECDSA can. For this reason, the strength-per-key-bit is substantially greater in an algorithm that uses elliptic curves (Johnson et al, 2001). More information about mathematical backgrounds of elliptic curves can be found in (Johnson et al, 2001) (Caelli, 1999).
Here is simple and short step-by-step key/signature generation and verification process in ECDSA (Vanstone and Hankerson, 2004).

Key Generation

An entity A's key pair is associated with a particular set of EC domain parameters D = (q, FR, a, b, G, n, h). E is an elliptic curve defined over F_q, and P is a point of prime order n in $E(F_q)$, q is a prime. Each entity A does the following:

1. Select a random integer d in the interval [1, n−1].
2. Compute Q = dP.
3. Public key of A is Q, private key of A is d.

Signature Generation

To sign a message m, an entity A with domain parameters D = (q, FR, a, b, G, n, h) does the following:

1. Select a random or pseudorandom integer k in the interval [1, n−1].
2. Compute

$$kP = x_1, y_1 \quad (1)$$

$$r = x_1 \bmod n \quad (2)$$

(where x_1 is regarded as an integer between 0 and q−1). If r = 0 then go back to step 1.

3. Compute

$$k^{-1} \bmod n \quad (3)$$

4. Compute

$$s = k^{-1} h(m) + dr \bmod n \quad (4)$$

where h is the Secure Hash Algorithm (SHA-1). If s= 0, then go back to step 1.

5. The signature for the message m is the pair of integers (r, s).

Signature Verification

To verify A's signature (r, s) on m, B obtains an authenticated copy of as domain parameters D = (q, FR, a, b, G, n, h) and public key Q and do the following:

1. Verify that r and s are integers in the interval [1, n−1].
2. Compute

$$w = s^{-1} \bmod n, h(m) \quad (5)$$

3. Compute

$$u1 = h(m)w \bmod n \quad (6)$$

$$u2 = rw \bmod n \quad (7)$$

4. Compute

$$u1\,P + u2\,Q = (x0,y0) \quad (8)$$

$$v = x0 \bmod n \quad (9)$$

5. Accept the signature if v = r.

The process of key generation and key verification of ECDSA is in figures 1, 2. The main advantage of ECDSA is that the party authenticating the peripheral is alleviated from the constraint to securely store a secret. The authenticating party will authenticate due to a public key which can be freely distributed.

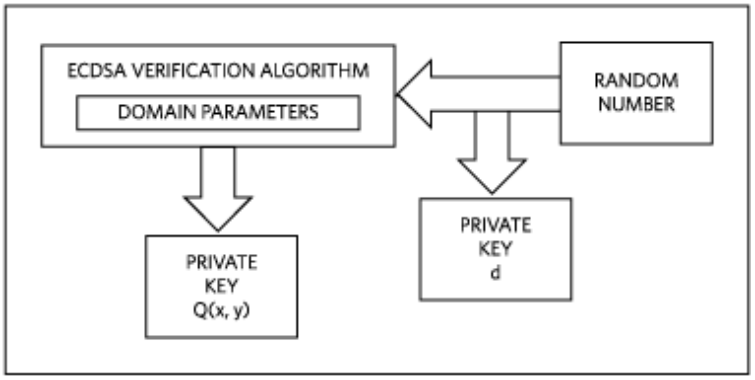

Figure (1) the process of key generation in ECDSA

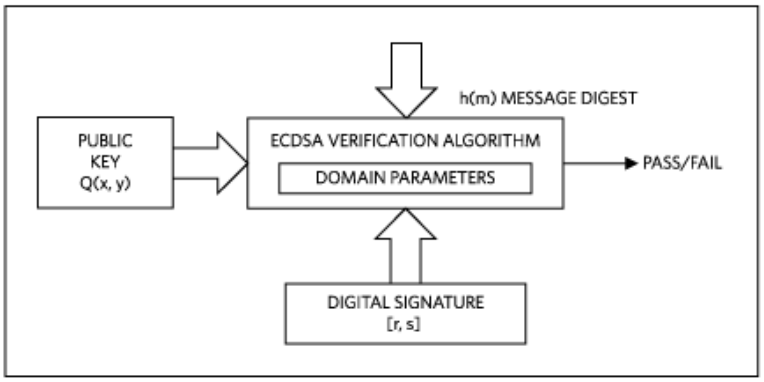

Figure (2) the process of key verification in ECDSA

3.4. ECGDSA

Another algorithm for digital signatures is Elliptic Curve German Digital Signature Algorithm (ECGDSA). In this scheme, the elliptic curve point $P_A' := [d_A^{-1} \bmod n]G$ is used as public key. As a consequence, the signature creation requires no computation of a multiplicative inverse mod n (Wang, 2009).

Like ECDSA in this method the signature depends on the random number k. If A signs the same document M twice, both signatures will be different.

This is very similar to ECDSA. Here is ECGDSA algorithms for generation and verification.

Signature generation: In order to sign the message m with the private key dA, the signer A performs the following steps:

1. A computes the hash value
   $h(m), 0 \leq h(m) < q$
2. A chooses a random integer in interval [1, q−1].
3. A determines
   $r = x(kG) \bmod q$ (10)

If r = 0, A goes back to step 2.

4. A computes the value
   $s = (krh(m))dA \bmod q$ (11)

If s = 0, A goes back to step 2.

5. The pair (r, s) is the signature of A for the message m.

Signature Verification

In order to verify whether a pair (r, s) is the signature of A for the message m, the variable B performs the following steps, using public key P_A of A:

1. B checks whether r and s are in [1, q−1]. If not, (r, s) is not accepted signature.
2. B computes the hash value h(m)<q.
3. B computes the value
   $u1 = r\text{-}1\ h(m) \bmod q$ (12)
4. B computes the value
   $u2 = r\text{-}1\ s \bmod q$ (13)
5. B determines
   $x(u1\ G + u2\ PA) \bmod q$ (14)
6. If the last value is equal to r, the pair (r, s) is accepted as signature of A for the message m.

3.5. EC-Schnorr

there is an other elliptic based signature scheme that is a conversion of Schnorr scheme (Giri and Gaur, 2015).

Key generation: Entity A selects a random integer dA from the interval [1, n-1] as her private key, and publishes $Q_A = -d_A G$ as her/his public key. Signing scheme:

1. Select a random integer k from interval [1, n-1].
2. Compute
   $R = kG = (x1, y1)$ (15)
   $r = x1 \bmod n$ (16)
   $e = h(r, m)$ (17)

where h(r,m) is hash function:$Fq \times \{0,1\}^{|m|} \rightarrow F_n$. If e = 0 then goto step 1.

3. Compute
   $w = k + dA\ e \bmod n$ (18)

if w = 0 then goto step 1.

4. (e, w) is the signature of A for message m.

Verifying scheme:

1. Verify that e, w are in the interval [1, n-1].
2. Compute
   $V = wG + eQA = (x1, y1)$ (19)
   $r0 = x1 \bmod n(r = x1)$ (20)
   $e = h(r0, m)$ (21)
3. Accept if and only if $e = e_0$.

3.6. BLS Algorithm

BLS (Boneh-Lynn-Shacham) signature scheme enables users to verify if a signer is authentic or not. The scheme uses a pairing function for verification and signatures are group elements in some elliptic curve.

BLS describes a short signature scheme where 170-bit signatures provide approximately 280 security, in the random oracle model. The system makes use of a group G where:

1. The Computational Diffie-Hellman problem is intractable
2. There is an efficiently computable, non-degenerate, bilinear map e: $G \times G \rightarrow G1$ for some group G1.

There are several examples of such groups from algebraic geometry where the bilinear map is implemented using the Weil pairing. Given such a group G of prime order q the signature scheme works as follows (Boneh and Boyen, 2004):

Key Generation

1. Pick an arbitrary generator g ∈ G.
2. Pick a random ∈{1,...,q}and set $y = g^{\alpha} \in G$.
3. Let H be a hash function H: $\{0, 1\}^* \rightarrow G$. Output (g, y, H) as the public key and (g, α, H) as the private key.

Signing

To sign a message $m \in \{0, 1\}^*$ using the private key (g,α,H) output $H(m)^{\alpha} \in G$ as the signature.

Verification

To verify a message/signature pair(m, s) $\in\{0,1\}^* \times$ Guessing the public key(g, y, H) test if e(g, s)= e(y, H(m)). If so, accept the signature. Otherwise, reject.

4. E-ECDSA Method

In this section we will describe our method (E-ECDSA) and our mechanism for robustness assessment of this work. A simple flowchart for E-ECDSA algorithm can be found in appendix. Obviously if the value of k in digital signature is insecure or pseudorandom, we will have a low security. So in our purposed method we generate two trusted integers k_1 and k_2. This may have more overhead than ECDSA but this is not so big.

4.1. Generate digital signatures in E-ECDSA

1. Select two random or pseudorandom integer k1 and k2 in the interval [1,n−1].
2. Compute
   r1 = x1 mod n (22)
   k1 × P = (x1,y1) (23)
   r2 = x2 mod n (24)
   k2 × P = (x2,y2) (25)

(where x1 is regarded as an integer between 0 and q−1). If r = 0 then go back to step 1.

3. Compute
   r = r1 + r2 (26)
4. Compute
   d−1 mod n (27)
5. Compute
   s = d−1(rk1 − h) mod P (28)

(where h is the Secure Hash Algorithm (SHA-1). If s = 0, then go back to step 1.

6. The resulted signature is (r_1, r_2, s).

4.2. Verification in E-ECDSA

1. Compute
   r = r1 + r2 (29)
2. Check if received r, s is in [1, n−1] or not.
3. Compute
   w = r−1 mod n, h(m) (30)
4. Compute
   u1 = h(m)w mod n (31)
   u2 = r2w mod n (32)
5. Compute
   u1P + u2Q = (x0,y0) (33)
   v = x0 mod n (34)
6. If v = r_1, the signature is verified.

4.3. Method description

Our method is more secure because re-generate integers K_1 and K_2 is very hard for any probable attacker. And this difficulty is independent to the method of generating random numbers, because after multiplication K_1, K_2 in P they are added and multiply to K_1 again.

To show robustness and speed of our method, we have measure the time of our algorithm and BLS algorithm under same conditions with 100,300 and 500 blocks. The simulation platform was JAVA and the used hardware had dual-core CPU with 2G RAM memory. And a 1.8MB Image was used to do our simulation.

5. Results and conclusions

Results show that our algorithm (E-ECDSA) can handle both generating signature on client side and verifying signature on server side, in a better time cost. As can be seen in table 1, time of our algorithm and BLS algorithm was compared. The test has done under condition described in section 3. As the table1 shows, the time for generating signature reduced 67% and the time for verifying it reduced more than 71%. That is a significant improvement in the setup time. In figures 4 and 5 also compared three algorithms BLS, ECDSA and E-ECDSA times for generate and verify digital signatures.

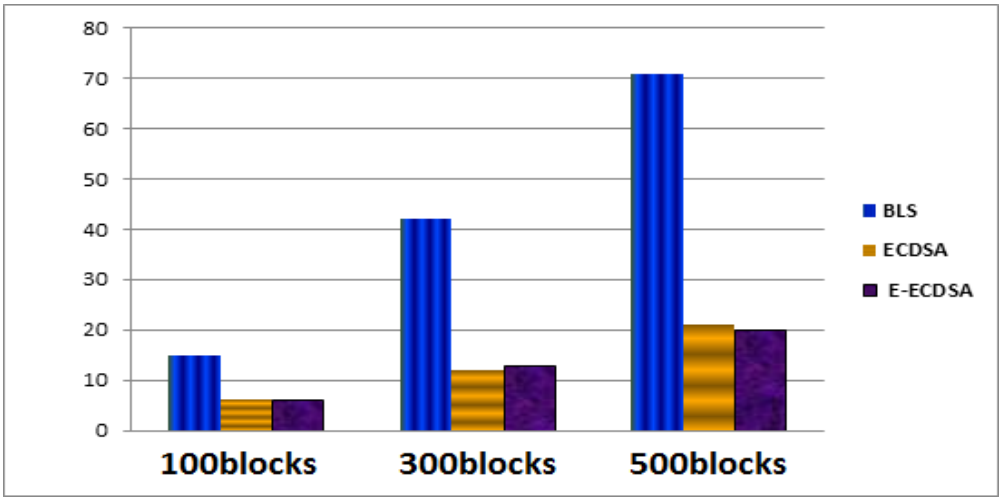

Figure (4) times of generation signature

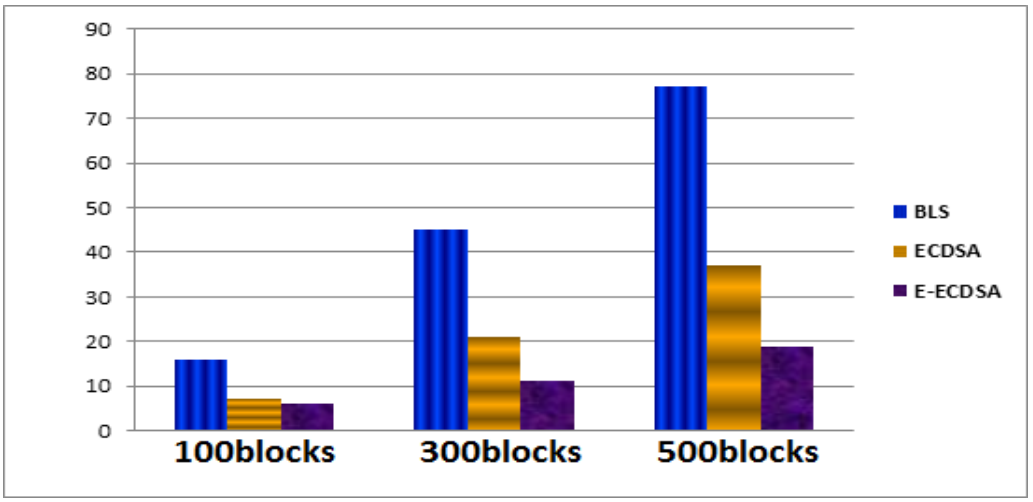

Figure (5) times of verifying signature

So the overall result shows that methods based on ECC, are faster than Weil Pairing algorithms (like BLS). And as described above our purposed method can solve the problem of spoofing private key with attackers by using K_1 and K_2 integers. But the worst case of setup time in E-ECDSA is slightly more than ECDSA, which can be ignored by Impressive advances in security and data integrity.
The most important advantage of E-ECDSA is the low growth of time, that means if the number of blocks increase the time needed for generating and verifying signatures increases very slightly in comparison to ECDSA and other algorithms.

Table 1-Experimental results

Number of Blocks	Time for GENERATING signature in E-ECDSA (sec)	Time for GENERATING signature in BLS (sec)	Time for VERIFYING signature in E-ECDSA (sec)	Time for VERIFYING signature in BLS (sec)
100	6	15	6	16
300	13	42	11	45
500	20	71	19	77

References

Ateniese, G., Pietro, R.D., Mancini, L. V. and Tsudik, G. (2008). Scalable and efficient provable data possession. IACR Cryptology ePrint Archive.
Boneh, D. and Boyen, X. (2004). Short signatures without random oracles. Advances in Cryptology. PP: 56–73.
Caelli, W.J., Dawson, E. P. and Rea, S. A. (1999). PKI, elliptic curve cryptography, and digital signatures. Computers and Security. PP: 47-66.
Cheng, Z. (2004). Simple tutorial on elliptic curve cryptography. School of Computing Science.
Erway, C.C., Kp, A., Papamanthou, C., and Tamassia, R. (2012). Dynamic provable data possession. ACM Conference on Computer and Communications Security. pp: 213–222.
Galindo, D. and Garcia, F.A. (2009). schnorr-like lightweight identity-based signature scheme. In: Proceedings of AfricaCrypt,LNCS 5580. Vol 5580, PP:135–48.
Giri, M. and Gaur, B.a. (2015). A survey on data integrity techniques in cloud computing.International Journal of Computer Applications. PP: 27-32.
Johnson, D., Menezes, A. and Vanstone, S. (2001). The elliptic curve digital signature algorithm (ECDSA). International Journal of Information Security . Vol 1, PP: 36–63.
Juels, A. and Jr., B.S.K. Pors. (2007). Proofs of retrievability for large files. IACR Cryptology ePrint Archive.
Koblitz, N., 1991. Elliptic curve implementation of zero-knowledge blobs. Journal of Cryptology.
Kumar, R. S. and Saxena, A. (2011). Data integrity proofs in cloud storage. in David B. Johnson and Anurag Kumar, ed., 'COMSNETS' , IEEE . PP: 1-4.
Pandya, R. and Sutaria, K. (2012). An analysis of privacy techniques for data integrity in the cloud environment. International Journal of Computer and Electronics Engineering.
Rivest, R.L., Shamir, A. and Adleman, L. (1978). A method for obtaining digital signatures and public-key cryptosystems. Communications of the ACM . PP: 120–126.
Ruiter, J. and Warnier, M. (2011). Privacy regulations for cloud computing. Compliance and Implementation in Theory and Practice. Computers, Privacy and Data Protection, Springer. pp: 361–376.
Shokrollahi, J. (2006). Efficient implementation of elliptic curve cryptography on fpgas. PhD diss, University of Bonn.
Vanstone, S. and Hankerson, D.a.M.A. (2004). Guide to elliptic curve cryptography. Springer-Verlag.
Wang, C., Wang, Q., Ren, K. and Lou, W. (2009). Ensuring data storage security incloud computing. IACR Cryptology ePrint Archive. Vol 81.

Appendix