Application of Elliptic Curve Cryptography in Pretty Good Privacy (PGP)

Sanchit Mehrotra
Integrated Dual Degree Student, Department of CSE
IIT (BHU), Varanasi
Varanasi, India
sanchit.mehrotra.cse11@iitbhu.ac.in

Prof. Arun Kumar Agrawal
Professor and Ex-Head, Department of CSE
IIT (BHU), Varanasi
Varanasi, India
akagrawal.cse@iitbhu.ac.in

***Abstract*— Cryptographic Security is not so much an understandable term. Those who understand it realize its significance and the role that it plays in various domains including finance, e-commerce, browsing the web, wireless security, nuclear researches, etc. Cryptographers have already realized that the conventional public cryptographic schemes based on RSA are bound to fail, keeping in mind the rising demand and usage of mobile devices running on low power, as well as the emergence of newer technology and specialized algorithms that can solve the problem of factoring large primes quickly and effectively. RSA which is based on the problem of factoring large primes is surely doomed because of these reasons. Elliptic Curve Cryptography (ECC) is hence the need of the hour. Not only is it suitable for mobile devices running on low power, as it can provide an equivalent security at lower key sizes, but also it is equally difficult to break, owing to a better trapdoor function and difficult mathematics, provided that it is implemented correctly with strong parameters. In this paper, we discuss about the application of ECC in an area of Electronic Mail Security, Pretty Good Privacy (PGP), showing how it can replace RSA completely in this field. I have also discussed the need of switching to this new scheme and the possible domains where ECC can replace RSA, partially if not completely, in the near future.**

Keywords*— *public key cryptography; Elliptic Curves; RSA; cryptographic security; Electronic Mail security; trapdoor; Digital Signatures; weak keys; high security; forgery; public key; private key; Abelian Group

I. INTRODUCTION

With the increase in computation power [12] and the emergence of better and specialized algorithms, the conventional public key cryptographic schemes like RSA are doomed. Most of the specialized algorithms and heuristics, at present, can successfully break RSA in polynomial time until the key sizes are further increased [8][17][19]. Increase in key size is not a solution at present because that will also increase the computational load on the platform where this scheme is deployed. The increasing demand and usage of mobile devices rules out this solution.

Recently, there has been developments in a newly found cryptographic scheme based on polynomials, namely the Elliptic Curves. This scheme, in the cryptographic world is more popularly known as the Elliptic Curve Cryptography (ECC) [1][5]. It has recently been replacing RSA in almost all cryptographic domains, be it browser security, mobile security, bitcoins, e-mail security, etc. It is also currently the best alternative to the defects of RSA since it is able to solve the problem of cryptographic security at much lower key sizes [1][7].

With the key sizes used with RSA today, ECC cannot be broken in polynomial time except for certain curve vulnerabilities which could be exploited [8][9]. As always, weak parameters always make a theoretically strong cryptographic system weak, same is the case with ECC as well. However, with strong parameters, the scheme is very difficult to break. It also proves to be of the best use in mobile devices due to little power consumption, owing to the smaller key sizes that provide an equivalent security as the currently available schemes [1][7].

In this paper, I shall discuss the emergence of Elliptic Curve Cryptography as the new public cryptographic scheme, in the niche area of *"Electronic Mail Security using Pretty Good Privacy (PGP)"*. I shall talk about the alternatives to RSA based PGP, in terms of the signing algorithm, the encryption and decryption algorithm, and discuss about the Elliptic Curve Digital Signature Algorithm (ECDSA), used as part of PGP based on Elliptic Curves.

II. NEED FOR ELLIPTIC CURVE CRYPTOGRAPHY

- Elliptic curves offer same security as RSA but at lower key sizes. This reduces the resources needed in order to implement this cryptosystem [1][7][9], as a result it could be used in mobile devices with limited resources offering a good security.
- In most cryptosystems like mobile-cryptocurrency, RSA has been showing vulnerability due to smaller key sizes offering pretty less security and can be broken by advanced prime factorization algorithms [17][19]. ECC offers greater security than RSA for the same key size and there are less methods to break an ECC. The known methods are also probabilistic and fail for curves with strong parameters [1][8].

ISBN: 978-1-5090-1666-2/16/$31.00 ©2016 IEEE

- Generating a new key-pair in ECC (not for the first time) is significantly faster than generating a new key-pair in RSA-based schemes.
- ECC can be widely optimized, both in terms of hardware and software, if we stick to secure curves of a single type. The difficulty in breaking will not be affected if we use only one type of curves [8][10].
- RSA can verify the digital signatures faster than generating them. ECC can generate the digital signatures faster and sometimes at the same speed at which they are being verified at the other end. This is useful because there is no bottleneck in generation and transmission of digital signatures.
- Elliptic curve point formulae over prime fields can be accelerated by various mathematical techniques [5][11]. Moreover the ECC operations can also be extended to a parallel architecture and further be made faster [10]. Deriving a public key from a private key can be made faster.
- The trapdoor function used in Elliptic curves is way stronger than that used in RSA. Specialized algorithms like the Quadratic Sieve and the General Number Field Sieve were created to tackle the problem of prime factorization, on which RSA is based. These algorithms are faster and less computationally intensive than the naive approaches.

These factoring algorithms get more efficient as the size of the numbers being factored get larger. As the resources available to decrypt numbers increase, the size of the keys needs to grow even faster [17]. This is not a sustainable situation for mobile and low-powered devices. All this means is that RSA is not the ideal system for the future of cryptography. In an ideal Trapdoor Function, the easy way and the hard way get harder at the same rate with respect to the size of the numbers in question [1][17].

III. ELLIPTIC CURVE CRYPTOSYTEM

A. Definition and Terminologies

Elliptic Curves form an *Abelian Group* over the operation of addition denoted by '+' here. For more information regarding *Abelian groups* one can refer [2][5].

For elliptic curve cryptography, an operation over elliptic curves, called addition, is used. Multiplication is defined by repeated addition. For instance,

K×A = A + A + A + A + A + ……., K times, where the addition operation is performed over the elliptic curve. This forms the basis of the discrete logarithm problem for an elliptic curve and it is this property that is also referred to as a *trapdoor* [1][17].

It can be explained as follows: Cryptanalysis requires us to find the value of K, given A and K×A. This is computationally hard if we stick to repeated addition method and K is very large (which usually is, since we are talking about security). Before exploring how this is a difficult problem let's see the equation which defines the elliptic curve [1][5]:

$$y^2 + axy + by = x^3 + cx^2 + dx + e. \quad (1)$$

This curve has been defined over real numbers and may include infinitely many points. For feasibility, we define the elliptic curves over prime field to include a finite set of points. We define the curves over primes by simplifying the above relation:

$$y^2 \equiv (x^3 + ax + b) \bmod p \quad (2)$$

However, not all values of a and b can form suitable Elliptic curves. The Elliptic curves are groups if and only if the above cubic equation does not have a double root. This can be ensured if the discriminant of the above cubic equation is not equal to zero [5]. Using mathematical principles this can be ensured by the relation:

$$(4a^3 + 27b^2) \bmod p \neq 0 \quad (3)$$

B. Cryptosystem

With this new curve representation (modulo a prime field), we can take messages and represent them as points on the curve. Any message can be set as the x- coordinate, and the y-coordinate can be solved for, to get a point on the curve [5][17].

An elliptic curve cryptosystem can be defined by picking a maximum and standard prime number (Mersenne primes are generally chosen [8]), a curve equation and a public point on the curve. A private key is a number N_a generally sufficiently large, and a public key is the public point G multiplied with itself, N_a times giving $N_a \times G$. Knowing the public key N_aG, the function that computes the private key N_a, is called the elliptic curve discrete logarithm function, in our case the Trapdoor Function [1][17].

The elliptic curve discrete logarithm is the underlying hard problem of elliptic curve cryptography analogous to prime factorization in RSA. Besides the naïve approach which cannot complete in polynomial time, mathematicians still cannot crack this problem, unlike the conventional RSA which has already been cracked by specialized factoring algorithms in polynomial time. This means that for numbers of the same size, solving elliptic curve discrete logarithms is significantly harder than factoring, which makes elliptic curve cryptosystems cryptographically more secure than their RSA counterparts [17].

To visualize how much harder it is to break, *Lenstra* in [7] recently introduced the concept of "Global Security." We can compute how much energy is needed to break a cryptographic algorithm or the trapdoor, and compare that with how much water that same energy could boil, a *cryptographic carbon footprint* indeed [17]. Using this measure, breaking a 228-bit RSA key requires lesser energy than it takes to boil a teaspoon of water. For the same key size, elliptic curve cryptosystems require as much energy as is required to boil all of the water on Earth. For this level of security with RSA, we need a key with 2,380-bits [1][7].

With ECC, we can use smaller keys to get the same levels of security. Small keys save power on mobile devices, so are desirable. In RSA, with larger key sizes even the forward computation of public keys can take a considerably higher time on mobile devices, as well as consume more power. We can continue to use RSA but at the cost of slower cryptographic performance, as we have to increase the current key size for desirable security. ECC comes in handy here: high security with short, fast keys [17].

IV. ECC IN PRETTY GOOD PRIVACY (PGP)

A. Emergence of ECC

After a slow start, elliptic curve based algorithms are gaining popularity and the pace of adoption is accelerating. Elliptic curve cryptosystems due to their high security at smaller key sizes are used in crucial security areas like nuclear defense and sensitive government communications. This cryptosystem has also started gaining popularity in the modern realm of financial commerce involving bitcoins. Simple encryption of sensitive DNA information is also being done using this cryptosystem. All in all, it has become a major part of the Secure Socket Layer in the Network protocol models and plays an important role in Transport Layer Security. [13][17].

Any recent enough version of Chrome or Firefox browser supports ECC. If we try to access any HTTPS website, we can confirm the usage of ECC in the security certificate of that site. The relevant portion of this security certificate is ECDHE_RSA. ECDHE stands for *Elliptic Curve Diffie Hellman Ephemeral* and is a key exchange mechanism based on elliptic curves. Some websites even use Elliptic Curves for digital signatures, in that case we seen the acronym ECDHE_ECDSA in the website's security certificate.

Using elliptic curve cryptography saves time, power and computational resources for both the server and the browser helping us make the web both faster and more secure. The utilization of ECC in SSL (Secure Socket Layer) during the handshake protocol can be reviewed in [13]. We hereby discuss how ECC can be used in an Electronic Mail Service, Pretty Good Privacy (PGP) which is more common among Internet users and briefly discuss the ECDSA and the ECC alternative of Elgamal cryptography [15][16] in PGP.

B. Pretty Good Privacy (PGP)

A detailed description about PGP can be found in [4]. PGP provides a confidentiality and authentication service that can be used for electronic mail and file storage applications. In the conventional PGP, RSA and DSS (Digital Signature Standard) is used for the purpose of authentication by digital signatures [4]. We discuss the usage of ECC for the purpose of digital signatures in the form of ECDSA in the forthcoming section and show how it can completely replace RSA in this field.

Using ECC offers many benefits, firstly a comparable security at lesser key size which is desirable for power and resource saving in most of the mobile applications. Moreover, the speed is not compromised, instead the mail service becomes faster than before because of the lower key size and lesser computations involved. RSA based schemes cannot be accelerated in any sense except for the fast exponentiation that they provide. On the other hand ECC scalar multiplication operations can be optimized and accelerated by a variety of means and mathematical techniques [9][10][11].

Another advantage of using smaller keys is that the size of the security certificates involved is also reduced. Since the certificates are part of the browser cache and are also transmitted across websites for verification, this improves the browser performance significantly [18].

An important point to note here is that ECDSA algorithm requires some unpredictable data as input for every signature. If the source of randomness is predictable to an attacker, then the private key can be figured out [18]. However, if the input is random enough and the source is unpredictable, then breaking this signature scheme to impersonate someone else, cannot be easily achieved.

PGP involves encryption of the symmetric key itself which will later be used as a session key for exchanging mails. This encryption of the symmetric key is done through public key cryptography, since the symmetric key needs to be delivered to the other party without letting an eavesdropper know. Asymmetric encryption is suitable here, since any attacker in this case will not be able to decrypt the session key in polynomial time, as it can only be decrypted by the receiver's private key not known to an attacker.

In the conventional PGP, the public key cryptography used is RSA, [4] for encrypting the symmetric key. In our discussion, we show how ECC can be used for the same [15][16], offering benefits, as discussed above. Using the illustration of the conventional scheme in *Fig. 1*, we continue our discussion.

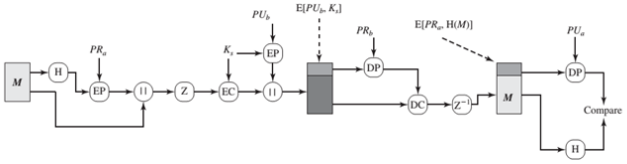

Fig. 1. Illustration of Pretty Good Privacy (PGP) [4]

Using the terminology as described in *Fig. 1*, we discuss the meaning of various terms in context of ECC. This scheme provides both confidentiality and authentication [4]:

1) H: Hash function (SHA-1) used on the message. This remains unchanged in our version of PGP, from ECC perspective.

2) EP: Public Key Encryption (for digital signature using private key) and PR_a is the private key of A used for digital signature. This is replaced by the ECDSA scheme [6] for generating digital signatures using N_a as the new private key, two values r and s (serve as signatures) are generated

after this step in our case both of which are concatenated with the message to be later used for verification of the signature. This is discussed in the next section. Conventional scheme used DSS here.

3) Z: Compression using the ZIP format (also unchanged in context of ECC).

4) EC: Symmetric Key encryption using a one-time key K_s (symmetric key) which is generated randomly by the *PGP_Random_Number_Generator* [3] (using the properties of the message itself via CAST-128 algorithm), the encryption uses CAST-128 / IDEA [3][4] which also remains unchanged in context of this discussion.

5) EP: Public Key Encryption using B's public key PU_b. In our case, this will be replaced by the public key generated by scalar multiplication on the elliptic curve which is $N_b \times G$, where N_b is the private key of B. The encryption is analogous to Elgamal scheme [15] and will generate two ciphertexts from the corresponding plaintext (in our case, the symmetric key K_s is the plaintext), and concatenate both of these ciphertexts to the encrypted message (encrypted under K_s).

6) R_{64}: This has not been showed in the diagram but is also present in the conventional scheme. The encrypted message is converted into ASCII printable characters by a radix-64 mapping. Since there are 64 printable characters in English, possible 256 characters are mapped into this 64 bit value (by dividing three 8-bit quantities into four 6-bit quantities). This increases the encrypted message size but since we had compressed the original message, the net result is compression only (ZIP is optimized for compression). The first task that the receiver of the message does is to apply an inverse of this function before proceeding to decrypt the key.

7) DP: Public Key Decryption using B's private key PR_b. In context of ECC, this is the private key of B represented as N_b. The encrypted key K_s, which was used for encrypting the message is now decrypted using ECC based decryption from the two ciphertexts that were concatenated with the message. We hence obtain the key K_s and use it again for symmetric key decryption (DC) of the encrypted message.

8) Z^{-1}: This is the decompression operation which outputs the original message along with the signed hash code (encrypted under A's private key). In context of ECC, it outputs the original message concatenated with values r and s that are used to verify whether the signature is correct and the message is authentic.

The verification is discussed in the following section (where we discuss the ECDSA algorithm) using only the values r and s (signatures) and the hash of the message which is recomputed. In the conventional case, we require the message's hash, its signature (encrypted hash) as well as the public key of A (PU_a) to decrypt the signature and verify whether it is equal to the recomputed hash or not. In context of ECC, knowledge of an extra term, i.e. the public key of the sender ($Na \times G$), in case of signature verification is not required.

An important point to note here is that the sender and the receiver may also use public / private key rings for choosing specific public and private keys for a single message exchange [4]. The corresponding Key IDs by which the rings are indexed also need to be shared securely. A detailed description on how to use these rings and how they are constructed can be found in [4].

We now move on to discuss the Elliptic Curve Digital Signature Algorithm (ECDSA) used for signing the hash of the message. Since the hash is smaller in size, it is optimal to sign the hash than the entire message (entire message signing can also be done but is cumbersome). Moreover, it will be difficult to represent an entire message as a point on the elliptic curve but easier to represent the hash on the elliptic curve as an affine or a projective coordinate point [5]. We also discuss the process of encrypting the symmetric key K_s using Elliptic curve based encryption that is analogous to the Elgamal Encryption Procedure [15].

C. ECDSA

i. Generating signature pair (r, s) from the hash of the message (sender's end): [6]

Let $e = H(m)$ be the hash of the message m.

Let z be L_n leftmost bits of e; L_n is bit length of group order n in case we are using Elliptic Curves over Finite Galios Field (2^n) else it can be chosen based on the maximum prime number in case of Elliptic Curves over primes, z is converted to an integer from the bit string obtained.

The signer arbitrarily choose a public point G on the elliptic curve and a public parameter n to be the smallest positive integer n such that $n \times G = O$ (focus of the elliptic curve)

He hence chooses a private key N_a such that it is less than n and generates a public key Q_a from it as $N_a \times G$.

Next the signer chooses a cryptographically secure random integer (private) less than n as k and calculates $k \times G$ with the affine coordinates of $k \times G$ being (x_1 , y_1).

The value of r, the first signature is calculated as

$$r = x_1 \bmod n \quad (4)$$

If r comes out to be zero then k is again chosen and the process is repeated till r is non zero.

Next the signer calculates the value of second signature term

$$s = k^{-1}(z + rN_a) \quad (5)$$

In both the Elliptic curves over primes and the Galios Field (2^n), the multiplicative inverses are represented as integers (in Elliptic Curves over primes we can use Extended Euclidean Algorithm to calculate multiplicative inverses [2]).

The signature is hence the pair **(**r, **s).**

ii. Verifying signature pair (r, s) on receiver's end: [5]

Verify whether r and s are between 1 and n-1 or not. If they are not the signature is invalid. (n, G, $Q_a = N_a \times G$ are all public parameters).

Re-compute e' = H (m'), the hash of the message m'.

Again, let z' be L_n leftmost bits of e'.

Next the verifier calculates

$$w = s^{-1} \bmod n \quad (6)$$

$$u_1 = z'w \bmod n \quad (7)$$

$$u_2 = rw \bmod n. \quad (8)$$

Finally, the verifier calculates a point (x_1', y1') on the Elliptic curve as $u_1G + u_2Q_a$ (both G and Q_a were public parameters).

He then verifies whether $x_1' \equiv r \bmod n$ or not. The signature is valid if the relation holds else it is invalid. Proof can be done by simply substituting the required values.

We have important points to note in the ECDSA. Firstly, the signature (r, s) cannot be forged because both depend on the private key of the sender and a number chosen by the sender himself (k). However, there exists a vulnerability in this signature scheme. First of all the number k should be unpredictable and generated by a good random source else if k can be predicted by brute force, the private key N_a used in generating signature can easily be compromised as:

$$N_a = (sk - z) / r \quad (9)$$

(r, s, z are known) and the signature can then be forged.

Furthermore, same values of k should not be used while signing two different messages m and m'. In this case also a differential attack could be made and the value of k could directly be found. If same k is used for two messages, the r values will also be same (see the scheme above), only the s values of the messages differ because they are dependent on z. The attacker can hence calculate $s - s' = k^{-1}(z - z')$. From this relation k can simply be calculated as:

$$k = (z - z') / (s - s') \quad (10)$$

This done, the value of the private key can again be calculated using (9). Hence for each different message that is signed ECDSA must ensure that a different value of k is generated meaning that k must act as a nonce; when all possible values of k have been used / generated, we can drop the current private key and choose another private key. Now, if the values are reused the hacker cannot guess the random number k because he does not know the difference between the current private key and the previous one. Additionally, the new private key must have a great difference with the previous one so that it cannot be easily guessed by the hacker to calculate k.

D. Encrypting Symmetric Key K_s by ECC

a. *Encryption*:

This technique is analogous to the Elgamal cryptosystem [15]. First of all K_s is mapped to a point on the Elliptic Curve. This can simply be done by first converting K_s into integer (from its bit version) and then setting the x-coordinate as K_s mod p (p is the maximum prime number used in Elliptic Curves over primes) and calculating the integral y- coordinate satisfying the equation of the curve. Let's call this mapped point as P_m.

A public point G on the elliptic curve and a public parameter n, the smallest positive integer n such that $n \times G = O$ (focus of the elliptic curve) was already chosen. The receiver's private key N_b (less than n) was chosen and the public key $Q_b = N_b \times G$ was generated [1].

Next the sender who is encrypting the symmetric key P_m chooses a cryptographically secure random integer (private) less than n as k and calculates $k \times G$ (scalar multiplication over prime Elliptic Curve). This is a point on the Elliptic Curve and forms the first ciphertext C_1 [1]. Hence $C_1 = k \times G$.

Next the sender calculates the second ciphertext $C_2 = P_m + k \times Q_b$. The addition as well as the scalar multiplication are performed over the prime Elliptic Curve used). C_2 is also a point on the Elliptic Curve. Both these ciphertexts are concatenated with the symmetrically encrypted message (using key K_s).

b. *Decryption*:

The receiver has the ciphertexts C_1 and C_2 and recovers symmetric key K_s from them using following procedure:

Receiver simply calculates $P_m = C_2 - (N_b \times C_1)$, where N_b is the private key of the receiver. Subtraction is defined on the Elliptic Curve as adding the negation of a point; negation of any point on the Elliptic Curve is obtained simply by negating the ordinate [1][5]. The negative values are then adjusted with the modulo prime operation on the Elliptic Curve to yield a positive value.

The x-coordinate of this point P_m is the key K_s (integer form) with which the symmetric encryption of the compressed message was carried out. K_s is converted into bit notation for facilitating symmetric decryption of the compressed message.

A caution to be taken here is that we must choose the maximum prime number for the Elliptic Curve over primes to

be greater than all possible values of K_s, else P_m cannot be uniquely calculated for every K_s making the inverse mapping ambiguous.

CONCLUSION

From the above discussion, it is clear that ECC can very easily replace RSA in the domain of Electronic mail security and even prove to be better and faster than that. Certain flexible properties like the hashing or the compression algorithms that were also part of PGP were unchanged since they have no role in public cryptography and can be dealt with separately as per the best scheme that is currently available.

We can justify that our approach involving ECC works better in the area of Electronic Mail Security because it has been experimentally proven that ECC can offer same levels of security as RSA, but at much lower key sizes. This is advantageous both in mobile devices as well as personal computers as the performance is significantly increased. This is desirable as well, since Electronic Mail Security is the most basic level of security that an Internet user can think of.

Moreover, RSA has already lived up to its reputation and the security realm needs a major change now; there are special algorithms and heuristics which can factor a large number in polynomial time easily, defeating the hard problem underlying RSA. The only solution for good security using RSA is to increase the key size. This has two major disadvantages. Firstly, an increased key size will reduce the performance for obvious reasons. Secondly, while it becomes harder now to calculate the public key, the difficulty in guessing the private key does not increase by the same level as is the case in an ideal Trapdoor. In fact, there are algorithms out there which perform better and factorize quickly, if the size of the number in question is larger.

ECC can easily overcome these difficulties offering similar security at lower key sizes. Also, for every scheme involving RSA, be it simple encryption-decryption or digital signatures or key exchanges there is an alternative and analogous ECC scheme available enabling Elliptic Curves to replace RSA in all the fields of security. Moreover, the known attacks on Elliptic Curves are probabilistic in nature and some side-channel attacks which can easily be avoided by choosing strong and standard curve parameters.

Still there are many domains out there where this new scheme has not yet reached, mostly because of the loss of confidence in this new scheme. This has mainly been due to the fact that ECC is not as easy to understand as RSA and requires a lot of mathematics, hence making it difficult to be adopted at a much larger scale and by common individuals and organizations. Elliptic Curves, as a mathematical tool, have recently been used to prove an age-old hypothesis, a million dollar problem, T*he Fermat's Last Theorem*.

ECC can be incorporated in the area of E-voting, multiparty computation problems, etc., but it is only possible when people start to realize that the conventional cryptography can no longer support them and a new scheme is the need of the hour. Recent attacks like *HeartBleed* (https://*heartbleed.com*) have given a lesson to the Internet community. ECC has already been adopted in *OpenPGP* as part of the *RFC 6637* and has been analyzed on various parameters. Hence, it is not wrong to say now that ECC will be the future of public key cryptography.

REFERENCES

[1] William Stallings, "Cryptography and Network security, principles and practice," 5th ed., 2011, pp. 308-320.

[2] William Stallings, "Cryptography and Network security, principles and practice," 5th ed., 2011, pp. 108-141.

[3] William Stallings, "Cryptography and Network security, principles and practice," 5th ed., 2011, Appendix 15C, pp. 480-482.

[4] William Stallings, "Cryptography and Network security, principles and practice," 5th ed., 2011, pp. 568-587.

[5] Lawrence C. Washington, "Elliptic Curves, number theory and Cryptography," 2nd ed., 2008, pp. 9-20.

[6] Lawrence C. Washington, "Elliptic Curves, number theory and Cryptography," 2nd ed., 2008, pp. 192-193.

[7] Arjen K. Lenstra, Thorsten Kleinjung and Emmanuel Thome, "Universal security from bits and mips to pools, lakes – and beyond," Number theory and Cryptography, Springer, LNCS 8260.

[8] Joppe W. Bos et al., "Elliptic Curve Cryptography in practice," Financial Cryptography and data security, 18th international conference FC 2014, Christ Church, Barabados, March 3-7, 2014, Springer, LNCS 8437.

[9] M.Prabu and Dr.R.Shanmugalakshmi, "A comparative and overview analysis of Elliptic Curve Cryptography over finite fields," 2009 International Conference on Information and Multimedia Technology,2009 IEEE, DOI 10.1109/ICIMT.2009.66.

[10] Qingwei Li, Zhongfeng Wang, and Xingcheng Liu, "Fast point operation architecture for Elliptic Curve Cryptography," IEEE Asia Pacific Conference on Circuits and Systems, 2008, APCCAS 2008, DOI 10.1109/APCCAS.2008.4745991.

[11] M. Bednara, M. Daldrup,J. Teich, and J. von zur Gathen,J. Shokrollahi, "Tradeoff analysis of FPGA based Elliptic Curve Cryptography," Proceedings, 2002 IEEE international Symposium on circuits and systems.

[12] Gordon E. Moore, "Cramming more components onto integrated circuits," Electronics, vol. 38, no. 8, April 19, 1965.

[13] Vipul Gupta et al., "Sizzle: A standards-based end-to-end security architecture for the embedded internet", pp. 7-11, SMLI TR-2005-145, June, 2005.

[14] Guicheng shen, Xuefeng Zheng, "Research on Implementation of Elliptic Curve Cryptosystem in E-Commerce", International Symposium on Electronic Commerce and Security,2008.

[15] Sarwono Sutikno, Andy Surya, Ronny Effendi, "An Implementation of ElGamal Elliptic Curves Cryptosystems",Integrated System Laboratory,Bandung Institute of Technology, 1998.

[16] Zengqiang Wu, Di Su, Gang Ding, "ElGamal algorithm for encryption of data transmission", 2014 International Conference on Mechatronics and Control (ICMC), IEEE, 3-5 July 2014.

[17] N. Sullivan, "A (Relatively Easy To Understand) Primer on Elliptic Curve Cryptography", CloudFlare, 2013. [Online]. Available: https://blog.cloudflare.com/a-relatively-easy-to-understand-primer-onelliptic-curve-cryptography/. [Accessed: 23- Dec- 2015].

[18] N. Sullivan, "ECDSA: The digital signature algorithm of a better internet", CloudFlare, 2013. [Online]. Available: https://blog.cloudflare.com/ecdsa-the-digital-signature-algorithm-of-a-better-internet/. [Accessed: 23- Dec- 2015].

[19] Andrea Pellegrini, Valeria Bertacco and Todd Austin, "Fault-based attack of RSA authentication", University of Michigan, {apellegrini,valeria,austin}@umich.edu.