Knapsack Based ECC Encryption and Decryption

R. Rajaram Ramasamy, M. Amutha Prabakar, M. Indra Devi, and M. Suguna
(Corresponding author: R. Rajaram Ramasamy)

Department of Computer Science and Engineering, Thiayagarajar College of Engineering
Thiruparankundram, Madurai, Tamil Nadu, India, 625 015, India (Email: rrajaram@tce.edu)
(Received Mar. 11, 2008; revised and accepted Aug. 22, 2008)

Abstract

Elliptic Curve Cryptography provides a secure means of exchanging keys among communicating hosts using the Diffie Hellman Key Exchange algorithm. Encryption and Decryption of texts and messages have also been attempted. This paper presents the implementation of ECC by first transforming the message into an affine point on the EC, and then applying the knapsack algorithm on ECC encrypted message over the finite field GF(p). In ECC we normally start with an affine point called P_m(x,y). This point lies on the elliptic curve. In this paper we have illustrated encryption/decryption involving the ASCII value of the characters constituting the message, and then subjecting it to the knapsack algorithm. We compare our proposed algorithm with RSA algorithm and show that our algorithm is better due to the high degree of sophistication and complexity involved. It is almost infeasible to attempt a brute force attack. Moreover only one parameter, namely the Knapsack vector ai alone needs to be kept secret. On the contrary in RSA, three parameters such as the modulus n, its factors p and q need to be kept secret.

Keywords: Discrete logarithm, elliptic curve cryptography (ECC), knapsack algorithm, public key cryptography, RSA algorithm

1 Introduction

Most of the products and standards that use public-key cryptography for encryption and digital signatures use RSA today. Recently, Elliptic Curve Cryptography has begun to challenge RSA. The principal attraction of ECC, compared to RSA, is that it appears to offer better security for a smaller key size, thereby reducing processing overhead. Elliptic curve cryptography makes use of elliptic curves in which the variables and coefficients are all restricted to elements of a finite field. In ECC we normally start with an affine point called P_m(x,y). These points maybe the Base point (G) itself or some other point closer to the Base point. Base point implies it has the smallest x,y co-ordinates, which satisfy the EC. A character in a message is first transformed into an affine point of the elliptic curve by using it as a multiplier of P_m. That is, if the ASCII value of a character is A, then we determine P'_m=A(P_m). This is one step towards introducing sophistication and complexity in the encryption process. The newly evaluated P'_m is a point on the EC, determined by applying the addition and doubling strategy of ECC technique. Then as per ECC algorithm, P'_m is added with kP_B, where k is randomly chosen secret integer and P_B is the public key of user B, to yield (P'_m+kP_B). This now constitutes second part of the encrypted version of the message. The other part, namely, kG, which is the product of the secret integer and the Base point, constitutes the first part. Thus the encrypted message is now made up of two sets of coordinates, namely, (kG, P'_m+ kP_B). In this paper we have assigned kG=(x_1,y_1) and (P'_m+kP_B)=(x_2,y_2). Not satisfied with the complexity involved in determining the encryption, we wish to introduce further complexity by applying the Knapsack concept to the encrypted version. The whole idea behind these rigorous exercises is to make decryption totally impossible, even if the Base Point G, secret integer k, the affine Point P_m are known to the crypt analyst. Now to recover the information from the encrypted version, first the knapsack process has to be reversed. Then we apply the decryption process of ECC, by applying the private key of recipient (n_B) on the first element (kG). This is subtracted from the second element to recover P'_m. Lastly by using the discrete logarithm concept, it is possible to evaluate the ASCII value and thereby recover the plaintext. The encrypted contents may be stored in CD/DVD or transmitted over the Net to a beneficiary. This promises to afford maximum security from intruders and hackers.

The originality of this paper rests on the following points. 1) Transforming the ASCII value of a character of the message into an affine point on the EC. 2) Knapsack algorithm introduces further non-linearity in the encryption. Other points such as applying ECC for encryption/Decryption, invoking the discrete logarithm concept

to recover the ASCII value are already existing and well documented. As far as the authors knowledge go no such attempt seems to have been presented so far. For the sake of comparison of the knapsack based ECC encryption/decryption, another public key algorithm, namely RSA, is used to encrypt/decrypt the same message. Unlike the ECC procedure, this yields only one integer for each character of the message. The time and space implications for both the schemes are discussed and analyzed. The paper justifies that despite the harsher requirements of time and space for the ECC methods, it is far superior due to the resistance it offers to any brute force attack.

Some recent works on application of ECC are cited here.Aydos et al. [1] Discusses the results of implementation of ECC over the field GF(p) on an 80 MHz, 32 bit RAM microprocessor. Kristin et al. [8] provides an overview of ECC for wireless security. It focuses on the performance advantages in the wireless environment by using ECC instead of the traditional RSA cryptosystem. Ray et al. [3] explains the design of a generator, which automatically produces a customized ECC hardware that meets user-defined requirements. Cilardo et al. [4] explains the engineering of ECC as a complex interdisciplinary research field encompassing such fields as mathematics, computer science and electrical engineering. McIvor et al. [10] introduces a novel hardware architecture for ECC over GF(p). Chen et al. [2] presents a high performance EC cryptographic process for general curves over GF(p). The standard specifications for public key cryptography are defined in [13].

The idea for extending knapsack algorithm to encryption/decryption was derived by Diffie [5]. The ECC concept is very well documented and illustrated by Williams Stallings [14]. In [15], Access control in sensor networks is used to authorize and grant users the right to access the network and data collected by sensors. This paper describes a public key implementation of access control in a sensor network. Kevin et al. [6], presents a brute-force attack on ECC implemented on UC Berkley's Tiny OS operating system for wireless sensor networks. The attack exploits the short period of the pseudorandom number generators used by cryptosystem to generate private keys. The paper, Moon [11] proposed a more efficient and novel approach of a scalar point multiplication method than existing double and add by applying redundant recoding, which originates from radix-4 Booths algorithm. Hedabou [7], proposes a heuristic analysis on the security of Joye and Tymen's technique. The paper Lee et al. [9], proposes 3 algorithms to perform scalar multiplication on EC defined over higher characteristic finite fields such as OEA (Optimal Extension Field). Yongliang et al. [16] shows that Aydos et al.'s protocol is vulnerable to man-in the-middle attack from any attacker but not restricted on the inside attacker. They proposed a novel ECC based wireless authentication protocol.

Shi et al. [12], investigates whether some architectural parameters such as word size may affect the choice of algorithms when implementing ECC with software. They have identified a set of algorithms for ECC implementation for low-end processor.

2 Proposed Method Description

The Weiestrass equation defining an elliptic curve over GF(p), for $q > 3$, is as follows:

$$E : y^2 = x^3 + ax + b, \tag{1}$$

where x, y are elements of GF(p), and a, b are integer modulo p, satisfying

$$4a^3 + 27b^2 \neq 0 \bmod p. \tag{2}$$

Here p is known as modular prime integer. An elliptic curve E over GF(p) consist of the solutions (x, y) defined by Equations (1) and (2), along with an additional element called O, which is the point of EC at infinity. The set of points (x, y) are said to be affine coordinate point representation.

The basic Elliptic curve operations are point addition and point doubling. Elliptic curve cryptographic primitives [13] require scalar point multiplication. Say, given a point $P(x, y)$ on an EC, one needs to compute kP, where k is a positive integer. This is achieved by a series of doubling and addition of P. Say, given $k = 386$, entails the following sequence of operations (refer to Table 1).

$$P, 2P, 3P, 6P, 12P, 24P, 48P, 96P, 192P, 193P, 386P.$$

Let us start with P(x_P, y_P). To determine $2P$, P is doubled. This should be an affine point on EC. Use the following equation, which is a tangent to the curve at point P.

$$S = [(3x_P^2 + a)/2y_P] \bmod p.$$

Then $2P$ has affine coordinates x_R, y_R given by

$$\begin{aligned} x_R &= (S^2 - 2x_P) \bmod p, \\ y_R &= [S(x_P - x_R) - y_P] \bmod p. \end{aligned}$$

Now to determine $3P$, we use addition of points P and $2P$, treating $2P = Q$. Here P has coordinates (x_P, y_P) and $Q = 2P$ has coordinates (x_Q, y_Q). Then

$$\begin{aligned} x_R &= (S^2 - x_P - x_Q) \bmod p, \\ y_R &= (S(x_P - x_R) - y_P] \bmod p. \end{aligned}$$

Therefore we apply doubling and addition depending on a sequence of operations determined for k. Every point x_R, y_R evaluated by doubling or addition is an affine point (points on the Elliptic Curve).

The proposed algorithm is shown in Algorithm 1.

Table 1: A series of doubling and addition of P

P	2P	3P	6P	12P	24P	48P	96P	192P	193P	386P
-	Doubling	Addition	Doubling	Doubling	Doubling	Doubling	Doubling	Doubling	Addition	Doubling

Algorithm 1 KnapsackBasedECC (U_{Alice},U_{Bob}, P_m)

1: Begin
2: Initiate the connection with U_{Alice} and U_{Bob}
3: **if** (U_{Alice} and U_{Bob} is legitimate user) **then**
4: U_{Bob} = {P_{Bob}, n_{Bob}}//Key pair for U_{Bob}
5: U_{Alice} = {P_{Alice}, n_{Alice}}//Key pair for U_{Alice}
6: U_{Bob} = Send (P_{Bob}, U_{Alice});//Send the Public key of U_{Bob} to U_{Alice}
7: U_{Alice} = Send (P_{Alice}, U_{Bob});//Send the Public key of U_{Alice} to U_{Bob}
8: U_{Bob} = Send (U_{Alice}, Public information);
9: U_{Alice} = Send (U_{Bob}, Public information);
10: **end if**
11: //Encryption – *Alice* encrypt the message and send it to *Bob*
12: **if** (Encryption = TRUE) **then**
13: Calculate the P_m': $P_m' = G.P_m$, where P_m is the ASCII value of Plain Text and G is the base point of EC
14: Calculate the kP_{Bob} value
15: Calculate the $P_m' + kP_{Bob} = (x_2, y_2)$
16: Calculate the $kG = (x_1, y_1)$
17: $S[x_1]$= Knapsack value (x_1);
18: $S[y_1]$= Knapsack value (y_1);
19: $S[x_2]$= Knapsack value (x_2);
20: $S[y_2]$= Knapsack value (y_2);
21: Send (U_{Bob}, C_m=(($S[x_1]$, $S[y_1]$), ($S[x_2]$, $S[y_2]$)));
22: **end if**
23: //Decryption – *Bob* decrypt the message from*Alice*
24: **if** (Decryption = TRUE) **then**
25: x_1 = Inverse Knapsack value ($S[x_1]$);
26: y_1 = Inverse Knapsack value ($S[y_1]$);
27: x_2 = Inverse Knapsack value ($S[x_2]$);
28: y_2 = Inverse Knapsack value ($S[y_2]$);
29: $kG = (x_1, y_1)$;
30: $P_m' + kP_{Bob} = (x_2, y_2)$;
31: Calculate $n_{Bob}kG = n_{Bob}(x_1, y_1)$;
32: Calculate $P_m' = P_m' + kP_{Bob} - n_{Bob}\ kG$;
33: Calculate the P_m value from P_m' using discrete logarithm
34: **end if**
35: End//End Algorithm KnapsackBasedECC

3 Implementation Details Of Our Proposed Algorithm

Once the defining EC is know, we can select a base point called G. G has $[x, y]$ coordinates which satisfy the equation $y^2 = x^3 + ax + b$. The Base point has the smallest x, y values which satisfy the EC.

The ECC method requires that we select a random integer $k(k < p)$, which needs to be kept secret. Then kG is evaluated, by a series of additions and doublings, as discussed above. For purpose of this discussion we shall call the source as host A, and the destination as host B. We select the private key of the host B, called n_B. k and n_B can be generated by random number generators to give credibility. That would be digressing away from the main discussion. Hence we make suitable assumptions for these two parameters. The public key of B is evaluated by

$$P_B = n_B G. \tag{3}$$

Suppose A wants to encrypt and transmit a character to B, he does the following. Assume that host A wants to transmit the character 'S'. Then the ASCII value of the character 'S' is used to modify P_m as follows: P_m'= SP_m. P_m we said is an affine point. This is selected different from the Base point G, so as to preserve their individual identities. P_m' is a point on the EC. The coordinates of the P_m' should fit into the EC. This transformation is done for two purposes. First the single valued ASCII is transformed into a x,y co-ordinate of the EC. Second it is completely camouflaged from the would-be hacker. This is actually intended to introduce some level of complexity even before the message is encrypted according to ECC.

As the next step of ECC, we need to evaluate kP_B, here P_B is a public key of user B. Determining this product involves a series of doubling and additions, depending on the value of k. For a quick convergence of the result, we should plan for optimal number of doubles and additions. The encrypted message is derived by adding P_m' with kP_B, that is, P_m'+kP_B. This yields a set of x_2, y_2 coordinates. Then kG is included as the first element of the encrypted version. kG is another set of x_1, y_1 coordinates. Hence the entire encrypted version for purposes of storing or transmission consists of two sets of coordinates as follows:

$$\begin{aligned} C_m &= (kG, P_m' + kP_B) \\ kG &\rightarrow x_1, y_1 \\ P_m' + kP_B &\rightarrow x_2, y_2. \end{aligned}$$

Thus far the modified plaintext has been encrypted by application of the ECC method. The modification of the plaintext in conjunction with Pm is a new innovation of this paper. However the authors have gone a step further, to make the encryption more secure, by proposing

the extension of the knapsack procedure. As far as our knowledge goes, this is perhaps the first attempt at applying the Knapsack algorithm to ECC encrypted message. This introduces thorough diffusion and confusion to shatter any attempt at brute force attacks.

Knapsack requires that we generate a series of vectors called a_i. There are several ways of generating these vectors. For the sake of illustration we shall take the first value as 1, and subsequent values as multiples of n Say

$$a_i = 1, n, n_2, n_3, \cdots, n_m 1 \leq i \leq m.$$

Here, n may be assumed as some random integer less than 10, or computed involving the p and k integers. Here p is a prime integer used in the modular arithmetic, k is the secret integer and m is the length of the binary bit string.

Next let us explore how the co-ordinates of the encrypted message are subjected to Knapsack process. Say, x_i, is one of the coordinate points, which can be represented in its binary form as:

$$x_i = b_1, b_2, \cdots, b_m 1 \leq i \leq m.$$

As per the knapsack algorithm we calculate a cumulative sum S[x_1],

$$S[x_1] = \sum_{i=1}^{m} a_i x_i.$$

In the final encrypted version the co-ordinate x_i is replaced by its equivalent S[x_1]. Similarly other coordinates like y_1, x_2, y_2 are transformed by the knapsack algorithm, so that the encrypted message shall now be represented as

$$C_m = ((\mathrm{S}[x_1], \mathrm{S}[y_1]), (\mathrm{S}[x_2], \mathrm{S}[y_2])).$$

Recall that this two pairs of integers represent just one character in a message. Depending on the number of characters in the message, there will be as many such pairs of integers. This is either stored in archival device like CD/DVD or transmitted to a beneficiary through the Net.

The recipient B has all relevant information for reversing the knapsack procedure and to recover bit pattern of the coordinates. For example B knows the ai series, his own secret key n_B, the base point G, a, b, p values of the EC. B receives the encrypted message, C_m=((S[x_1], S[y_1]), (S[x_2], S[y_2])). Let us discuss how to reverse the Knapsack process, by taking one example. Consider

$$S[x_1] = \sum_{i=1}^{m} a_i x_i,$$

which is the knapsack representation of x_1. The x_1 value is recovered in an iterative fashion as follows:

$$S[x_1] - n^m.$$

If this value is positive i.e., S[x_1] - $n^m > 0$, then a binary bit 1 is assigned at the $(m)^{th}$ position. The current value is S[x_1] = S[x_1] - n^m. If however the value is negative, then a 0 bit is assigned and the S[x_1] remains unchanged.

Now subtract n^{m-1} from the current S[x_1]. Depending upon whether it is +ve or -ve, assign 1 or 0 at the relevant bit position. Continue this subtraction until the a_i series is exhausted. This will recover the binary bit pattern of x_1. These procedures are repeated for y_1, x_2, and y_2.

Recall that kG is represented by x_1, y_1, and $P'_m + kP_B$ is represented by x_2, y_2. In order to pull out P'_m from $P'_m + kP_B$, B applies his secret key n_B and multiplies kG so that,

$$n_B kG = kP_B.$$

Subtract this from $P'_m + kP_B$, to get P'_m as follows:

$$P'_m = P'_m + kP_B - n_B kG.$$

This subtraction is another ECC procedure involving doubling and addition. But the only difference is that the negative term will have its y co-ordinate preceded by a minus sign. With this subtle change in mind, the expression of determining the slope, new values of x_R, y_R are the same. Wherever y figures, it is substituted as -y. This will yield P'_m.

Using the discrete logarithm concept the ASCII value of 'S' can be retrieved as follows, P'_m= S P_m.

4 Implementation Of The Proposed Algorithm

The Elliptic Curve is

$$y^2 \bmod 487 = (x^3 - 5x + 25) \bmod 487.$$

The base point G is selected as (0, 5). Base point implies that it has the smallest x, y co-ordinates which satisfy the EC. P_m is another affine point, which is picked out of a series of affine points evaluated for the given EC. We could have retained G itself for P_m. However for the purpose of individual identity, we choose Pm to be different from G. Let P_m =(1,316). The choice of P_m is itself an exercise involving meticulous application of the ECC process on the given EC. Further we need to generate the secret integer k, and the private key n_B of the recipient B. We have at our disposal a series of random number generators. But that would be digressing from the main path of thought. Hence we shall assume that $k = 225$, and $n_B = 277$.

Plaintext is "S", whose ASCII value is 83. Therefore,

$$\begin{aligned} P_B &= n_B G = 277(0,5) = (260,48) \\ P'_m &= 83(1,316) = (475,199) \\ kP_B &= 225(260,48) = (212,151) \\ P'_m + kP_B &= (475,199) + (212,151) = (51,58) \\ kG &= 225(0,5) = (99,253). \end{aligned}$$

Encrypted version of the message is: ((99, 253), (51, 58)), where $x_1 = 99$, $y_1 = 253$, $x_2 = 51$, and $y_2 = 58$.

Table 2: Knapsack based ECC encryption

Character	Encryption before knapsack applied kG, P'_m+kP_B	Encryption after Knapsack Applied
S=83	(99,253),(51,58)	(18756,82031),(3756,656)
A=65	(99,253),(116,280)	(18756,82031),(656,3751)
V=86	(99,253),(427,287)	(18756,82031),(472006,488126)
E=69	(99,253),(135,341)	(18756,82031),(96876,406901)

Apply knapsack algorithm using a_i vector

$$\begin{aligned} a_i &= 1, 5, 25, 125, 625, 3125, 15625 \text{ (where } n = 5) \\ x_{99} &= 99 \rightarrow 1100011 \text{ [}binary\ value\ of\ 99\text{]}. \end{aligned}$$

Therefore

$$\begin{aligned} S[x_1] &= \sum_{i=1}^{m} a_i x_i \\ [99] &= 1 + 5 + 0 + 0 + 0 + 3125 + 15625 \\ &= 18756. \end{aligned}$$

Similarly

$$\begin{aligned} S[253] &= 82031 \\ S[51] &= 3756 \\ S[58] &= 656. \end{aligned}$$

Hence the transmitted message is $(18756, 82031)$, $(3756, 656)$. The recovery of bit pattern for x_{99} is done as follows:

$$\begin{aligned} 18756 - 15625 &= 3131 \rightarrow 1 \\ 3131 - 3125 &= 6 \rightarrow 1 \\ 6 - 625 &= -ve \rightarrow 0 \\ 6 - 125 &= -ve \rightarrow 0 \\ 6 - 25 &= -ve \rightarrow 0 \\ 6 - 5 &= 1 \rightarrow 1 \\ 1 &= 1 \rightarrow 1. \end{aligned}$$

Therefore, $x_{99} = 1100011$ (read from bottom up). Similarly other coordinates are recovered by applying reverse knapsack algorithm. Thus we are able to recover the encrypted version

$$(99, 253), (51, 58).$$

From this P_m should be retrieved, using B's private key n_B.

$$\begin{aligned} 277(99, 253) &= (212, 151) \\ P'_m &= (51, 58) - (212, 151) = (475, 199). \end{aligned}$$

Now apply discrete logarithm concept to get the ASCII value of "S".

$$S(1, 316) = (475, 199).$$

Therefore, $S = 83$. Thus we retrieve the character "S". Detailed working for the remaining characters is given in the Appendix A.

5 Discussions and Conclusions

A plaintext message 'SAVE' is taken for implementing the algorithm proposed in this paper. Each character in the message is represented by its ASCII value. Each of these ASCII value is transformed into an affine point on the EC, by using a starting point called Pm. This Pm may be selected to be different from the Base point G. Transformation of the plaintext ASCII value by using an affine point is one of the contributions of this work.The purpose of this transformation is two fold. Firstly a single digit ASCII integer of the character is converted into a set of co-ordinates to fit the EC. Secondly the transformation introduces non-linearity in the character thereby completely camouflaging its identity. This transformed character of the message is encrypted by the ECC technique. ECC itself is a very secure algorithm for encryption. However, not satisfied with it, we apply the knapsack algorithm, so that the entire encrypted version turns into an ensemble of confusing integers, thereby discouraging a potential cryptanalyst from attempting a brute force attack. Applying Knapsack algorithm to the ECC encrypted message is another new contribution of this work, which has not been attempted so far.

The table below shows the results for the message "SAVE". Encryption with Knapsack incorporated is shown in Table 2 and Decryption with Knapsack reversed is shown in Table 3

Table 3: Knapsack based ECC decryption

Reversal of knapsack	Decryption	Discrete Logarithm
99,253,51,58	475,199	83
99,253,116,280	298,182	65
99,253,427,287	68,91	86
99,253,135,341	83,111	68

The above table shows encryption of 'SAVE' by using ECC technique. The results before and after applying Knapsack algorithm are shown. Decryption of ECC encrypted message is itself quite a formidable task, unless we have knowledge about the private key n_B, the secret integer k, the affine point P_m. Now by extending the Knapsack algorithm to the ECC encrypted message, we

Table 4: RSA encryption and decryption

M	$C = M^e \bmod n$	$M = C^d \bmod n$
S=83	458	83
A=65	401	65
V=86	260	86
E=69	345	69

Table 5: Execution time

	Execution Time (Both Encryption and Decryption)
With Knapsack	$\sim$ 60.9 ms
With out Knapsack	$\sim$ 55.6 ms

introduce thorough confusion and diffusion. Thereby the message becomes absolutely secure.

The strength of Knapsack algorithm lies in the selection of the a_i vectors. As said earlier, there are infinite ways of selecting these vectors. All the series available in the world such as Taylor's series, Maclaurin's series or fanciful generations like

$$a_i \rightarrow 1, n, (n+1)^1, (n+2)^2, (n+3)^3, \cdots, (n+p)^p,$$

can be used. The limit to this selection is left to the imagination of the researcher. Even assuming that the opponent knows all the relevant parameters such as private key n_B, the secret integer k, and the modular prime value p, he will not be able to figure out the ai vectors. More than that the 'n' used in the ai is another hurdle to be crossed. Thus, this work introduces two new concepts, such as the transformation of the ASCII into an affine point on the EC, and extending the Knapsack algorithm, which makes ECC algorithm one of the most challenging and formidable one, among all the encryption strategies. This is one of the most secure schemes for storing massive personal and sensitive data for archival purposes, such as National registry.

The same message 'SAVE' is also encrypted using the public key cryptography scheme called RSA. The RSA used a public key $e = 31$, private key $d = 159$ and modulus value $n = 667$. It yields the following result as shown in the Tables 4 and 5.

The RSA took 36.26ms to do the encryption and decryption. The same message with Knapsack based ECC approach took 60.9 ms to perform Encryption/Decryption. This is 24.64ms more than time taken for RSA. The storage space required for writing the ECC results on to the CD/DVD is more than twice, as compared to the space required for RSA results. The following table illustrate the execution time taken for with knapsack and with out knapsack.

Then the question arises why use ECC strategy at all! In using the Knapsack based ECC algorithm we need to keep only the a_i vectors secret. The crypt analyst may know all other values. The decryption process is totally infeasible in Knapsack based ECC and is worthwhile for storing/transmitting sensitive data of national significance. The RSA algorithm is equally secure. But many values like modulus operator n, and its prime factors like p,q need to be kept secret. Moreover RSA needs use of numbers of 120 digits for 'n' for better security.

6 Future Scope

What further work can be done as an extension of the present work? There is plenty of scope to toy with the selection of the ai vector. Depending on the amount of memory requirement, one can analyze the use of Knapsack based ECC in small memory devices like smart cards and mobile devices. One can try using random and pseudo random number generators for choosing the secret integer k, and the private key n_B. ECC is a vase field where there is large scope for higher research.

Acknowledgement

The authors are grateful to the management of Thiagarajar College of Engineering Madurai, India, for granting permission to undertake this research work. They express their gratitude to Smart and Secure Project sponsored by the National Technical Research Organization, Government of India, for providing financial support. Our thanks are due to the Head of the Department of Computer Science and Engineering of Thiagarajar College of Engineering for allowing us the use of the laboratories and computing facilities.

References

[1] M. Aydos, T. Yanik, and C. K. Kog, "High-speed implementation of an ECC-based wireless authentication protocol on an ARM microprocessor," *IEE Proceedings of Communication*, vol. 148, no. 5, pp. 273-279, Oct. 2001.

[2] G. Chen, G. Bai, and H. Chen, "A high-performance elliptic curve cryptographic processor for general curves Over GF(p) based on a systolic arithmetic unit," *IEEE Transactions on Circuits System - II: Express Briefs*, vol. 54, no. 5, pp. 412-416, May. 2007.

[3] R. C. C. Cheng, N. J. Baptiste, W. Luk, and P¿ Y. K. Cheung, "Customizable elliptic curve cryptosystems," *IEEE Transactions on VLSI Systems*, vol. 13, no. 9, pp. 1048-1059, Sep. 2005.

[4] A. Cilardo, L. Coppolino, N. Mazzocca, and L. Romano, "Elliptic curve cryptography engineering," *Proceedings of the IEEE*, vol. 94, no. 2, pp. 395-406, Feb. 2006.

[5] W. Diffie, "The first ten years of Public Key cryptography," *Proceddings of IEEE*, vol. 76, no. 5, pp. 560-577, May 1988

[6] K. M. Finnigin, B. E. Mullins, R. A. Raines, and H. B. Potoczny, "Cryptanalysis of an elliptic curve cryptosystem for wireless sensor networks," *International Journal of Security and Networks*, vol. 2, no. 3/4, pp. 260-271, 2006.
[7] M. Hedabou, "A frobenius map approach for an efficient and secure multiplication on Koblitz curves," *International Journal of Network Security*, vol. 3, no. 3, PP. 239-243, Nov. 2006.
[8] K. Lauter, "The advantages of elliptic cryptography for wireless security," *IEEE Wireless Communications*, pp. 62 - 67, Feb. 2006.
[9] J. Lee, H. Kim, Y. Lee, S. M. Hong, and H. Yoon, "Parallelized scalar multiplication on elliptic curves defined over optimal extension field," *International Journal of Network Security*, vol. 4, no. 1, PP. 99-106, Jan. 2007.
[10] C. J. McIvor, M. McLoone, and J. V. McCanny, "Hardware elliptic curve cryptographic processor over GF(p)," *IEEE Transactions on Circuits Syst. I: Reg. Papers*, vol. 53, no. 9, pp. 1946-1957, Sep. 2006.
[11] S. Moon, "A binary redundant scalar point multiplication in secure elliptic curve cryptosystems," *International Journal of Network Security*, vol. 3, no. 2, PP. 132-137, Sep. 2006.
[12] Z. J. Shi, and H. Yan, "Software implementation of elliptic curve cryptography," *International Journal of Network Security*, vol. 7, no. 2, pp. 157-166, Sep. 2008.
[13] *Standard Specifications for Public Key Cryptography*, IEEE Standard p1363, 2000.
[14] W. Stallings, *Cryptography and Network Security*, Prentice Hall, 4th Edition, 2006.
[15] H. Wang, B. Sheng, and Q. li, "Elliptic curve cryptography-based access control in sensor networks," *International Journal of Security and Networks*, vol. 1, no. 3/4, pp. 127-137, 2006.
[16] L. Yongliang, W. Gao, H. Yao, and X. Yu, "Elliptic curve cryptography based wireless authentication protocol," *International Journal of Network Security*, vol. 4, no. 1, PP. 99-106, Jan. 2007.

Appendix A

Plaintext is "A", whose ASCII value is 65. Therefore,

$$\begin{aligned} P_B &= n_B G = 277(0,5) = (260,48) \\ P'_m &= 65(1,316) = (298,182) \\ kP_B &= 225(260,48) = (212,151) \\ P'_m + kP_B &= (298,182) + (212,151) = (116,280) \\ kG &= 225(0,5) = (99,253) \end{aligned}$$

Encrypted version of the message is: $((99,253), (116,280))$, where $x_1 = 99$, $y_1 = 253$, $x_2 = 116$, $y_2 = 280$.

Apply knapsack algorithm using a_i vector

$$\begin{aligned} a_i &= 1,5,25,125,625,3125,15625 \text{ (where } n = 5) \\ x_{280} &= 280 \rightarrow 100011000 \text{ [}binary\ value\ of\ 280\text{]}. \end{aligned}$$

Therefore

$$\begin{aligned} S[x_1] &= \sum_{i=1}^{m} a_i x_i \\ S[280] &= 1+0+0+0+625+3125+0+0+0 \\ &= 3751. \end{aligned}$$

Similarly

$$\begin{aligned} S[253] &= 82031 \\ S[116] &= 656 \\ S[99] &= 18756. \end{aligned}$$

Hence the transmitted message is (18756,82031), ((656,3751). The recovery of bit pattern for x_{280} is done as follows:

$$\begin{aligned} 3751 - 390625 &= -ve \rightarrow 0 \\ 3751 - 78125 &= -ve \rightarrow 0 \\ 3751 - 15625 &= -ve \rightarrow 0 \\ 3751 - 3125 &= 626 \rightarrow 1 \\ 626 - 625 &= 1 \rightarrow 1 \\ 1 - 125 &= -ve \rightarrow 0 \\ 1 - 25 &= -ve \rightarrow 0 \\ 1 - 5 &= -ve \rightarrow 0 \\ 1 &= 1 \rightarrow 1. \end{aligned}$$

Therefore, $x_{280} = 100011000$ (read from bottom up). Similarly other coordinates are recovered by applying reverse knapsack algorithm. Thus we are able to recover the encrypted version: $(99,253), (116,280)$.

From this P_m should be retrieved, using B's private key n_B.

$$\begin{aligned} 277(99,253) &= (212,151) \\ P'_m &= (116,280) - (212,151) = (298,182). \end{aligned}$$

Now apply discrete logarithm concept to get the ASCII value of "A".

$$A(1,316) = (298,182).$$

Therefore, $A = 65$. Thus we retrieve the character "A". Plaintext is "V", whose ASCII value is 86. Therefore,

$$\begin{aligned} P_B &= n_B G = 277(0,5) = (260,48) \\ P'_m &= 86(1,316) = (68,91) \\ kP_B &= 225(260,48) = (212,151) \\ P'_m + kP_B &= (68,91) + (212,151) = (427,287) \\ kG &= 225(0,5) = (99,253). \end{aligned}$$

Encrypted version of the message is: $((99,253), ((427,287))$, where $x_1 = 99$, $y_1 = 253$, $x_2 = 427$, $y_2 = 287$. Apply knapsack algorithm using a_i vector

$$\begin{aligned} a_i &= 1,5,25,125,625,3125,15625,78125 \text{ (where } n = 5) \\ x_{253} &= 253 \rightarrow 11111101 \text{ [}binary\ value\ of\ 253\text{]}. \end{aligned}$$

Therefore,

$$\begin{aligned} S[x_1] &= \sum_{i=1}^{m} a_i x_i \\ S[253] &= 1+5+25+125+625+3125+0+78125 \\ &= 82031. \end{aligned}$$

Similarly,

$$\begin{aligned} S[99] &= 18756 \\ S[427] &= 472006 \\ S[287] &= 488126. \end{aligned}$$

Hence the transmitted message is $(18756, 82031)$, $(472006, 488126)$. The recovery of bit pattern for x_{253} is done as follows:

$$\begin{aligned} 3751-390625 &= -ve \rightarrow 0 \\ 82031-78125 &= 3906 \rightarrow 1 \\ 3906-15625 &= -ve \rightarrow 0 \\ 3906-3125 &= 781 \rightarrow 1 \\ 781-625 &= 156 \rightarrow 1 \\ 156-125 &= 31 \rightarrow 1 \\ 31-25 &= 6 \rightarrow 1 \\ 6-5 &= 1 \rightarrow 1 \\ 1 &= 1 \rightarrow 1. \end{aligned}$$

Therefore, $x_{253} = 11111101$ (read from bottom up). Similarly other coordinates are recovered by applying reverse knapsack algorithm. Thus we are able to recover the encrypted version: $(99, 253)$, $(427, 287)$.

From this P_m should be retrieved, using B's private key n_B:

$$\begin{aligned} 277(99,253) &= (212,151) \\ P'_m &= (427,287)-(212,151) = (68,91). \end{aligned}$$

Now apply discrete logarithm concept to get the ASCII value of "V".

$$V(1,316) = (68,91).$$

Therefore, $A = 86$. Thus we retrieve the character "V". Plaintext is "E", whose ASCII value is 69. Therefore,

$$\begin{aligned} P_B &= n_B G = 277(0,5) = (260,48) \\ P'_m &= 69(1,316) = (83,111) \\ kP_B &= 225(260,48) = (212,151) \\ P'_m + kP_B &= (83,111)+(212,151) = (135,341) \\ kG &= 225(0,5) = (99,253). \end{aligned}$$

Encrypted version of the message is: $((99,253), (135,341))$, where $x_1 = 99$, $y_1 = 253$, $x_2 = 135$, $y_2 = 341$. Apply knapsack algorithm using a_i vector:

$$\begin{aligned} a_i &= 1,5,25,125,625,3125,15625,78125 \text{ (where } n=5) \\ x_{135} &= 135 \rightarrow 10000111 \; [\textit{binary value of } 253]. \end{aligned}$$

Therefore,

$$\begin{aligned} S[x_1] &= \sum_{i=1}^{m} a_i x_i \\ S[135] &= 1+0+0+0+0+3125+15625+78125 \\ &= 96876. \end{aligned}$$

Similarly,

$$\begin{aligned} S[253] &= 82031 \\ S[99] &= 18756 \\ S[341] &= 406901. \end{aligned}$$

Hence the transmitted message is $(18756, 82031)$, $(96876, 406901)$. The recovery of bit pattern for x_{135} is done as follows:

$$\begin{aligned} 3751-390625 &= -ve \rightarrow 0 \\ 96876-78125 &= 18751 \rightarrow 1 \\ 18751-15625 &= 3126 \rightarrow 1 \\ 3126-3125 &= 1 \rightarrow 1 \\ 1-625 &= -ve \rightarrow 0 \\ 1-125 &= -ve \rightarrow 0 \\ 1-25 &= -ve \rightarrow 0 \\ 1-5 &= -ve \rightarrow 0 \\ 1 &= 1 \rightarrow 1. \end{aligned}$$

Therefore, $x_{135} = 10000111$ (read from bottom up). Similarly other coordinates are recovered by applying reverse knapsack algorithm. Thus we are able to recover the encrypted version:

$$(99,253), (135,341).$$

From this P_m should be retrieved, using B's private key n_B.

$$\begin{aligned} 277(99,253) &= (212,151) \\ P'_m &= (135,341)-(212,151) = (83,111). \end{aligned}$$

Now apply discrete logarithm concept to get the ASCII value of "E".

$$E(1,316) = (83,111).$$

Therefore, $E = 69$. Thus we retrieve the character "E".

R. Rajaram Ramasamy Dean of CSE/IT, Thiagarajar College of Engineering, has BE degree in Electrical and Electronics Engineering from Madras University in 1966. He secured the M Tech degree in Electrical Power Systems Engineering in 1971 from IIT Kharagpur, and the Ph.D. degree on Energy Optimization from Madurai Kamaraj University in 1979. He and his research scholars have published/presented more that 45 research papers in Journals and Conferences. Six of his scholar secured the Ph.D. degree in computer science and communications

areas. Two have submitted thesis and awaiting their results. Six are currently pursuing their Ph.D. research in Anna University with his guidance. His current areas of interest are Mobile Agents, Cryptography and Data Mining. He has published more than 13 text books on Computer languages and Basic Communications. He attended the International Seminar on Solar Energy at University of Waterloo, Canada during 1978. He has served the Makerere University at Uganda during 1977-1978 and University of Mosul during 1980-1981. He secured two best technical paper awards from the Institution of Engineers India and one from Indian Society for Technical Education. He has traveled to Malaysia, London, Paris, Belgium New York, Toronto, Nairobi.

M. Amutha Prabakar received the B. E. degree in Computer Science and Engineering, in 2003; the M. E. in Computer Science and Engineering, in 2005. He had worked as a lecturer in the department of Computer Science and Engineering, R. V. S. College of Engineering and Technology, India from 2004-2007. Now he is worked as a Research Associate in Smart and Secure Environment Lab under IIT, Madras. His current research interests include Cryptography and Security.

M. Indra Devi received her the B.E. degree in Computer Science and Engineering from Madurai Kamaraj University in 1990 and M.E. degree in Computer Science and Engineering from Madurai Kamaraj University in 2003. She is currently a senior lecturer at Information Technology Department at Thiagarajar College of Engineering, Madurai. She has published fifteen papers in national and international conferences and two papers in international journals. Her research interests include machine learning applications and web mining. She is a life member of Computer Society of India, Institution of Engineers, India and Indian Society for Technical Education.

M. Suguna is a research assistant in Thiagarajar college of Engineering, Madurai for the Smart and Secure Environment project. Her research includes wireless networks and system security. She has received a degree A.M.I.E in computer science from The Institute of Engineers(India),kolkata.