Implementation of ECC with Hidden Generator Point in Wireless Sensor Networks

Ravi Kishore Kodali
Department of E and C E
National Institute of Technology
Warangal, 506004, India

***Abstract*—With ever growing demand for Wireless Sensor Networks (WSNs) in military and commercial application areas, the urge for secure data exchange over the network is also on the increase. The standard cryptographic algorithms, such as the RSA can not address the security issue due to its computational complexity and the resource constrained nature of the constituent nodes. Another public key cryptographic (PKC) algorithm, Elliptic curve cryptography (ECC) has been emerging as a promising alternative to be used in WSN nodes, as it is capable of providing a similar security level using smaller key length compared to that of the RSA. As WSN nodes are deployed randomly over the field, these nodes are more vulnerable to the man-in-middle (MIM) attack. In traditional ECC algorithm, the *Generator point* is published along with other domain parameters. An intruder, launching MIM attack, could crack the public key, leading to a security breach in the network. This work proposes a technique for ECC with a hidden generator point in order to overcome the MIM attack. Three different algorithms based on distribution of points on the elliptic cure (EC), using a different generator point for each encrypted message and selecting different generator points for each session are discussed. A comparison based on the computational cost and security for three different techniques is also presented.**

Keywords- ECC, MIM attack, generator point.

I. Introduction

Generally, WSN nodes are deployed randomly in remotely located hostile environments. These nodes and the information exchanged among them are easily accessible to an intruder, who can take advantage of the resource constrained nature of the nodes to break the public key, thereby breaching the security of the network. This type of attack is called man-in-middle (MIM) attack [6] and WSN nodes are vulnerable to such attacks. The WSN nodes using traditional ECC as their cryptographic algorithm, publish their *Generator point*, (G), which helps an intruder to launch a MIM attack easily. This work discusses techniques, which make use of the ECC, with a hidden generator point to overcome the MIM attack.

II. ECC with Hidden Generator Point

Initially, both Alice and Bob agree on a particular elliptic curve (EC) in the prime field F_P, with a specific base point termed as the *generator point*, G. The G, is one of the valid points on the EC, which has the highest order [10]. Both Alice and Bob choose their respective private keys independently, by selecting randomly any scalar integer in the prime field, F_P. The corresponding public keys, Q_A and Q_B, are computed by multiplying the *generator point*, G, with the corresponding private keys: K_A and K_B. These public keys are then shared over the network between them, who again multiply these with the corresponding private key and generate a shared secret key, given as $T = K_A * Q_B = K_B * Q_A$. In traditional ECC, the *Generator point*, G, is made known to everyone including an intruder. This makes the network more prone to the MIM attack. To overcome this problem and to make the network secure against such vulnerabilities, ECC is implemented with a *Hidden generator point*. Here, three different protocols are presented to perform the same.

A. First Protocol [13]

This protocol depends on the distribution of points on the EC. A common algorithm for *Generator point*, G selection is designed and spread over the network with the help of *Certificate Authority* (CA), to different nodes for selection [13]. Consider a prime field, $P = 31$, with the domain parameters, $a = -3, b = 1$, the points on the EC, $E_P(a, b)$, are given as:

$(0,1)$ $(0,30)$ $(3,9)$ $(3,22)$ $(5,7)$ $(5,24)$
$(10,14)$ $(10,17)$ $(11,11)$ $(11,20)$ $(12,9)$ $(12,22)$
$(13,12)$ $(13,19)$ $(15,13)$ $(15,18)$ $(16,9)$ $(16,22)$
$(19,13)$ $(19,18)$ $(20,6)$ $(20,25)$ $(23,3)$ $(23,28)$
$(24,12)$ $(24,19)$ $(25,12)$ $(25,19)$ $(28,13)$ $(28,18)$

An EC point, C, such that $C \in E_P(a, b)$, can be chosen as the *Generator point*, G. This assumption is made on the basis that each curve point in the given example has the same order. The points on the EC are distributed into different groups to assist in the selection of the *Generator point*.

During the selection of *Generator point*, G, a criterion like max(x), then max(y) is used. A point, $(28, 18)$ is selected as the G, for the EC under consideration. After G is selected, the communication among the nodes can be performed in same way as in traditional method to generate the shared key. This protocol is easy to implement and requires less computation. However, the requirement of a *Certificate Authority* (CA) in the WSN, makes it difficult to implement. In case, the CA signed digital envelope is captured, the selected protocol stands exposed leading to security breach in the network.

B. Second Protocol [13]

This second protocol addresses the issue [13], where Bob and Alice want to exchange information but have no prior

978-1-4799-3635-9/14/$31.00 ©2014 IEEE

knowledge regarding the public keys of each other and there is no CA for Generator point selection. Figure 1 shows the message exchange using this second protocol.

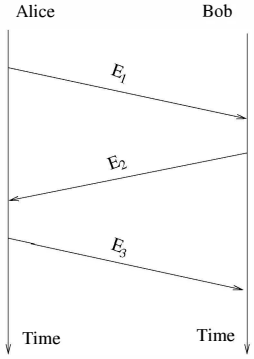

Fig. 1. Message exchange using Second Protocol [13]

Few message points are exchanged between Alice and Bob, before receiving the actual message. It is assumed that Alice wants to forward a message, M to Bob. First, Alice selects the *Generator point*, G and her private key, K_A. Then, Alice computes the private key inverse, K_A^{-1}, such that, $K_A * K_A^{-1} = 1$. Bob also selects his private key, K_B and Bob computes the inverse of the private key, K_B^{-1}. In the beginning, Alice transmits a message point, E_1, such that:

$$E_1 = (K_A^{-1}G, K_A^{-1}M + K_A^{-1}G)$$

Upon receiving E_1, Bob manipulates it, multiplies it with his inverse private key and transmits it back to Alice.

$$K_A^{-1}M = K_A^{-1}G + K_A^{-1}G - K_A^{-1}G$$

$$E_2 = K_A^{-1}K_B^{-1}M$$

Upon receiving, Alice multiplies E_2 with her private key and transmits the same to Bob, where he again multiplies it with his private key to extract the message.

$$E_3 = K_A^{-1}K_B^{-1}M * K_A = K_B^{-1}M$$

At Bob

$$Message = K_B^{-1}M * K_B = M$$

Finally, Bob is able to decrypt the message by multiplying E_3 by his private key. These steps need to be carried out every time a new message needs to be exchanged, demanding large computational overhead.

C. Third Protocol

As observed in the second protocol, for a single message exchange in ECC with hidden generator point, many computations are required. Amongst them, scalar multiplication is used many times, which is the most time consuming operation in ECC. This kind of implementation in WSN is compute-expensive and drains out the battery energy quickly.

In this protocol, a new mechanism is proposed. It focusses on generating a unique shared key for each session of communication between the two nodes. This attempts to circumvent the large computational overhead of the second protocol and yet maintain the same security level. Fig 2 shows generation of the unique shared key between two nodes without having a unique generator point.

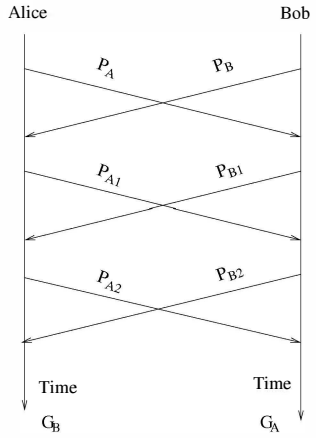

Fig. 2. Key exchange mechanisms in Protocol Third

In this protocol, ECC is implemented over the hidden generator point, generating a shared key between the nodes require many message exchanges during the initial stage of a session. Let us consider that Alice and Bob are the two nodes in the WSN. It is assumed that both Alice and Bob have selected their unique Generator points, G_A and G_B and also their unique private keys, K_A and K_A. The inverse of the private keys, K_A^{-1} and K_B^{-1}, are also computed to satisfy the unity property. After initializing the keys, both Alice and Bob generate their public keys, P_A and P_B and transmit the same to each other.

$$P_A = G_A * K_A^{-1}, \qquad P_B = G_B * K_B^{-1}$$

After receiving the public key, they multiply it with their respective inverse private key and then again transmit the modified key.

$$P_{A1} = G_B * K_B^{-1} * K_A^{-1} \qquad At\ Alice$$

$$P_{B1} = G_A * K_A^{-1} * K_B^{-1} \qquad At\ Bob$$

Upon receiving P_{A1} and P_{B1}, Alice and Bob multiply it with their respective private keys and transmit again.

$$P_{A2} = G_A * K_A^{-1} * K_B^{-1} * K_A = G_A * K_B^{-1}$$

$$P_{B2} = G_B * K_B^{-1} * K_A^{-1} * K_B = G_B * K_A^{-1}$$

When P_{A2} reaches Bob, he multiplies it with his private key to obtain the Generator point of Alice. Similar action is performed by Alice after receiving P_{B2}.
At Alice,

$$G_B * K_A^{-1} * K_A = G_B$$

At Bob,

$$G_A * K_B^{-1} * K_B = G_A$$

As of now, both Alice and Bob have the Generator points, G_A and G_B, they compute their shared key by adding both the Generator points.

$$Sharedkey(G_C) = G_A + G_B$$

After calculating the shared key, the messages between Alice and Bob can be securely exchanged in a conventional manner. This shared key is used for only one communication session between the nodes.

III. Comparison

In this paper, the three different protocols for ECC with hidden generator point are discussed and these are implemented using the IRIS MEMSIC WSN nodes over the 192- NIST prime field. The prime field and the domain parameters for 192-NIST field is given in Table I.

TABLE I
DOMAIN PARAMETERS FOR 192-BIT NIST FIELD

Parameter	192-bit value(hex)
P_{192}	fffffffffffffffffffffffffffffffeffffffffffffff
a	fffffffffffffffffffffffffffffffefffffffffffffc
b	64210519e59c80e70fa7e9ab72243049feb8deecc146b9b1

As the IRIS WSN nodes have Atmel's ATmega 128L micro-controller with an 8- bit CPU, all of the prime field arithmetic operations like addition, subtraction, multiplication and division are performed over 8-bit. The 192-bit values are stored in an 8-bit array of size 24. The inversion operation is performed using extended euclidean algorithm. The three basic ECC operations, point addition, point doubling and scalar multiplication are also implemented using the IRIS nodes. The point addition and point doubling are implemented over Affine as well Projective coordinate systems and the scalar multiplication operation is implemented using Binary method and Sliding window method. The scalar multiplication operations used in these protocols are performed over Projective coordinate system. The timing results for different scalar multiplication algorithms are shown in Table II.

TABLE II
TIMING RESULTS FOR SCALAR MULTIPLICATION

Multiplication method	Time (seconds)
Binary method over affine coordinate	76
Sliding window method over projective coordinate	17

TABLE III
ELLIPTIC CURVE POINTS

Points	Coordinate(x,y)
G_1	188da80e b03090f6 7cbf20eb 43a18800 f4ff0afd 82ff1012
	07192b95 ffc8da78 631011ed 6b24cdd5 73f977a1 1e794811
G_2	d458e7d1 27ae671b 0c330266 d2467693 53a01207 3e97acf8
	32593050 0d851f33 6bddc050 cf7fb11b 5673a164 5086df3b
G_3	f22c4395 213e9ebe 67ddecdd 87fdbd01 be16fb05 9b9753a4
	26442409 6af2b359 7796db48 f8dfb41f a9cecc97 691a9c79

In the implementation of the first protocol, the generator point selection process is: the first point from the node's memory is selected as a generator point, G_1. N_A and N_B, are the private keys selected by the nodes, which are given in Table IV. The shared key, $K(x_k, y_k)$, generated using the first protocol is given as:

TABLE IV
PRIVATE KEYS FOR THE NODES

Private Key	192-bit value (hex)
N_A	a78a236d60baec0c5dd41b33a542463a8255391af64c74ee
N_B	a542463a8255391af64c74eea78a236d60baec0c5dd41b33

x_k=45ddFF0d61fa5930a5ab98a1a16603fc49aa72e6f37055ab
y_k=733bd0d661dfae0d2e1090f4db65a92aa98c79cfc8113f0f

The total time taken to generate the shared key by the first protocol is 39 seconds.

In the implementation of the second protocol, it is considered that the message, M, is exchanged between the nodes and it is also an EC point and G_1 is selected as the generator point by the node, which has to transmit a message. The intermediate values in the implementation of the second protocol are shown in Table V. In this table, only the values of Y coordinate are shown.

TABLE V
INTERMEDIATE VALUES IN THE SECOND PROTOCOL

	Y coordinate (hex)
E_1	25cddef317e266c1f6296d55f962219094f2d86241914e7f
	b1b0ec87b71a1b5768ab1a96fdd42508cacb14226027a53a
E_2	65442ff6863b43586ac73b0f54d837fda1780c2cddf2ddc3
E_3	eaa3c3dbdb370b9e950521c4a19756fbd968712145d68f4a

The total time taken by the second protocol for one message exchange is 79 seconds.

During the implementation of the third protocol, G_2 and G_3 are selected as the generator points by two nodes and N_A and N_B are their respective private keys. The intermediate values in the third protocol are given in Table VI. In this table, only the values of Y-coordinate are shown. The shared key, $K(x_k, y_k)$ generated using the third protocol is given as:

TABLE VI
INTERMEDIATE VALUES IN PROTOCOL THIRD

	Y coordinate (hex)
P_A	25cddef317e266c1f6296d55f962219094f2d86241914e7f
P_{A1}	8b56d2e3022cf16975fe8c202f114ed7b8ce7deea525e2
P_{A2}	b35e1a5a31a42b85f4cb76342e4971658b42aee1cdeb0a75
P_B	133198b6223c6ce00d4967b2b22a103fd0b10761b43b8b449d
P_{B1}	de0d89cf384662151df54bfc146dbfa0af0ef8c265e12aaf
P_{B2}	e6ff8970f0fc4fdebc53bb45a803405c50dffbd3f761d177

x_k=2ca3ae14478c6bc1a4707a48e165c5a042bcad3d2efbd8ab
y_k=e811f047757ebeee379ef07842795ded1b812bef0a9c2518

The total time taken by the third protocol for shared key generation is 123 seconds.

Table VII provides a comparison on the basis of computational cost and security level among these three different protocols used for ECC implementation over the hidden generator point.

TABLE VII
COMPARISON OF PROTOCOLS

Protocols	Computational Cost	Security Level
First	Low	Low
Second	Very High	Very High
Third	High	High

The first protocol is easy to implement and requires the same number of computations as in traditional ECC. However, it uses a CA for spreading the common parameter for generator point selection making it vulnerable, if CA is compromised. Thus the amount of security level provided by this protocol is low.

The second protocol selects a different generator point, whenever a new message is to be exchanged, this leads to a very high level of security. But this security comes at the cost of large number of computations required by this method, which overshadows the security leverage provided by it.

The proposed third protocol uses a unique scheme to find the shared key between the two nodes without having a common generator point. This uses the key exchange mechanism during the initial stage of the session for generating the shared key, henceforth encryption and decryption is done in conventional manner. Hence, the number of computations required in this method is little more than the first protocol and the security level is little less than that of the second protocol.

When two nodes decide to communicate with each other, key exchange mechanism, encryption and decryption are the basic processes, which are performed during each communication session. If only one session of communication is considered between the two nodes and in that session if we consider that the message is exchanged 10 times. The number of basic ECC operations performed by one node during this session is given in Table VIII.

TABLE VIII
COMPARISON OF PROTOCOLS FOR MIM ATTACK

Protocol Name	No. of Scalar Multiplications	No. of Point Additions	Time (seconds)
First Protocol	2	10	89
Second Protocol	30	10	790
Third Protocol	4	11	123

As can be observed from Table VIII, the first protocol requires least number of computations and the second protocol has maximum number of computations. The number of computations required by the third protocol is very close to the first protocol, which also provides good amount of security compared to the second protocol.

IV. CONCLUSION

When ECC is implemented on the resource constrained devices like WSN nodes, the network becomes vulnerable to the man-in the-middle (MIM) attack. To circumvent the MIM attack in a WSN, ECC is implemented in the nodes with hidden generator point. Among the three protocols discussed for ECC with hidden generator point, the third protocol is the most optimum. It provides a balance between computational cost and security level compared to the first protocol, which has very low security level and compared to the second protocol, which has very high computational cost. All these protocols and scalar multiplication techniques are implemented using IRIS WSN nodes. It has been observed that the sliding window scalar multiplication method is much more efficient than the scalar multiplication over affine coordinate system. The second Protocol implementation requires maximum time and the first protocol by taking less time. The implementation of the third protocol has taken slightly longer time than the first protocol, however providing much higher security.

REFERENCES

[1] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: a survey," *Computer networks*, vol. 38, no. 4, pp. 393–422, 2002.

[2] C. T. Inc., "Micaz wireless measurement system," June, 2004.

[3] G. Werner-Allen, K. Lorincz, M. Ruiz, O. Marcillo, J. Johnson, J. Lees, and M. Welsh, "Deploying a wireless sensor network on an active volcano," *Internet Computing, IEEE*, vol. 10, no. 2, pp. 18–25, 2006.

[4] M. Shand and J. Vuillemin, "Fast implementations of rsa cryptography," in *Computer Arithmetic, 1993. Proceedings., 11th Symposium on*. IEEE, 1993, pp. 252–259.

[5] N. Gura, A. Patel, A. Wander, H. Eberle, and S. Shantz, "Comparing elliptic curve cryptography and rsa on 8-bit cpus," *Cryptographic Hardware and Embedded Systems-CHES 2004*, pp. 925–943, 2004.

[6] X. Huang, P. G. Shah, and D. Sharma, "Protecting from attacking the man-in-middle in wireless sensor networks with elliptic curve cryptography key exchange," in *Network and System Security (NSS), 2010 4th International Conference on*. IEEE, 2010, pp. 588–593.

[7] D. Hankerson, A. Menezes, and S. Vanstone, *Guide to elliptic curve cryptography*. Springer, 2004.

[8] M. Brown, D. Hankerson, J. López, and A. Menezes, *Software implementation of the NIST elliptic curves over prime fields*. Springer, 2001.

[9] R. K. Kodali and H. S. Budwal, "High performance scalar multiplication for ecc," in *Computer Communication and Informatics (ICCCI), 2013 International Conference on*. IEEE, 2013, pp. 1–4.

[10] O. YAYLA, "Scalar multiplication on elliptic curves," Ph.D. dissertation, Master?s Thesis, Department of Cryptography, Middle East Technical University, August 2006. Available at http://www3. iam. metu. edu. tr/iam/images/3/3e/O% C4% 9Fuzyaylathesis. pdf(link tested 01-Dec-2011), 2006.

[11] X. Huang, D. Sharma, and H. Cui, "Fuzzy controlling window for elliptic curve cryptography in wireless sensor networks," in *Information Networking (ICOIN), 2012 International Conference on*. IEEE, 2012, pp. 312–317.

[12] X. Huang, D. Sharma, and P. Shah, "Efficiently fuzzy controlling with dynamic window in elliptic curve cryptography sensor networks," in *Proceeding of the International MultiConference of Engineers and Computer Scientists*, vol. 1, 2011.

[13] X. Huang, P. G. Shah, and D. Sharma, "Fast scalar multiplication for elliptic curve cryptography in sensor networks with hidden generator point," in *Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC), 2010 International Conference on*. IEEE, 2010, pp. 243–249.