A New Key Management Scheme for Wireless Sensor Networks using an Elliptic Curve

Soram Ranbir Singh¹, Ajoy Kumar Khan² and Takhellambam Sonamani Singh¹

¹Department of Computer Science and Engineering, Manipur Institute of Technology, Imphal – 795001, Manipur, India; ranbirsoram@gmail.com, sona.cse2013@gmail.com
²Department of Computer Science and Engineering, Assam University, Silchar – 788011, Assam, India; ajoyiitg@gmail.com

Abstract

Objectives: Our objective is to design a new key management scheme for wireless sensor networks.

Methods/Analysis: The paper describes an effective and novel key management scheme for wireless sensor networks using elliptic curve public key cryptosystem. The proposed key management scheme uses multiple private keys and a pair of public keys.

Findings: The proposed key management scheme was tested for performance evaluation using various elliptic curves of different parameters. The experimental results showed that our scheme performed well in spite of using multiple private keys and a pair of public keys.

Applications/Improvements: The proposed key management scheme can be used in wireless sensor networks as elliptic curve cryptosystem uses small key length or size in contrast to other public key cryptosystems.

Keywords: Decryption, Elliptic Curve, Encryption, Group Key Management, Wireless Sensor Network

1. Introduction

A WSN generally comprises many low energy, low memory and low computational autonomous sensors, each of which has the capabilities of sensing, detecting and computing a task. It has gained much popularity and attention in the last few decades. They are used in military, disaster management, health care sectors, habitat monitoring in forests, and weather sensing. A base station in a WSN has relatively more energy and computing resources. The sensors in the WSN are used to gather some data and send the data to the base station. The base station then executes some computations on the data to obtain some information. A reliable and secure data link is required between the data source and the base station for the efficient data transfer. The security of WSN is provided by key management schemes. However, it is tricky to decide the proper KMS owing to the low computational and communication resources of WSN. Symmetric key cryptography is very efficient and more suitable for WSN. Symmetric key cryptography has a difficulty in the exchange of secret keys among the nodes.

One of the alternatives to the exchange of secret keys can be pre-deployment of the keys in the nodes. However, the unfriendly environment of WSN makes it susceptible to different attacks.

Public key cryptography (PKC) uses a public key and a private key. The public key is freely announced to the people without risking and compromising the private key, which is to be clandestinely kept by its owner. The public key is used to encrypt plaintext messages and to verify signatures; the private key is used sign signatures and to decrypt the ciphertext to obtain plaintext messages. In the literatures, we can find only a few key KMSs for WSN based on public key cryptography. PKC has not been initially considered in WSNs to secure data communication due to their large key sizes and high computation capability requirement. It has been considered too expensive and very much resource demanding. But, it is investigated in some research works and argued that public key cryptography can still be used in small devices like a sensor node with low computation and memory. There exists very little work towards using the combination of both public key cryptography and symmetric key cryptography.
Elliptic curve cryptographies have, in recent decade, come into consideration, particularly by IEEE, as alternatives to well established cryptosystems such as the RSA and the Digital Signature Standard. Elliptic curve cryptographies have a number of interesting properties, which may make them appropriate tools for meeting security requirements in WSNs. Recent development on ECC provides us new techniques to utilize PKC in wireless sensor networks. The realization of 160-bit key length of ECC on an 8-bit processor of 8Hz clearly shows that an elliptic curve point multiplication takes less than one second, which demonstrates that the ECC is practical in sensor networks. Compared with symmetric key cryptography, PKC provides a more lithe and simple edge, requiring no prior key distribution, and no pairwise key exchange and sharing. The Elliptic Curve Digital Signature Algorithm uses ECC to generate signature for authentication and other security related purposes.

2. Elliptic Curve

An elliptic curve is the set of all the coordinates satisfying the polynomial equations of the form given in equation (1).

\[ y^2 + xy = x^3 + Ax + B \]

or

\[ y^2 = x^3 + Ax + B \]

or

\[ y^2 = x^3 + Ax + B \]

The constant \( A, B \) and variables \( x, y \) can be real, complex, polynomial, integer, and any other field elements.

![Figure 1. Arbitrary Points A and -A.](image)

One of the elliptic curve properties that makes it usable in cryptography is that if we take any two points on the curve, then we can sketch a straight line passing through the two points and intersecting the elliptic curve at a third point. The negative of this third point is defined as the sum of the two points.

Let's take a point \( A = (x_1, y_1) \) on any curve. The formula for finding \( -A = (x_1, -y_1) \) as shown in Figure 1.

![Figure 2. Adding two Points.](image)

Let us take any two points, \( A = (x_1, y_1) \) and \( B = (x_2, y_2) \), on the curve. Then sum, of \( A \) and \( B \), \( A + B = (x_3, y_3) \), is calculated as in Figure 2.

![Figure 3. Doubling a Point.](image)

To define the double of \( A \), we draw the tangent line to the curve at \( A \). This line will surely intersect the curve at a point as given in Figure 3. Then, \( C = 2A = (x_3, y_3) \) is the reflection of this point about the x-axis.

2.1 Group Law of Elliptic Curves

Let \( E \) be a given elliptic curve defined under field \( F \). The following formulae allow us to form algebra on the curve.

2.1.1 Group law for \( y^2 = x^3 + ax + b \)

- Identity element: \( A(x_1, y_1) + 0 = 0 + A(x_1, y_1) = A(x_1, y_1) \) for all \( A(x_1, y_1) \in E(F) \).
• Negative of a point: Let \( A=(x, y) \in \mathbb{E}(F) \), then 
\[ (x, y) + (x, -y) = 0. \] The point \( (x, -y) \) is denoted by \(-A\) and is the negative of \( A \). We must note that \(-A\) is also a point in curve. Also, as in normal algebra, \(-0 = 0\).

• Addition of Points: Let \( A=(x, y) \in \mathbb{E}(F) \) and \( B=(x, y) \in \mathbb{E}(F) \) where \( A \neq \pm B \). Then, the addition of two points is given by \( C=(x, y) = A + B \), where 
\[ x_3 = \frac{y_1 - y_2}{x_1 - x_2}, \quad y_3 = \frac{(x_1 - x_3)y_1}{x_1 - x_2}. \]

• Doubling of a Point: Let \( A=(x, y) \in \mathbb{E}(F) \), where \( A \neq \pm A \). Then \( \frac{1}{2} A \) can be evaluated as \( \mu \) where 
\[ x_3 = \frac{y_1 - y_2}{x_1 - x_2}, \quad y_3 = \frac{(x_1 - x_3)y_1}{x_1 - x_2}. \]

3. Elliptic Curve Discrete Logarithm Problem

Let \( \mathbb{E} \), be an elliptic curve, \( A \) be any point on the curve, and \( \mu \) be an integer. The point \( B=\mu A \) can be evaluated by repeated additions of point, \( A \). However, it is very tough to determine the value of \( \mu \) knowing the two points: \( \mu A \) and \( A \). The Elliptic Curve Logarithm Problem (ECDLP)\(^{10,11}\) says that “Given a point \( A \) and the point \( B=\mu A \) on the curve \( \mathbb{E} \), find the value of the integer \( \mu \)”.

4. Proposed Key Management Scheme

We use an elliptic curve in this KMS. The curve parameters \( D=(p,A,B,G) \), are given in Table 1. The curve parameters are public.

| A,B,p | A,B,p in \( y^2 = x^3 + Ax + B \) mod p | G(x,y) | base point of the curve |
|-------|---------------------------------|-------|---------------------|

4.1 Key Generation

Step 1: Alice and Bob choose a common curve \( y^2 \) mod \( p \) = \( x^3 + ax + b \) mod \( p \), with the generator point as \( G(x, y) \).

Step 2: Alice chooses \( N \) number of integers \( d_i^A, d_i^B, \ldots, d_i^N \in [1, n-1] \) and \( N \) points \( p_i^B(x, y) = \alpha_i^B G(x, y) \), \( 1 \leq i \leq N, 1 \leq \alpha_i^B < n \). He uses the integers as his private keys. He then calculates two points \( P_i^AA(x, y) = \sum_{i=1}^{N} d_i^A p_i^B(x, y) \), \( P_i^BB(x, y) = P_i^AA(x, y) + \sum_{i=1}^{N} d_i^B G(x, y) \) on the curve as his public keys.

Step 3: Bob also chooses \( N \) number of integers \( d_i^A, d_i^B, \ldots, d_i^N \in [1, n-1] \) and \( N \) points \( p_i^B(x, y) = \alpha_i^B G(x, y) \), \( 1 \leq i \leq N, 1 \leq \alpha_i^B < n \). He uses the integers as his private keys. He then calculates two points \( P_i^AA(x, y) = \sum_{i=1}^{N} d_i^A p_i^B(x, y) \), \( P_i^BB(x, y) = P_i^AA(x, y) + \sum_{i=1}^{N} d_i^B G(x, y) \) on the curve as his public keys.

4.2 Encryption

Supposing, Alice is dispatching the message \( m=(x, y) \) to Bob. His steps are given below.

Step 1: Alice chooses a random integer \( k \).

Step 2: Using the group law, he computes a pair of points, \( C=kG(x, y) \) and \( C'=P + kP(x, y) - P(x, y) \).

Step 3: Alice sends the two pair of points, \( C, C' \) as ciphertext to Bob.

4.3 Decryption

Step 1: Bob obtains the ciphertext, \( C, C' \) from Alice.

Step 2: Using the group law, he calculates the point, \( m P = C - C' d \).

The proposed KMS can also be used in group key management. Each entity can be assigned an integer as a private key. They collectively form a pair of elliptic curve points as public keys. There are \( N \) such different entities i.e. Bobs. Alice uses the public keys to encrypt the message sent to them. Each of the \( N \) Bobs will collectively and sequentially decrypt the ciphertext to obtain the plaintext sent by Alice. The \( N^{th} \) Bob who finally decrypt the ciphertext will ultimately obtain the plaintext. If an \( M^{th} \) Bob, \( 1 \leq M < N \), refuse to participate in the decryption process then the \( N^{th} \) Bob will not be able to obtain the plaintext.

4.4 Security Analysis

The security of the scheme can be analyzed from two angles. In the first angle, Eve could try to recover the private keys used by Bob(s) with one or more key(s) compromised. In the second angle, Eve could try to recover the plaintext from the ciphertext with...
one or more key(s) compromised. For simplicity, we assume here that one of the keys i.e., the Nth key is compromised. But in real life applications, many keys could be compromised, and still the algorithms will work provided one key is not compromised. We assume here that Eve attacks the ciphertext and then he keys in \(d_i^B, d_{i+1}^B, ..., d_N^B\). Then, he will get the following equation (2).

\[
P_m = C_2 - C_1 \left( \sum_{i=1}^{N} d_i^B + d_N^B \right)
\]

(2)

\[
P_m = P_m + k \sum_{i=1}^{N} p_i^B \text{ where } x_g = 446467696323197, y_g = 5229680988957888047450137799097.
\]

Finally, Eve gets the equation, \(P_m = P_m \mu G(x_g, y_g)\). It is simplified further to get \(P_m^m = \mu G(x_g, y_g)\). Eve cannot find the scalar \(\mu\) due to ECDLP.

### 4.5 Example

Both Alice and Bob agree upon an elliptic curve of the form \(y^2 = x^3 + ax + b \mod p\), where \(a = 3210947681291476023\) and \(b = 430782315140218274262276694\). The base point or generator is \(G(x_g, y_g)\) where \(x_g = 446467696323197, y_g = 5229680988957888047450437799097\).

Bob chooses 4 integers \(d_1^B = 121214, d_2^B = 121010, d_3^B = 1313135435, d_4^B = 141414344\) and 4 points,

\[
p_1^B(x_1, y_1) = 12222G(x_g, y_g) \text{ where } x_1 = 53320990236729793185976130720963, y_1 = 1153799294876769338359516702125.
\]

\[
p_2^B(x_2, y_2) = 13333G(x_g, y_g) \text{ where } x_2 = 42485049154930610035907955876250, y_2 = 46108246434404595031738464362660.
\]

\[
p_3^B(x_3, y_3) = 14444G(x_g, y_g) \text{ where } x_3 = 2513062767126785018616711478504, y_3 = 595197587375611988330612937872.
\]

\[
p_4^B(x_4, y_4) = 15555G(x_g, y_g) \text{ where } x_4 = 3541782168323731139058297576205, y_4 = 38797700135783525689646774935276.
\]

He uses the 4 integers as his private keys. He then calculates the following two points as his public keys.

\[
p_1^B(x_{i1}, y_{i1}) = p_1^B(x_{i1}, y_{i1}) + \sum_{i=1}^{d_i^B} p_i^B(x_{i1}, y_{i1}) \text{ where } x_{i1} = 22593231796335863229141595264432, y_{i1} = 41990988719475256858082264534265.
\]

Assume, Alice is sending the message “Networking” to Bob. He first encodes it to an integer value which is given as “370216099114244838616679”. Next, he transforms it to an x-coordinate of a point on the curve. The encoded point is \(P_m(x_m, y_m)\) where \(x_m = 1110648297342734515850037, y_m = 283443658374212801015584425352921\). Next, Alice picks a random integer \(k = 977\) and calculates the ciphertext points \(P_c\) using Bob’s public keys:

\[
C_m = <C_1, C_2> \text{ where } C_1 = KG(x_g, y_g) \text{ and } C_2 = P_m + k P_1^B \text{ (x,y) - P}(x,y).\]

The ciphertext computed by Alice is the two pair of points, \(C_m = <C_1, C_2>\) where \(C_1 = (x_1, y_1), C_2 = (x_2, y_2)\) and \(x_1 = 480134833113573141469601406468462, y_1 = 430384888180568399815117063777231\) \(x_2 = 214015252560351147106227332026381, y_2 = 321034293459414448490029187565879\).

Bob can recover the plaintext from the ciphertext using \(p_m(x_m, y_m) = C_1 - C_2 \sum_{i=1}^{d_i^B} d_i^B\) where \(x_m = 11106482973427345158500370, y_m = 283443658374212801015584425352921\). The x-coordinate point is transformed back to the integer value “370216099114244838616679”, which is mapped to the text message, “Networking”.

In ECC, the message block is encoded as \(x_1\), the x-coordinate of a point, \(P_m = (x_1, y_1)\), on the curve. Our message \(m\) can be represented by \(x = mk + i\) where \(0 \leq i \leq k\) and \(k\) is an integer whose value is normally less than 10. We calculate \(x_i\) and \(y_i\) of the equation \(y_i^2 = f(x_i) = x_i^3 + Ax_i + B\). We take \(P_m = (x_i, y_i)\) such that \(y_i^2 = f(x_i)\). If \(f(x_i)\) is not a squared, then we increment \(i\) by 1 and continue again until we find an \(x_i\) such that \(y_i^2 = f(x_i) = x_i^3 + Ax_i + B\) and \(0 \leq i \leq k\). To recover \(m\) from \((x_i, y_i)\), compute the value of \(x_i/k\).

### 5. Results and Discussions

This section presents the computational results of the proposed approach. We take three different curves of the form \(y^2 = x^3 + Ax + B \mod p\). The curve parameters are listed in Table 2.

In ECCs, a private key is a positive integer, which is at most 256-bit length (as of now), chosen at random. In uncompressed form, a public key is a point consisting
of two 256-bit integers. The position of the public key coordinate on the elliptic curve is purely determined by the private key, while it’s unfeasible to obtain the private key from the public key coordinate. The y-coordinate can be often omitted as it can be obtained from the curve formula and the x-coordinate. The compressed form saves memory by forgoing the y-coordinate value. The uncompressed form takes 520 bits: the constant 04 prefix, the 256-bit x-coordinate, and the 256-bit y-coordinate.

We can easily convert a public key in uncompressed form to a compressed form; we just leave out the y-coordinate and alter the value of the prefix according to its value. The first byte becomes 02 for even values of y-coordinate and 03 for odd values. The approach presented in this work is coded in Java on an Intel PC of 2.13GHz and 2GB of memory running windows XP. The generators are reported in compressed form.

Typical elliptic curve public cryptosystem uses only one secret integer as private key and one point on the curve as public key\textsuperscript{10,11}. On the other hand, our scheme can use many integers as private keys and a pair points on the curve as public keys. The additional private and public keys may have a burden on the computational cost.

In Figure 4, we report the public key generation time taken in nanosecond(ns). Each result is the average of six test runs. The key generation time taken increases as the number of keys increases.

![Key generation time taken in nanosecond.](image1)

![Encryption time taken to encrypt “Networking”.](image2)

### Table 2. Curve parameters used for performance evaluation

| Curves  | Parameters                                                                 |
|---------|-----------------------------------------------------------------------------|
| ECC-97  | A=2203996953648318990036551108                                              |
|         | B=84893153137031737415457908143                                              |
|         | P=113466572691320833494871051829                                              |
|         | G(x,y)=0200DF7E84C42FEF50C5316C508A                                             |
| ECC-109 | A=321094768129147601892514872825668                                          |
|         | B=430782315140218274262276694323197                                       |
|         | P=56453825208444155552470169072735257                                      |
|         | G(x,y)=030233857E4E88B5F0055126E7DB7C                                       |
| ECC-131 | A=13992675737635781587905235971153316710                                   |
|         | B=1009296542191532464076260367525816293976                                  |
|         | P=15500317978334347859248576414813139942411                                  |
|         | G(x,y)=0203A6F04FC62E2DA1ED0BF62C3FFB568                                    |
| ECC-163 | A=6128664532462971392107386013310947315623793878703                         |
|         | B=299369465704850817450327154462342108212188049461                         |
|         | P=744157000815125078325059510076520610606604922267                       |
|         | G(x,y)=0201DC1E9A482085B3DFA722EB7A541D5050ED31DCA                           |

![Figure 4](image1)

![Figure 5](image2)
Table 3. Performances of our Scheme

| Parameter | Time Taken | Ciphertext Size in byte |
|-----------|------------|-------------------------|
| Key Length in bit | Key Generation(ns) | Encryption(ns) | Decryption(ns) |
| No. of Private keys | Networking |  |
| 10 | ECC-97 | 1 | 7029520 | 1783860 | 2312532 | 116 |
| | | 2 | 5395909 | 1503426 | 4793512 | 116 |
| | | 4 | 8323437 | 1089541 | 1147867 | 117 |
| | | 8 | 9980379 | 987352 | 1480095 | 116 |
| | ECC-109 | 1 | 6129891 | 1285518 | 1244456 | 132 |
| | | 2 | 6782216 | 1855252 | 4892434 | 131 |
| | | 4 | 8901570 | 1151134 | 1854319 | 132 |
| | | 8 | 10131095 | 1144601 | 3641445 | 131 |
| | ECC-131 | 1 | 8181587 | 2481447 | 5918048 | 158 |
| | | 2 | 8703726 | 1858985 | 1362510 | 159 |
| | | 4 | 9640684 | 2462782 | 1378840 | 159 |
| | | 8 | 11127313 | 2443184 | 2162749 | 159 |
| | ECC-163 | 1 | 8975762 | 2569170 | 2361527 | 196 |
| | | 2 | 9339719 | 2131954 | 1559887 | 195 |
| | | 4 | 12606009 | 1728800 | 1605614 | 195 |
| | | 8 | 13644221 | 2107223 | 2214543 | 196 |

6. Conclusion

The paper proposes a new key management scheme for wireless sensor networks using elliptic curve public cryptosystem. The key management scheme uses multiple private keys and a pair of public keys. The key management scheme was tested for performance evaluation using different elliptic curves. The experimental results showed that our scheme performed well in spite of using multiple private keys and a pair of public keys. The key management scheme can also be used in group key management for group authentication. Future enhancement can be done on optimizing elliptic curve operations such as inversion and point multiplications.

7. References

1. Nasir HJ, Ku-Mahamud KR. Wireless Sensor Network: A Bibliographical Survey. Indian Journal of Science and Technology. 2016 Oct 17; 9(38).
2. Ramya R, Saravanakumar G, Ravi S. MAC protocols for wireless sensor networks. Indian Journal of Science and Technology. 2015 Dec 1; 8(34):1. Crossref
3. Roopashree HR, Kanavalli A. SARDS: Secured anonymous routing with digital signature in wireless sensor network. Indian Journal of Science and Technology. 2016 Feb 2; 9(7). Crossref

4. Hendaoui F, Eltaief H, Youssef H, Trad A. CGK: A Collaborative Group Key Management Scheme. High Performance Computing and Communications, 2014 IEEE 6th International Symposium on Cyberspace Safety and Security, 2014 IEEE 11th International Conference on Embedded Software and System (HPCC, CSS, ICESS), 2014 IEEE International Conference. 2014 Aug 20; p. 514–7. Crossref

5. Liu Z, Ma J, Huang Q, Moon S. Asymmetric key pre-distribution scheme for sensor networks. IEEE transactions on wireless communications. 2009 Mar; 8(3):1366–72. Crossref

6. Malan DJ, Welsh M, Smith MD. A public-key infrastructure for key distribution in TinyOS based on elliptic curve cryptography. IEEE SECON 2004. 2004 First Annual IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks. 2004 Oct 4; p. 71–80. Crossref

7. Wander AS, Gura N, Eberle H, Gupta V, Shantz SC. Energy analysis of public-key cryptography for wireless sensor networks. 2005. PerCom 2005, Third IEEE International Conference on Pervasive Computing and Communications. 2005 Mar 8; p. 324–8. Crossref

8. Salam MI, Kumar P, Lee H. An efficient key pre-distribution scheme for wireless sensor network using public key cryptography. IEEE, 2010 Sixth International Conference on Networked Computing and Advanced Information Management (NCM). 2010 Aug 16; p. 402–7.

9. Gura N, Patel A, Wander A, Eberle H, Shantz SC. Comparing elliptic curve cryptography and RSA on 8-bit CPUs. Springer Berlin Heidelberg: International Workshop on Cryptographic Hardware and Embedded Systems 2004 Aug 11; p. 119–32.

10. Washington LC. CRC Press: Elliptic curves: number theory and cryptography. 2008 Apr 3.

11. Hankerson D, Menezes AJ, Vanstone S. Springer Science and Business Media: Guide to elliptic curve cryptography. 2006 Jun 1.