TEST Engineering & Management

November-December 2019
ISSN: 0193-4120 Page No. 1799 – 1810

An Efficient Identity Authentication protocol for layer 2 security sourced on ECC for IoT communications

Iqra Hussain[1], Dr. Nitin Pandey[2], Dr. Mukesh Chandra Negi [3]

[1]Amity Institute of Information Technology,
Amity University Uttar Pradesh,
Noida
iqrahussain4@gmail.com

[2]Assistant Professor
Amity Institute of Information Technology, Amity
University Uttar Pradesh, Noida
npandey@amity.edu

[3]Delivery Manager, Tech
Mahindra Ltd
A7, Sector 64, Noida
MN00330419@techmahindra.com

Article Info
Volume 81
Page Number: 1799 - 1810
Publication Issue:
November-December 2019

Article History
Article Received: *5 March 2019*
Revised: *18 May 2019*
Accepted: *24 September 2019*
Publication: *09 December 2019*

Abstract

In IoT communications we have networks comprising of nodes which are capable of sensing a parameter, processing it and transmitting it. Primarily, the networks are employed in environmental sensing and monitoring. As these networks are data centric, the cryptographic parameters like authentication, confidentiality, integrity and availability are strongly needed. The nodes in such networks are resource constrained in terms of memory, bandwidth and power. Due to this limitation traditional Asymmetric Public Key cryptographic algorithms are not well suited for such networks.

In this paper, the suitability and usage of ECC in these networks has been discussed. Using TinyOS as platform and Tiny ECC, this paper presents how ECC provides a better Asymmetric alternative compared to other PKI's in these networks. The paper also presents a light weight node authentication protocol i.e. an Identity Authentication Protocol using ECC. This protocol has been simulated in TinyOS using Tossim and Simulator.

Keywords: *IoT,WSN (Wireless Sensor Networks); TinyOs; Elliptic Curve Cryptography; TinyECC; IdentityAuthentication Protocol.*

1. Introduction

Architectures that are based on Internet of Things comprise systems and devises that are linked transversely to networks that are heterogeneous in nature and employ various proprietary protocols and basic standards. Networks like these provide services that are powerful but even expose the systems to various threats like message falsification or eavesdropping the messages. Therefore for protection against such risks the Internet of Things devices or systems need to have proper communication security capacities. Therefore if we talk about the different layers of communication protocols, they too need to be defended against various security attacks and threats and similarly a breach in the Data Link Layer may end up disrupting the whole communication and compromise in terms of security.

Information security in these networks is widely unwrapped and fresh subject. The core issues that need to be solved are confidential issues, identity authenticating issues, text integrities along with unmarking issues too. The security coding in such a network is achieved by physical layers, while as the process for encrypting of data frames along with the route information is handled by the data and network layers. Furthermost the key managements and message exchanging is considered by the application layers.

The main broad tools for solving the problems related to security of information are dealt by Public-key cryptosystems and the extensively used cryptosystems are ECC, DSA and RSA. Public key cryptography or Asymmetric cryptography is the most acknowledged and accepted way to solve the problem of information security in networks. The RSA algorithm is one of the most adapted Asymmetric techniques. However, the usage of RSA in these networks is not appropriate because of accepted memory, computing and bandwidth constraints. For using RSA for practical purposes, the recommended key size must be more than 1024 bits. This is due to the fact that there exist sub-exponential algorithms which can break and attack RSA. Due to this limitation other Asymmetric cryptosystems needs to be evaluated for feasibility in low power devices. The main motivation of this work lies in exploring ECC Asymmetric cryptosystems for their usage in such networks. If the security qualities per bit key amongst the present public key cryptosystems are taken in account then accordingly ECC is the one.

The characteristics owed by these cryptosystems constitute small keys, saving bandwidths, faster implementations, less power consumptions, less requirement for hardware's, small public keys and small system parameters. Primary objective is to develop a lightweight node identity authentication protocol based on ECC for effective key sharing in resource constraint in these networks

2. Literature Survey

In recent years it has been a major challenge for the researchers in the field of WSN for reduction of computational complexities, along with minimizing memory usage of memories for the traditional asymmetric cryptographic algorithms like El-Gamal, RSA, DSA and ECC. Among all these algorithms ECC is considered to be most suitable for wireless environments because its memory and resource consumption is least as compared to all others.

2.1 RSA Algorithm

This is one among many algorithms utilized by recent computer systems for the process of encryption and decryption of texts and falls in the category of algorithms that are asymmetric in nature. This indicates that two different keys are used which is also termed as public key cryptography as one of the two keys can be shared by all while as the other key must be reserved as private. This holds its sources from the actuality that the integer factor finding is difficult. This algorithm was named afterRon Rivest,Adi Shamirand Leonard Adleman and in the year 1978 it was described publicly for the first time. When RSA is used at first any client has to generate and further announce the multiplied result of two big prime numbers plus a supplementary value which acts as the public key. Any information regarding the prime factors must be kept undisclosed. For the encryption process the public key can be used by all and when the public key is sufficiently big, the decoding of the texts can only be done when the prime factors are known.

Algorithm

Choose two random prime numbers

1. $P = 61$ and $Q = 53$
2. Compute $n = PQ$
3. $n = 61 * 53 = 3233$
4. Compute$\O(n) = (p-1)(q-1)$
5. $\O(n) = (61-1)(53-1) = 3120$
6. Choose $e > 1$ co prime *to 3120*
7. $e = 17$
8. Choose d to satisfy $de = 1 \ (mod\ \O\ (n))$
9. $d = 2753$
10. $17 * 2753 = 46801 = 1 + 15 * 3120$

Public key is $(n = 3233, e + 17)$.

For a padded message m the encryption function is: $C = m^e \bmod n$ $m^{17}\ mod\ 3233$ Private Key is: $(n = 3233, d = 2753)$. Decryption function is: $m = c^d$ mod$n$ = c^{2753} mod *3233*

For example, to encrypt $m = 123$, we calculate $c = 123^{17}$mod $3233 = 855$

To decrypt $c = 855$, we calculate $m = 855^{2753}$mod $3233 = 123$. *[3]*

.

2.2 Elliptical Curve Cryptography

Elliptic Curve Cryptography (ECC) was discovered in 1985 by Victor Miller (IBM) and Neil Koblitz (University of Washington) as an alternative mechanism for implementing public-key cryptography. In convoluted information systems it's the public key algorithms that are utilized for creating mechanisms that are further put to use for allocating keys amid huge number of entities or participants. Compared to rest of popularly used algorithms for example RSA, ECC becomes hard to defy at corresponding key lengths since it is sourced on discrete logarithms. [4]

2.2.1 ELLIPTIC CURVE GROUPS OVER REAL NUMBERS

If a set of points say x, y are said to fulfil an elliptical curved equation of the type: $y^2 = x^3 + ax + b$, in which x, y, a &b are real numbers, it is called as elliptical curve over real numbers.

The numbers a & b at every choice give a dissimilar elliptical curve, say an equation $y^2 = x^3 - 4x + 0.67$ is obtained when a = -4 and b = 0.67 accordingly. The graphical representation of the curve is given in figure 1. The elliptical curve $y^2 = x^3 + ax + b$ can be utilized for framing a group only when $x^3 + ax + b$ include none recapped factors or else comparably if $4a^3 + 27b^2$ is not equal to 0.

An individual point O that is termed as the infinity point along with other points that are present on corresponding elliptical curves are included in an elliptical curve group over real numbers.

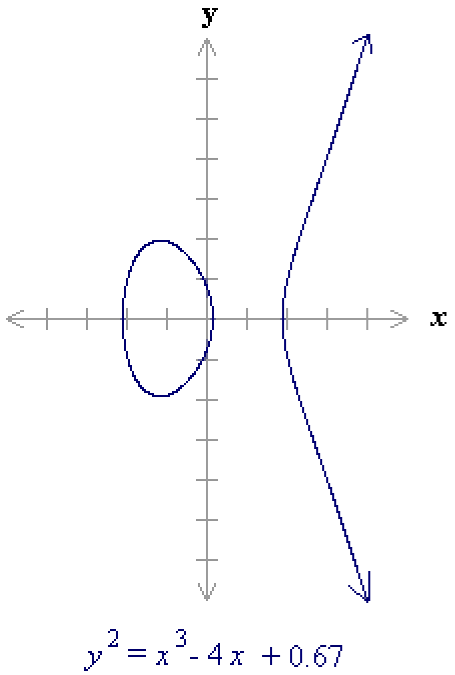


Figure 1: Elliptical Curve

2.2.2 ELLIPTICAL CURVE GROUPS OVER FP

If we consider cryptography an important characteristic constitutes of the groups having finite number of points.
Because of the round of errors the calculation over real numbers is time consuming along with inaccuracy while as crypto graphical application demands quick and accurate arithmetic's; therefore elliptical curve group over finite field of F and F_2m are made use of.
The numbers from 0 to p – 1 are utilized by the fields Fp while as the computation finishes by receiving the remainder on division by p. Taking an example; in F_{24}, field constitutes the integers ranging from 0 to 23 and whichever operations contained in this field will always result in integers as well between 0 and 23.
Elliptical curves through primary fields of Fp are developed only if the variables i.e. a, b are chosen inside the fields of F_p. The elliptical curves constitute the whole points (x,y) that fulfil the elliptical curve equations modulo p (in which x & y are numbers in F_p). Taking an example; y^2 mod p = x^3 + ax + b mod p has a primary field of Fp if a & b are in Fp.

The elliptical curve $y^2 = x^3 + ax + b$ can be utilized for framing a group only when $x^3 + ax + b$ include none recapped factors or else comparably if $4a^3 + 27b^2$ is not equal to 0 An individual point O that is termed as the infinity point along with other points that are present on corresponding elliptical curves are included in an elliptical curve group over real numbers. On such elliptical curves many finite points are present.

Taking an example; considering an elliptical curve on top of the field F_{23}. Together with a = 1 and b = 0, the elliptical curve equation will be $y^2 = x^3 + x$. The points (9,5) satisfy the equation given that:

y^2 mod p = x^3 + x mod p

25 mod 23 = 729 + 9 mod 23

25 mod 23 = 738 mod 23

2 = 2

The 23 points satisfying the equation will be:

(21,17) (21,6) (20,19) (20,4) (19,22) (19,1)

(18,13) (18,10) (17,13) (17,10) (16,15) (16,8)

(15,20) (15,3) (13,18) (13,5) (11,13) (11,10)

(9,18) (9,5) (1,18) (1,5) (0,0)

The graphical representation of the points will be as:

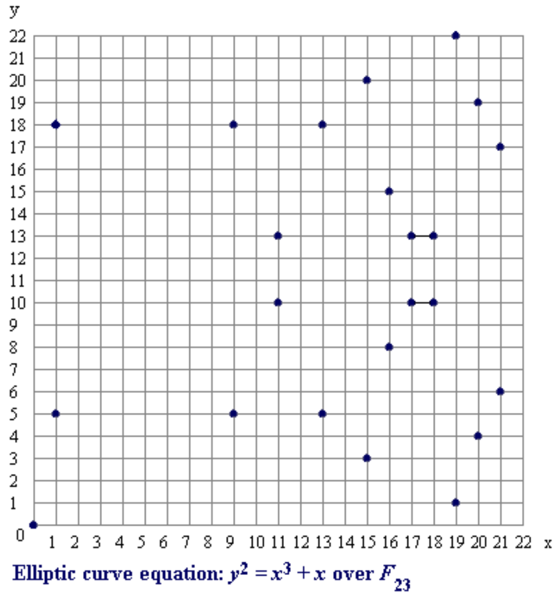


Figure 2: Elliptical Curve Graph

It should be noted that two points exist for every values of x. The graphs look random but its still symmetric for y = 11.5. And we know that in case of elliptical curves over real numbers a negative point exists for every point that reflects on x-axis. For field F_{23}, modulo 23 is taken for components that tend to be negative in y-values that result in positive numbers as differences from 23. In this -P = (x_P,(y_P Mod 23)) [4]

2.2.3 SCALAR MULTIPLICATIONS

Since an elliptical curve group is usually defined in terms of additive notations but some relevance can also be given if the multiplicative notations are used too. In particular the operations for scalar multiplications under additive notations are considered i.e. calculating kP which means the addition of k copies of point P. Now if the multiplicative notation is used which means again multiplying together the k copies of point P resulting in the point P*P*P*P&.*P = Pk [4]

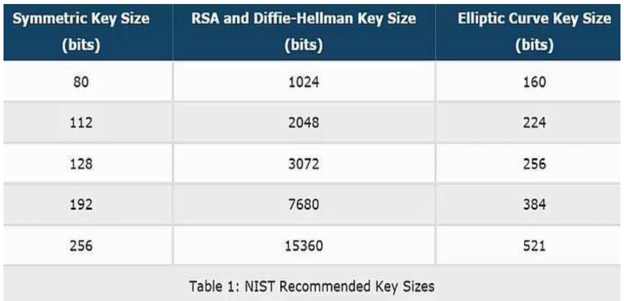
2.2.4 THE ELLIPTICAL CURVE DISCRETE LOGARITHM PROBLEMS

The discrete logarithmic problems in multiplicative set Zp* can be stated as; element r & q of the set, and prime number p, a number k needs to be found so that r=qk mod p. The discrete logarithmic problems in the case when the elliptical curve groups are defined by making use of multiplicative notations ; given The point P & Q given in groups, finding a number as in Pk = Q; k will be known as discrete logarithmic of Q to base P. Further describing elliptical curve groups by making use of additive notations, elliptical curve discrete logarithmic problems are; P & Q as given point in groups, number k needs to be found so that Pk = Q

2.2.5 Comparison between RSA and ECC

ECC tends to be an advancing plus afresh area for working because it tends to be a cost effective approach for performing encryptions for different entities along with securing data, information, image transmissions through internet. Elliptical curve is said to offer better securities using small key size that makes it very constructive in several applications. Due to small size of keys execution time becomes fast for different processes which in turn becomes a positive achievement for systems where in real time performances are important factors. Elliptical curves can also be implemented over Palm OS devices.
[5][6]

The underneath data in the table states the comparisons between ECC and RSA considering number of bits, key generations, security level and performances:



Symmetric Key Size (bits)	RSA and Diffie-Hellman Key Size (bits)	Elliptic Curve Key Size (bits)
80	1024	160
112	2048	224
128	3072	256
192	7680	384
256	15360	521

Table 1: NIST Recommended Key Sizes

Figure 3: RSA vs ECC in bits

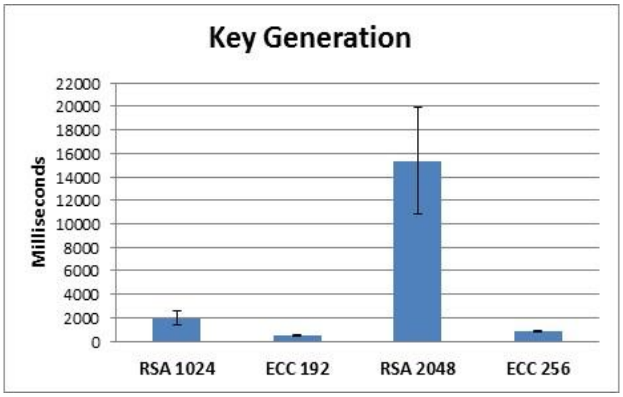


Figure 4 : RSA vs ECC in Key Generation

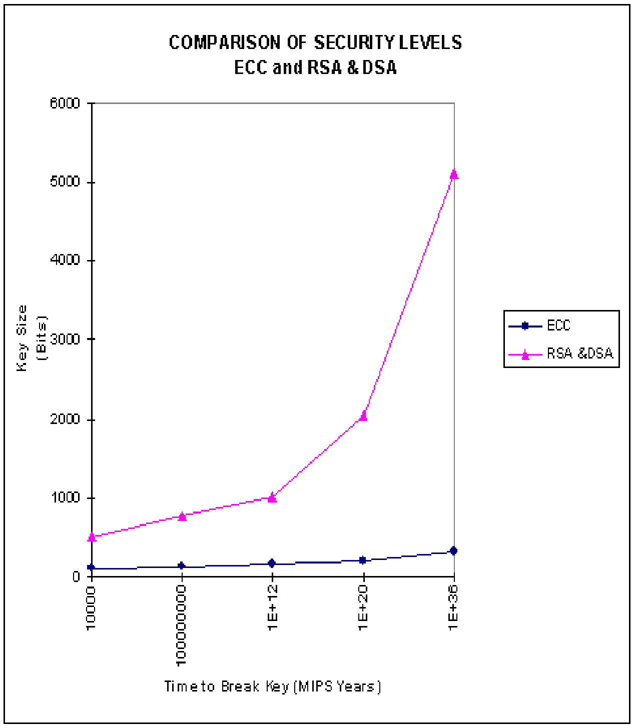


Figure 5: RSA vs ECC in Security Levels

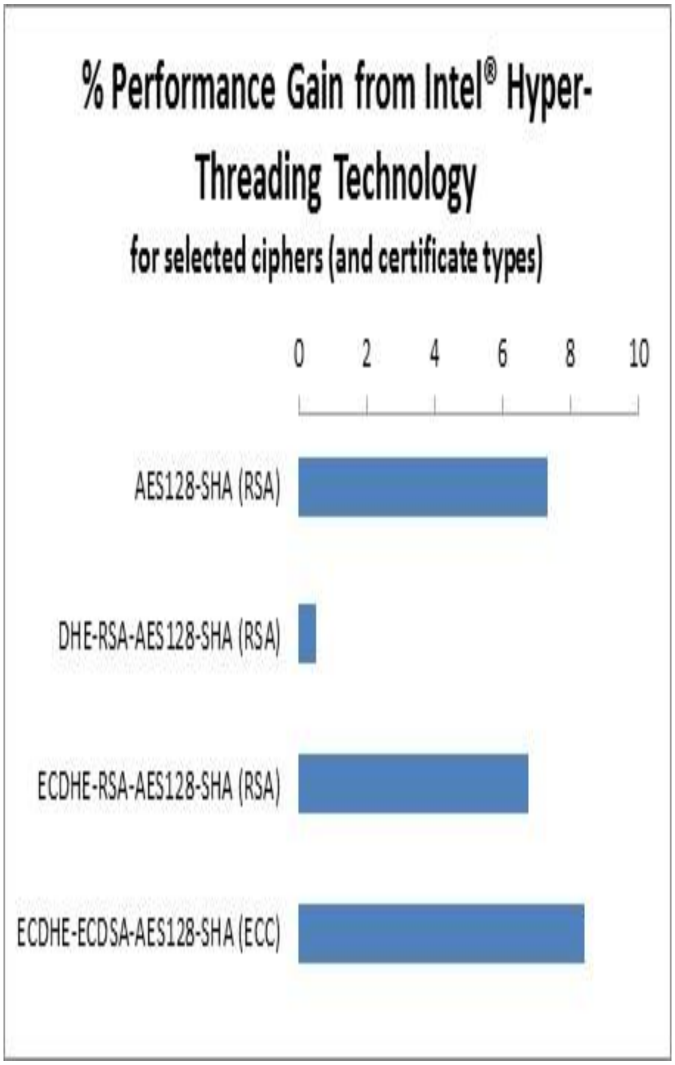


Figure 6: RSA vs ECC in Performance

3. PROPOSITION

We have proposed a light weight Identity Authentication Protocol based on ECC. The usage of ECC in IoT networks has been demonstrated by implementing this light weight identity authentication protocol. The protocol has been developed on TinyOS embedded operating system using NesC Language and simulated on Tossim. Memory requirements of the protocol have also been calculated.

3.1 Proposed Identity Authentication Protocol

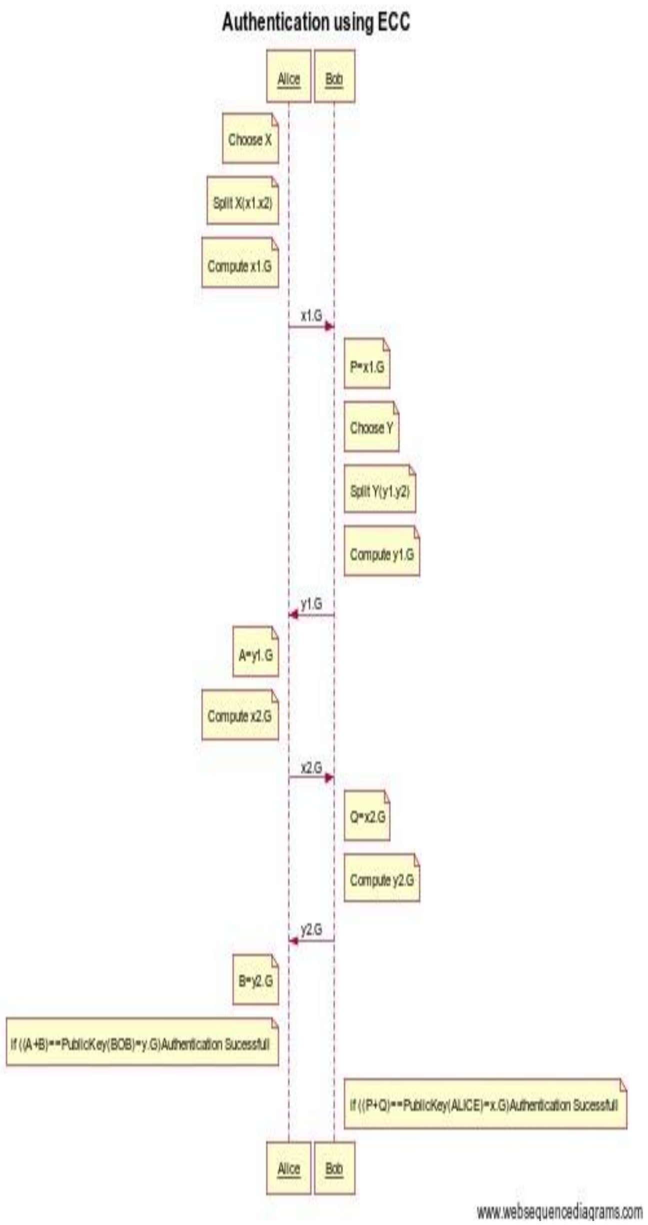


Figure 7 Identity Authentication Protocol

Let's make an assumption that Alice has to set up a key that needs to be shared along with Bob; however the lone route accessible by them might be spied upon by some intruder. Thus the algorithm that has been worked out carries out succession of steps for the establishment of a sharing key amongst two participants and along with including mechanisms for authentication.

3.1.1 Initialize

Prior to running the protocols the system needs to be properly initialized. A private key along with a public key is needed to be allotted for each node. The public key is always taken as a point on the elliptical curve while as the private one is any among random numbers. Assuming, Alice chooses some random number X as its private key, therefore the public key will be X.G, in which G tends to be a generator point on Elliptical Curve. Similarly, Bob too chooses a random number as Y as that tends to be its private key. Therefore the public key will be Y.G. Also each node has the knowledge against the other nodes public keys.

3.1.2 Algorithm

Assuming Alice and Bob have the need for authenticating themselves prior to any communication taking place.

Alice has generated private key X at random and public key X.G.

Bob has generated the private key Y at random and thus public key as Y.G.

This protocol will be separated in these steps accordingly:

Step 1: Alice will split its private key X into integers x1 and x2 using AND masking.

Step 2: Alice will compute x1.G and send it to Bob who receives P = x1.G.

Step 3: Bob will split its private key Y into integer's y1 and y2 using AND masking.

Step 4: Bob will compute y1.G and send it to Alice who receives A = y1.G.

Step 5: Alice computes x2.G and sends it to Bob who receives Q = x2.G.

Step 6: Bob will compute y2.G and send it to Alice who receives B = y2.G.

Step 7: Alice calculates A+B. If (A+B) = Y.G (i.e. public key of Bob), authentication is successful.

Step 8: Bob calculates P+Q. If (P+Q) = X.G (i.e. public key of Alice), authentication is successful.

All this comprises the dual authentication of the identities.

The algorithm can also work for one way authentication.

3.2 Platform

The platform used for implementing this protocol:

3.2.1 TinyOS

This is without plus open software components based operating system also targeting platforms wireless sensor network(WSN). This particular OS tends to form embedding operating systems that is recorded in nesC programmed languages forming cooperation tasks and processes.

Implementation: NesC that itself is a dialect of C language that has been updated against the memory limitations in the case of sensor networks, TinyOS applications are written in this language only. Java along with shell script front-ends forms its main auxiliary tools.

The programs in TinyOS represent various hardware abstractions along with its building blocks as the software components. The shared abstractions like routing, actuations, storages, communications and sensing components and interfaces are provided by TinyOS as well.

.

TinyOS is totally non blocking with just a single stack so as a result every single input output operation lasting more than several hundred micro seconds tend to be asynchronous along with call backs. TinyOS makes use of nesC attributes for the linking of calling events plus callbacks for the purpose of enabling native compilers for improved optimizations through calling boundary. The non block feature makes it possible for TinyOS for upholding higher concurrency with single stack along with holding the programmers for wring complicated logics by compiling altogether tiny event handlers. It provides tasks same as deferring procedure calls or even the interrupting handler bottom half for supporting large computation. In TinyOS components can be posting tasks that can further be scheduled for running afterwards. Here the tasks function in FIFO orders and being nonpreemptive in nature.

The code in TinyOS is statically connected with programming codes plus further compiled to tiny binary numbers by making use of customized toolchains. Related utility is given for the completion of development platforms to work along TinyOS. [7]

3.2.2 NesC Language

This language NesC stands for Network Embedded Systems C that is pronounced as "NES-see". This forms a component-based and event driven programming language for the use of building various applications meant forTinyOSplatforms.

3.2.3 Tossim

All the TinyOS applications are simulated by TOSSIM. Its working is based on the replacement of components along simulated implementation. The components are replaced at a very flexible level like if we consider simulation implementations of millisecond timer which substitutes HilTimerMilliC and we have implementations also for atmega128 platform which substitutes the HPL component of hardware clock. The first one is generalised therefore can also be put to use by any platforms but lack fidelity for the capture of a definite chip performance compared to the second one that does it. Likewise packet level communicating components can also be replaced for packet level simulations and also replacing low level radio chip components to a much accurate simulations of code executions by TOSSIM.

TOSSIM forms a distinct event simulator and while running it extracts event from event queues that are sorted by time and further gives them for execution. The simulated events can either be representing hardware interrupts or higher levelled system events like packet receptions that totally depends on the level of simulations.

3.2.4 TinyECC

TinyECC forms a software packet that provides ECC based PKC operation that will be flexible to configure or even integrate to sensor networking application. It also presents key exchanging protocols like ECDH, digital signature schemes like ECDSA, plus public key encrypting schemes like ECIES too. It also makes use of various optimizing switching techniques that can further turn particular optimizations on or off in relevance to the needs of a developer. [10]

4. EXPERIMENTAL RESULTS

4.1 Simulation and Results

The Tossim Simulator has ben used for carrying out simulation. The outputs of various calculation and transmissions in the protocol have been captured using DBG flags. TOSSIM helps to give the configurations for identification and rectification of outputs during run time. Most of TinyOS basics contain coding for rectifying records. Every record that gets amended will be accompanied either by a single or many modal flags. When simulator starts, it first comprehends the DBG domain variables for determining which mode needs to be enable. In our simulations DBG variable has been set to USR1.

4.1 Proposed Identity Authentication Protocol

The Screenshot of TinyViz is shown below in Figure 9 .The screenshot depicts 2 motes as Alice (Mote 0) and Bob (Mote 1).

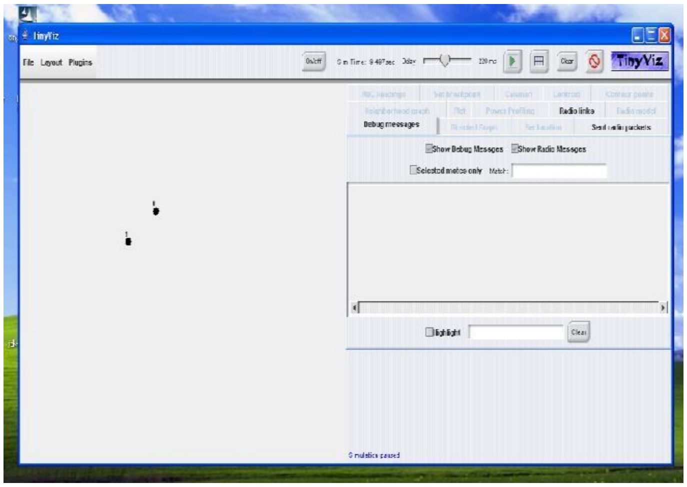
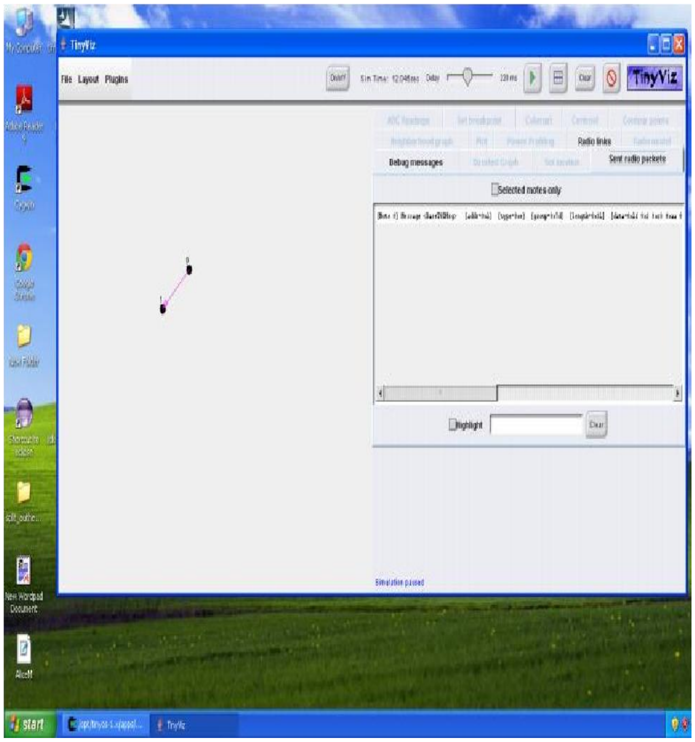


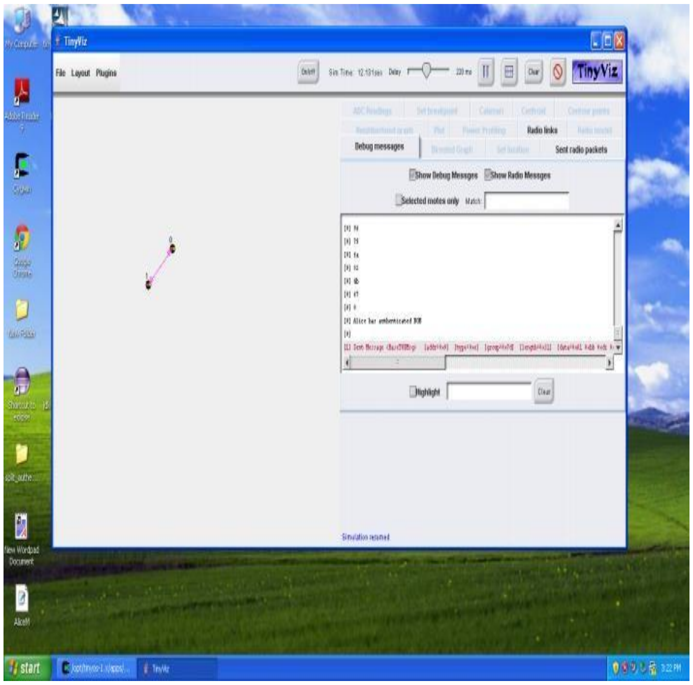
Figure 8 :Alice (Mote 0) and Bob(Mote 1) in TinyViz

The below depicted Figure 9 shows that the motes are executing process of the transferring packets, where Alice has sent 1 packet as shown in Debug Messages column in the centre right of the Tinyviz window.



```
[Mote 0] Message <BaseTOSMsg> [addr=0x0] [type=0xe] [group=0x7d] [length=0x1b] [data=0x0 [illegible]
```

Figure 9: Transmission of packets



```
[0] [illegible]
[0] [illegible]
[0] [illegible]
[0] [illegible]
[0] [illegible]
[0] [illegible]
[0] [illegible]
[0] Alice has authenticated BOB
[0]
[1] Sent Message <BaseTOSMsg> [addr=0x0] [type=0xe] [group=0x7d] [length=0x1b] [data=0x0 [illegible]
```

Figure 11: Alice authenticated Bob

The Complete Simulation output of the protocol The Figure 10 shows that the Bob has authenticated taken through command shell is: Alice and thus the Led's have turned on. Here, we can say the protocol has reached the One way identity

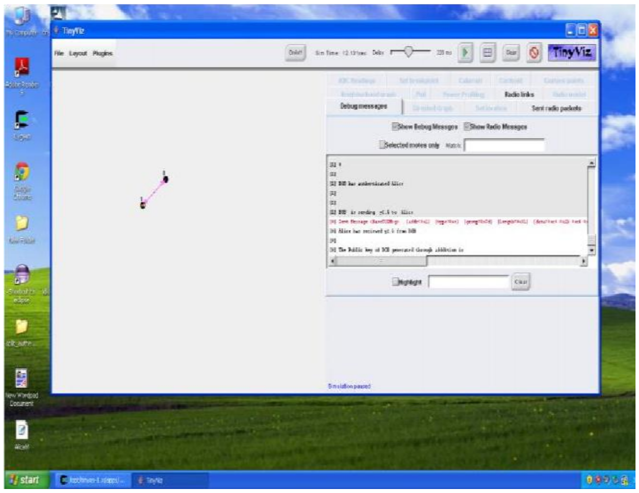
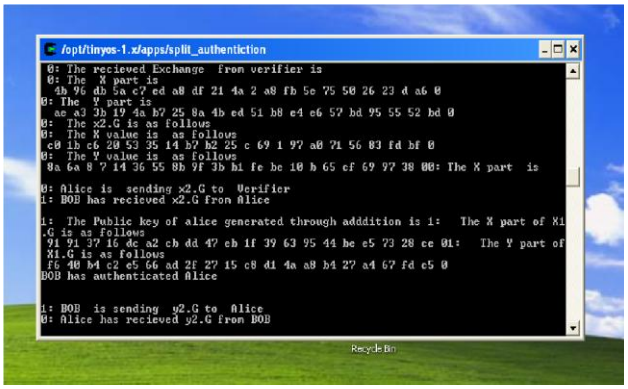


Figure 8 Bob authenticated Alice

In figure 11, transmission of packets is over and thus both of the motes have turned on led's and the direct communication line in pink is displayed. This signifies the protocol has worked for identification of both motes. The mote 0 i.e. Alice has authenticated Bob and mote 1 i.e. Bob has authenticated Alice .

authentication for mote 0 i.e. Alice.



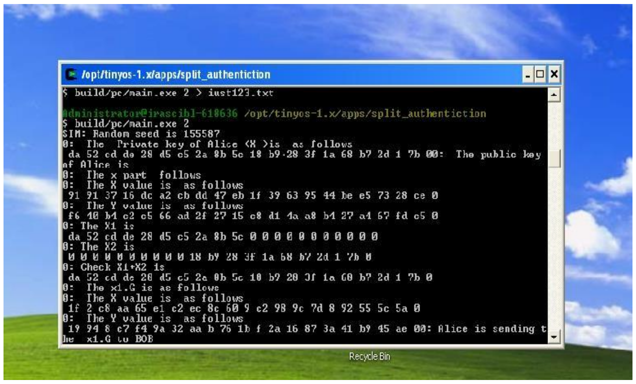
```
/opt/tinyos-1.x/apps/split_authentiction
 0: The recieved Exchange  from verifier is
 0: The  X part is
  4b 96 db 5a c7 ed a8 df 21 4a 2 a8 fb 5e 75 50 26 23 d a6 0
0: The  Y part is
  ae a3 3b 19 4a b7 25 8a 4b ed 51 b8 e4 e6 57 bd 95 55 52 bd 0
0:  The x2.G is as follows
0:  The X value is  as follows
 c0 1b c6 20 53 35 14 b7 b2 25 c 69 1 97 a0 71 56 83 fd bf 0
0:  The Y value is  as follows
 8a 6a 8 7 14 36 55 8b 9f 3b b1 fe be 10 b 65 cf 69 97 38 00: The X part  is

0: Alice is  sending x2.G to  Verifier
1: BOB has recieved x2.G from Alice

1:  The Public key of alice generated through adddition is 1:   The X part of X1
.G is as follows
 91 91 37 16 dc a2 cb dd 47 eb 1f 39 63 95 44 be e5 73 28 ce 01:   The Y part of
 X1.G is as follows
 f6 40 b4 c2 e5 66 ad 2f 27 15 c8 d1 4a a8 b4 27 a4 67 fd c5 0
BOB has authenticated Alice


1: BOB  is sending  y2.G to  Alice
0: Alice has recieved y2.G from BOB
```

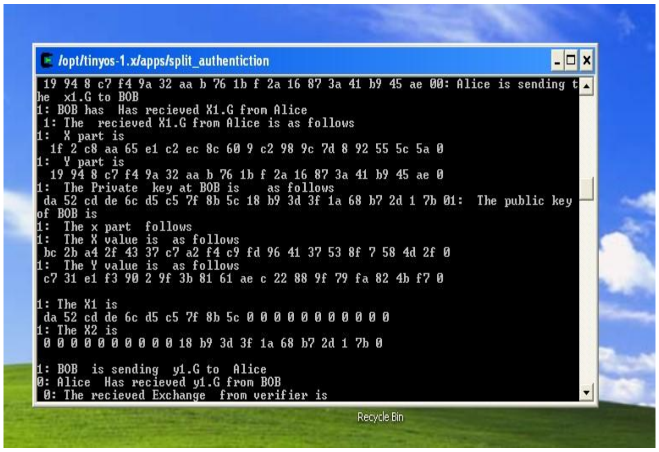
Figure 11 cygwin shell output "c"(continued)



```
/opt/tinyos-1.x/apps/split_authentiction
$ build/pc/main.exe 2 > iust123.txt

Administrator@irascibl-618636 /opt/tinyos-1.x/apps/split_authentiction
$ build/pc/main.exe 2
SIM: Random seed is 155587
0:  The  Private key of Alice <X >is  as follows
 da 52 cd de 28 d5 c5 2a 8b 5c 18 b9 28 3f 1a 68 b7 2d 1 7b 00:  The public key
of Alice is
0:  The x part  follows
0:  The X value is  as follows
 91 91 37 16 dc a2 cb dd 47 eb 1f 39 63 95 44 be e5 73 28 ce 0
0:  The Y value is  as follows
 f6 40 b4 c2 e5 66 ad 2f 27 15 c8 d1 4a a8 b4 27 a4 67 fd c5 0
0: The X1 is
 da 52 cd de 28 d5 c5 2a 8b 5c 0 0 0 0 0 0 0 0 0 0 0
0: The X2 is
 0 0 0 0 0 0 0 0 0 0 18 b9 28 3f 1a 68 b7 2d 1 7b 0
0: Check X1+X2 is
 da 52 cd de 28 d5 c5 2a 8b 5c 18 b9 28 3f 1a 68 b7 2d 1 7b 0
0:  The x1.G is as follows
0:  The X value is  as follows
 1f 2 c8 aa 65 e1 c2 ec 8c 60 9 c2 98 9c 7d 8 92 55 5c 5a 0
0:  The Y value is  as follows
 19 94 8 c7 f4 9a 32 aa b 76 1b f 2a 16 87 3a 41 b9 45 ae 00: Alice is sending t
he  x1.G to BOB
```

Figure 9 cygwin shell output "a"(continued)

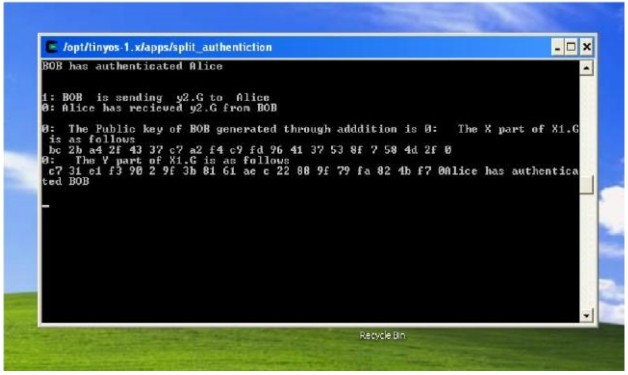


```
/opt/tinyos-1.x/apps/split_authentiction
 19 94 8 c7 f4 9a 32 aa b 76 1b f 2a 16 87 3a 41 b9 45 ae 00: Alice is sending t
he  x1.G to BOB
1: BOB has  Has recieved X1.G from Alice
 1: The  recieved X1.G from Alice is as follows
1:  X part is
  1f 2 c8 aa 65 e1 c2 ec 8c 60 9 c2 98 9c 7d 8 92 55 5c 5a 0
1:  Y part is
  19 94 8 c7 f4 9a 32 aa b 76 1b f 2a 16 87 3a 41 b9 45 ae 0
1:  The Private  key at BOB is    as follows
 da 52 cd de 6c d5 c5 7f 8b 5c 18 b9 3d 3f 1a 68 b7 2d 1 7b 01:  The public key
of BOB is
1:  The x part  follows
1:  The X value is  as follows
 bc 2b a4 2f 43 37 c7 a2 f4 c9 fd 96 41 37 53 8f 7 58 4d 2f 0
1:  The Y value is  as follows
 c7 31 e1 f3 90 2 9f 3b 81 61 ae c 22 88 9f 79 fa 82 4b f7 0

1: The X1 is
 da 52 cd de 6c d5 c5 7f 8b 5c 0 0 0 0 0 0 0 0 0 0 0
1: The X2 is
 0 0 0 0 0 0 0 0 0 0 18 b9 3d 3f 1a 68 b7 2d 1 7b 0

1: BOB  is sending  y1.G to  Alice
0: Alice  Has recieved y1.G from BOB
 0: The recieved Exchange  from verifier is
```

Figure 10 cygwin shell output "b"(continued)



```
/opt/tinyos-1.x/apps/split_authentiction
BOB has authenticated Alice


1: BOB  is sending  y2.G to  Alice
0: Alice has recieved y2.G from BOB

0:  The Public key of BOB generated through adddition is 0:   The X part of X1.G
 is as follows
 bc 2b a4 2f 43 37 c7 a2 f4 c9 fd 96 41 37 53 8f 7 58 4d 2f 0
0:   The Y part of X1.G is as follows
 c7 31 e1 f3 90 2 9f 3b 81 61 ae c 22 88 9f 79 fa 82 4b f7 0Alice has authentica
ted BOB

_
```

Figure 12 cygwin shell output "d"

The Complete Simulation output of the protocol taken through redirection of above command shell is shown below:

SIM: Random seed is 155587

```
0:  The Private key of Alice (X) is as follows

 da 52 cd de 28 d5 c5 2a 8b 5c 18 b9 28 3f 1a 68 b7 2d 1
7b 00:

The public key of Alice is

0:  The x part follows

0:  The X value is as follows

 91 91 37 16 dc a2 cb dd 47 eb 1f 39 63 95 44 be e5
73 28 ce 0 0:  The Y value is as

follows

 f6 40 b4 c2 e5 66 ad 2f 27 15 c8 d1 4a a8 b4 27 a4 67 fd
c5 0

0: The X1 is

  da 52 cd de 28 d5 c5 2a 8b 5c 0 0 0 0 0 0 0 0 0 0 0

  0: The X2 is

   0 0 0 0 0 0 0 0 0 0 18 b9 28 3f 1a 68 b7 2d 1 7b 0

  0: Check X1+X2 is

   da 52 cd de 28 d5 c5 2a 8b 5c 18 b9 28 3f 1a 68 b7 2d
  1 7b 0

  0:  The x1.G is as follows

  0:  The X value is  as follows

  1f 2 c8 aa 65 e1 c2 ec 8c 60 9 c2 98 9c 7d 8 92 55 5c
  5a 0

  0:  The Y value is as follows

   19 94 8 c7 f4 9a 32 aa b 76 1b f 2a 16 87 3a 41 b9 45
  ae 00:

  Alice is sending the x1.G to BOB

  1: BOB has has received X1.G from Alice

   1: The received X1.G from Alice is as follows

  1:  X part is

    1f 2 c8 aa 65 e1 c2 ec 8c 60 9 c2 98 9c 7d 8 92 55
  5c 5a 0

  1:  Y part is

  19 94 8 c7 f4 9a 32 aa b 76 1b f 2a 16 87 3a 41 b9 45
 ae 0

 1:  The Private Key at BOB is    as follows

  da 52 cd de 6c d5 c5 7f 8b 5c 18 b9 3d 3f 1a 68 b7 2d 1
 7b 01:

 The public key of BOB is

 1:   The x part follows 1:   The

 X value is as follows

 bc 2b a4 2f 43 37 c7 a2 f4 c9 fd 96 41 37 53 8f 7 58 4d
 2f 0

 1:  The Y value is as follows

  c7 31 e1 f3 90 2 9f 3b 81 61 ae c 22 88 9f 79 fa 82 4b
 f7 0

 1: The X1 is

da 52 cd de 6c d5 c5 7f 8b 5c 0 0 0 0 0 0 0 0 0 0 0

1: The X2 is

0 0 0 0 0 0 0 0 0 0 18 b9 3d 3f 1a 68 b7 2d 1 7b 0

1: BOB is sending y1.G to Alice

0: Alice has received y1.G from BOB

 0: The received Exchange from verifier is

 0: The X part is

  4b 96 db 5a c7 ed a8 df 21 4a 2 a8 fb 5e 75 50 26 23 d
a6 0 0: The Y part is

  ae a3 3b 19 4a b7 25 8a 4b ed 51 b8 e4 e6 57 bd 95 55
52 bd 0

0:    The  x2.G  is  as  follows  0:

The X value is as follows

 c0 1b c6 20 53 35 14 b7 b2 25 c 69 1 97 a0 71 56 83 fd
bf 0

0:  The Y value is as follows

 8a 6a 8 7 14 36 55 8b 9f 3b b1 fe be 10 b 65 cf69 97 38
00:

 The X part is
```

0: Alice is sending x2.G to Verifier

1: BOB has received x2.G from Alice

1: The Public key of alice generated through adddition is 1:

The X part of X1.G is as follows

91 91 37 16 dc a2 cb dd 47 eb 1f 39 63 95 44 be e5 73 28 ce 01: The Y part of X1.G is as follows

f6 40 b4 c2 e5 66 ad 2f 27 15 c8 d1 4a a8 b4 27 a4 67 fd c5 0

BOB has authenticated Alice

1: BOB is sending y2.G to Alice

0: Alice has recieved y2.G from BOB

0: The Public key of BOB generated through adddition is 0: The X part of X1.G

is as follows

bc 2b a4 2f 43 37 c7 a2 f4 c9 fd 96 41 37 53 8f 7 58 4d 2f 0

0: The Y part of X1.G is as follows

c7 31 e1 f3 90 2 9f 3b 81 61 ae c 22 88 9f 79 fa 82 4b f7 0

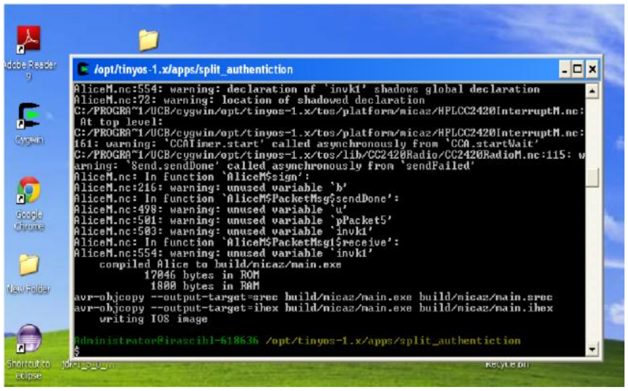
Alice has authenticated BOB

5. PERFORMANCE ANALYSIS

We know that the low end devices in WSN posses controlled resources. The functioning forms a significant aspect. Efficiency of the proposed Identity Authentication Protocol lies in the operation of scalar multiplication.
Here together A & B require calculating scalar multiplications twice. This consumption becomes acceptable for two way authentications. Since elliptical curve discrete logarithmic problems can be solved therefore the security will always be guaranteed as the protocols itself functions on elliptical curve discrete logarithms. Also as the numbers in initial steps are generated randomly hence it is confirmed that the playback attacks will be defended by the protocol.

Also, the protocol has been memory and time efficient as we have calculated the total memory in terms of RAM and ROM used and total time taken by the protocol for complete dual mode authentication.

The screenshots below in figure 16 and figure 17 depicts the memory consumption and total time taken by the protocol.

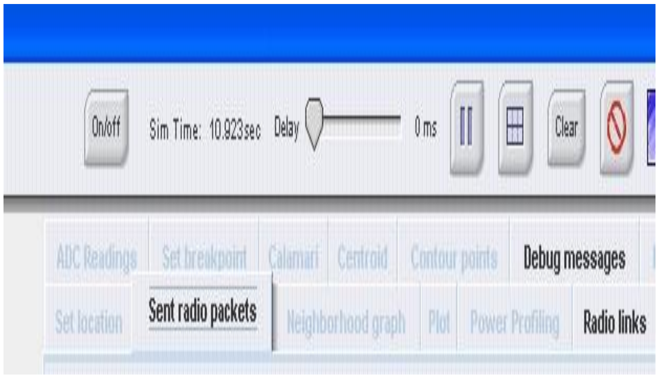


/opt/tinyos-1.x/apps/split_authentiction

```
AliceM.nc:554: warning: declaration of `invk1' shadows global declaration
AliceM.nc:72: warning: location of shadowed declaration
C:/PROGRA~1/UCB/cygwin/opt/tinyos-1.x/tos/platform/micaz/HPLCC2420InterruptM.nc:
 At top level:
C:/PROGRA~1/UCB/cygwin/opt/tinyos-1.x/tos/platform/micaz/HPLCC2420InterruptM.nc:
161: warning: `CCATimer.start' called asynchronously from `CCA.startWait'
C:/PROGRA~1/UCB/cygwin/opt/tinyos-1.x/tos/lib/CC2420Radio/CC2420RadioM.nc:115: w
arning: `Send.sendDone' called asynchronously from `sendFailed'
AliceM.nc: In function `AliceM$sign':
AliceM.nc:216: warning: unused variable `b'
AliceM.nc: In function `AliceM$PacketMsg$sendDone':
AliceM.nc:498: warning: unused variable `u'
AliceM.nc:501: warning: unused variable `pPacket5'
AliceM.nc:503: warning: unused variable `invk1'
AliceM.nc: In function `AliceM$PacketMsg1$receive':
AliceM.nc:554: warning: unused variable `invk1'
    compiled Alice to build/micaz/main.exe
            17046 bytes in ROM
             1800 bytes in RAM
avr-objcopy --output-target=srec build/micaz/main.exe build/micaz/main.srec
avr-objcopy --output-target=ihex build/micaz/main.exe build/micaz/main.ihex
    writing TOS image

Administrator@irascibl-618636 /opt/tinyos-1.x/apps/split_authentiction
$
```

Figure 13 : Memory Consumption in RAM, ROM



Sim Time: 10.923sec Delay 0 ms

Fig 14: Simulation Time

6. CONCLUSION AND FUTURE SCOPE

This paper first introduced the use of ECC in IoT networks based on Wireless Sensor Networks. A lightweight identity authentication protocol based on Tiny ECC has been designed for use.

The simulation result shows that Elliptical Curve Cryptosystems can run on Tiny OS. Its distinctiveness is appropriate for limitedly resourced mobile devices and low end devices such as small embedded devices.

The developments of comprehensive ECC systems on WSN are accurate. Bilinear maps can be build from elliptical curves.

Our next step will further be researching on making use of ECC for enhancing counter attacks on this Identity Authentication Protocols in Wireless Sensor Networks.

References

1. https://en.wikipedia.org/wiki/Wireless_sensor_network
2. https://www.nics.uma.es/sites/default/files/papers/Lopez2009.pdf
3. https://en.wikipedia.org/wiki/RSA_(cryptosystem)
4. https://www.certicom.com/ecc
5. Comparison of ECC and RSA Algorithm in Resource
6. Constrained Devices/IEEE paper/DOI:10.1109/ICITCS.2013.6717816/Dated/16-18 Dec. 2013
7. Comparison of ECC and RSA algorithm in multipurpose smart card application/IEEE paper/DOI: 10.1109/CyberSec.2012.6246121/Dated/26-28 June 2012
8. https://en.wikipedia.org/wiki/TinyOS
9. nesC: https://en.wikipedia.org/wiki/NesC
10. Tossim: http://tinyos.stanford.edu/tinyoswiki/index.php/TOSSIM
11. [10]tinyecc:http://discovery.csc.ncsu.edu/software/TinyECC/

Mrs. Iqra Hussain has received her M.Sc. Information Technology from Islamic University of Science and Technology Jammu and Kashmir and B.sc Information Technology from Kashmir University, India. Her major areas of interests include Network security in Internet of Things. She has research publications in areas of Network security, cryptography, IOT and layer two protocols published in International Conferences. She has consistently maintained percentage above distinction throughout her academic career. She is currently pursuing her PhD as a full time Research Scholar from Amity University Noida, India.

Dr. Nitin Pandey was born in India. He is an Assistant Professor at Amity Institute of Information Technology, Amity University Uttar Pradesh. His area of research is Coding theory, Cryptography and Network Security. He is completed his Ph.D. from Mewar University Chittorgarh. He has done B.Sc. and M.Sc. in Mathematics from Deen Dayal Upadhaya University Gorakhpur Uttar Pradesh. He has completed Master of Computer Application from Maharishi Dayanand University Rohtak Haryana. He is CISCO Certified Instructor. He is the author and co-author of more than 20 publications in technical journals and conferences.

Dr. Mukesh Negi is a Ph. D. (Comp. Sc.) from JJT University, Rajasthan, Masters of Computer Applications (MCA) from M.D University, Rohtak, Haryana, M.Sc. (Computer Science) from M.D University, Rohtak, Haryana, B.Sc. (PCM) from Kumaon University, Nainital, Uttarakhand. He is working as Sr. Technical Project Manager, India Business Group with IT MNC TechMahindra Ltd, Noida, India and has a total IT Industry Experience of 16+ Years.