Optimized Elliptic Curve Cryptography as Fine Balance for Wireless Sensor Network

Arpit and Ashwini Kumar

Abstract—**Wireless Sensor Network communication is naturally unreliable and can cause communication packets to be damaged or dropped. This unpredictability in communication poses additional threats to the nodes if dropped packets are taken over by adversaries. Optimized Elliptic curve cryptography (O-ECC) can assist more secure towards WSN security and improved protocol design. Optimized Elliptic curve cryptography is not only emerged as an attractive public key crypto-system for mobile / wireless environments but also provides bandwidth savings. This paper presents a light security algorithm i.e. Optimized ECC which is enhancement of traditional Elliptic Curve Cryptography for wireless sensor network.**

Index Terms—**Cryptography, ECC, security, wireless sensor.**

I. Introduction

Speedy technological enhancement in the areas of micro electro-mechanical systems and miniaturization has encouraged the development of a new kind of network. This network is comprised of small-scale, relatively low in price sensors adequate to intelligent sensing. Sensor network envision a future in which thousands to millions of tiny sensor nodes will be engrafted in almost every aspect of life. The intention is to create an intelligent environment which is adequate to collecting massive amounts of relevant information, acknowledging significant events automatically, and reacting suitably. Without a doubt security schemes optimized for wireless sensor networks have not been fully developed. Current techniques face weaknesses in certain situations, as aggregation or routing aspects prevent top efficiency. The ad hoc nature of sensor networks poses unique challenges regarding their security and reliability. The limited memory, power, processing abilities, and low coverage of the sensor nodes makes them vulnerable to intrusion, interception, modification and fabrication so traditional security techniques cannot ensure confidentiality, integrity, reliability and availability.

Sensor devices, also addressed motes or nodes, typically consist of a sensing unit, a transceiver/communicating unit, a processing unit, and a power supply unit. Depending on the type of application, the sensing unit may supervise diverse types of data including acoustic, seismic, visual, and temperature data. The transceiver unit consists of a short-range RF circuit that performs data transmission and reception for a short range (tens of meters). The processing unit consists of memory and a processor with a severely constrained size and speed. Wireless sensor motes are powered by a battery energy source which is impossible to replace or recharge in most application scenarios. Designers desire to bulk produce nodes for a very low cost per device and deploy them generously as disposable devices. Communication normally consists of source nodes which sense the data and send it to sink node over multiple hops. Sink nodes may be an ordinary sensor nodes or specialized based stations with larger resources. Wireless communication is inherently unreliable and can cause packets to be damaged or dropped. This unreliability in communication poses additional threats to the nodes if dropped packets are taken over by adversaries. Optimized Elliptic curve cryptography (O-ECC) can assist more secure towards WSN security and better protocol design. In following sections contain introduction to Elliptic curve cryptography and the propose method on the mechanism to Optimized it and simulate it on Tiny OS [4].

Manuscript received July 15, 2011; revised August 8, 2011. (Write the date on which you submitted your paper for review).

The authors are with the IERT, 26 Chatham lines, Prayag Allahabad (U.P.) INDIA.(e-mail:arpittabelabux@gmail.com),(e-mail:simplyashwini@gmail.com).

II. Elliptic Curve Cryptography

Elliptic curve cryptography (ECC) [1] is an approach intended to deal public-key cryptography which is founded on the mathematics of elliptic curves. It offers fast decryption and digital signature processing by using Elliptic Curve DSA (ECDSA) [2] and key establishment by using Elliptic Curve Diffie-Hellman (ECDH) [3]. The main advantage of ECC is that under certain situations it applies smaller keys than other methods such as RSA while offering a same or higher level of security. ECC employs points on an elliptic curve to derive a 160-bit public key which is same as in strength to a 1024-bit RSA key. Hence smaller numbers of key contribute to faster key operation and less memory overhead. It is said to be ideal for resource-constrained devices because it provides more "security per bit" than other types of asymmetric cryptography in lesser cost.

A. Elliptic Curve: Its Derivation and Use

Note that elliptic curves are not ellipses. They are so named because of the fact that ellipses are formed by quadratic curves. Elliptic curves are always cubic and have a relationship to elliptic integrals in mathematics [10] where the elliptic integral can be used to determine the arc length of an ellipse. An elliptic curve in its "standard form" is described by

$$y^2 = x^3 + ax + b$$

For the polynomial $x^3 + ax + b$, the discriminant can be given as $D = -(4a^3 + 27b^2)$

This discriminant must not become zero for an elliptic curve polynomial $x^3 + ax + b$ to possess three distinct roots.

If the discriminate is zero, that would imply that two or more roots have coalesced, giving the curves in singular form. It is not safe to use singular curves for cryptography as they are easy to crack. Due to this reason we generally take non-singular curves for data encryption.

B. Elliptic Curves over F^{2m}

What makes the binary finite fields more convenient for hardware implementation is that the elements of GF (2^m) [12] can be represented by m-bit binary codeword. The addition operation in GF (2^m) is like the XOR operation on bit fields.

That is $x + x = 0$ for all $x \in GF(2^m)$.

This implies that a finite field of form GF (2^m) is of characteristic 2. The equation of the elliptic curve on a binary field F_2^m is

$$y^2 + xy = x^3 + ax^2 + b, \text{ where } b \in 0.$$

Here the elements of the finite field are integers of length maximum m bits. These numbers can be considered as a binary polynomial of degree m – 1. In binary polynomial the coefficients can only be 0 or 1.The addition of two points on a curve over is F_2^m defined as

$$(x_1,y_1) + (x_2,y_2) = (x_3,y_3)$$

where $(x_3,y_3) = (\alpha^2 + \alpha + x_1+x_2 +a , \alpha (x_1+x_3)+x_3+y_1)$

where $\alpha = (y_1+y_2)/(x_1+x_2)$

C. Polynomial Arithmetic

Elliptic curve over field F^{2m} involves arithmetic of integer of length m bits. These numbers can be considered as binary polynomial of degree m – 1. The binary string $(a_{m-1} \ldots a_1 a_0)$ can be expressed as polynomial

$$Am\text{-}1xm\text{-}1 + a_{m-2}x^{m-2} + \ldots + a_2x^2 + a_1x + a_0$$

where a_i= 0 or1. For e.g., a 4 bit number 1101_2 can be represented by polynomial as $x^3 + x^2 + 1$. Similar to the modulus p on modular arithmetic, there is an irreducible polynomial of degree m in polynomial arithmetic. If in any operation the degree of polynomial is greater than or equal to m, the result is reduced to a degree less than m using irreducible polynomial also called as reduction polynomial. In binary polynomial representation the coefficients of the polynomial can be either 0 or 1. If in any operation the coefficient becomes greater than 1, it can be reduced to 0 or 1 by modulo 2 operations on the coefficient. All the operations below are defined in field F_2^4 are on irreducible polynomial $f(x) = x^4 + x + 1$. Since here m = 4 the operation involves polynomial of degree 3 or lesser.

1) Addition

Consider two polynomial $A = x^3 + x^2 + 1$ and $B=x^2+x$.

On polynomial addition $A + B$ gives $x^3 + 2x^2 + x+ 1$.

Taking mod 2 over coefficients, $A + B = x^3 + x + 1$.

On binary representation A = 1101_2, B = 0110_2

$A + B = 1011_2$ which is an *XOR* operation between A and B, this is true in all cases. Hence addition of two polynomials can be achieved by simple *XOR* of two numbers.

I.e. $A + B = A \text{ XOR } B$

2) Multiplication

Consider two polynomial $A =x^3 + x^2 + 1$ and $B = x^2 + x$.

On polynomial multiplication A *B gives

$$x^5 + x^3 + x^2 + x.$$

Coefficients are reduced to mod 2. Since m = 4 the results are to be reduces to a degree less than 4 by irreducible polynomial $x^4 + x + 1$.

$$\text{i.e. } x^5 + x^3 + x^2 + x \text{ (mod } f(x))$$
$$= (x^4 + x + 1)x + x^5 + x^3 + x^2 + x$$
$$= 2x^5 + x^3 + 2x^2 + 2x$$

$= x^3$, on reducing the coefficient on mod 2

On binary representation

$$A = 1101_2, B = 0110_2$$
$$A * B = 1000_2$$

For multiplication the algorithm have following steps

Step1:-Let P be a point on elliptic curve and k is the number for multiplication, then

$$k = \sum_{i=0}^{n-1} b_i 2^i$$

Step2:-Convert P to projective P1

Step3:-Set Q1 = P1

Step4:- For I from m-2 down to 0

Set Q1=Q1+Q1

If bi =1 then

Set Q1=Q1+P1

Step5:-Convert Q1 to affine Q

Step6:-Return Q

By this algorithm we can get the result *kP*=Q.

3) Irreducible polynomial

If in any polynomial arithmetic operation the resultant polynomial is having degree greater than or equal to m, it is reduced to a polynomial of degree less than m by the irreducible polynomial. NIST recommended curves m∈ {113,131,163,193,233,239,283,409,571} with the following irreducible functions.

F_2113 $f(x) = x^{113} + x^9 + 1$
F_2131 $f(x) = x^{131} + x^8 + x^3 + x^2 + 1$
F_2163 $f(x) = x^{163} + x^7 + x^6 + x^3 + 1$
F_2193 $f(x) = x^{193} + x^{15} + 1$
F_2233 $f(x) = x^{233} + x^{74} + 1$
F_2239 $f(x) = x^{239} + x^{36} + 1$
F_2283 $f(x) = x^{283} + x^{12} + x^7 + x^5 + 1$
F_2409 $f(x) = x^{409} + x^{87} + 1$
F_2571 $f(x) = x^{571} + x^{10} + x^5 + x^2 + 1$

D. Why Elliptic Curve Cryptography

It has been claimed by some researchers that public key cryptosystems are not viable to implement in these tiny devices because they are resource constrained but this is not true. Let‘s see some of the features of elliptic curve cryptography (ECC) and later see the justification so as to need this in the sensor networks.

1. ECC offers considerably greater security for a given key size.
2. The smaller key size also makes possible much more compact implementations for a given level of security, which means faster cryptographic operations, running on smaller chips or more compact software. This means less heat production and less power consumption — all of which is of particular advantage in constrained devices, but of some advantage anywhere else.
3. There are extremely efficient, compact hardware implementations available for ECC exponentiation operations, offering potential reductions in implementation footprint even beyond those due to the smaller key length alone.

In short: asymmetric cryptography is demanding but looking at the cryptosystem for more security per bit, ECC is a better choice.

E. Optimized-Elliptic Curve Cryptography

Traditional Elliptic Curve cryptography [5] is not optimized one for resource constraint sensor nodes because it adds extra overhead in terms of computation and memory cost. So there is need for optimization, We have considered that problem and optimized the existed elliptic curve cryptography Optimization can be achieved by simplifying the calculations such as modular multiplication, or by reducing number of steps required for point addition and point doubling. Next section elaborates optimizations which are done to reduce complexity while maintaining same security level as ECC.

F. Optimizations for Large Integer Operations

1) Barrett Reduction [6]

As we know that public-key cryptosystems Elliptic curve Cryptography is based mainly on modular operations (modular multiplication and modular exponentiation) of very large integers, ranging on the order of 38-616 decimal digits, or 128-2048 binary bits. Performing computation of numbers of this large size with multiple precisions is not easy or fast to implement. Most methods rely on modular reduction algorithm functions to reduce the size and complexity of the required arithmetic operations to carry out their public-key cryptosystem implementations more efficiently. Barrett Reduction is a nothing but method of reducing a number modulo another number. Barrett reduction, when used to reduce a single number, is slower than a normal division algorithm. However, by pre computing some values, one can easily far exceed the speed of normal modular reductions. A straightforward style to perform large integer modular reductions is to use division [6]. A nice side effect is that it reuses the code of division, thus resulting in more compact code size. So Barrett reduction converts the reduction modulo an arbitrary integer to two multiplications and a few reductions modulo integers of the form $2n$.In Optimized-Elliptic Curve Cryptography, since almost all the modular operations are modulo the same prime number q, Barrett reduction can potentially speed up the computation. However, this requires the implementation of a separate reduction algorithm, which implies larger code size (i.e., greater ROM requirement) on sensor nodes. In addition, Barrett reduction also increases RAM use. Assume the target microcontroller has a w-bit word size. Given a finite field Fq, where q is a k words long prime number, Barrett reduction requires the pre-computation of $\mu = \texttt{floor}(b^{2k}/q)$, where $b = 2w$ *or* where b is the "base" of the integers used (e.g., $b = 2^8$ on a 8-bit processor). This number m has to be stored and used throughout all the modular reductions. Thus, to exchange for faster computation, Barrett reduction requires more ROM and RAM than the traditional division based modular reduction. A normal division algorithm or Classical Division Algorithm is as follows:-

The classical algorithm is a formalization of the ordinary l-k (l is size of argument, k is size of m) step pencil-and-paper method, each step of which is the division of a *(k+1)* digit number x by the k-digit divisor m. This yields the one-digit quotient q and the k digit remainder r. Each remainder r is less than m, so that it can be combined with the next digit of the dividend into the (k+1) digit number rb + (next digit of dividend) to be used as the new x in the next step.The pseudo code of the classical algorithm given $m_{k-1} \geq b/2$ follows:

if $(x > mb^{l-k})$ *then*
$x = x - mb^{l-k}$;
for $(i = l- 1; i > k - 1; i--)$ *do*
{
if $(x_i = = m_{k-1})$ *then*
$q = b - 1$;
else
$q = (x_i b + x_{i-1}) \text{ div } m_{k-1}$;
while $(q (m_{k-1} b + m_{k-2}) > x_i b^2 + x_{i-1} b + x_{i-2})$ *do*
$q = q- 1$;
$x = x- q\ m\ b^{i-k}$;
if $(x < 0)$ *then*
$x = x + m\ b^{i-k}$;
}
//Whereas for Barrett Reduction the generalized algorithm will be as follows
(for $\mu = floor\ (b^{2k} / m)$ *)*
$q = ((x \text{ div } b^{k-1})^1\ \mu \text{ div } b^{k+1}$;
$x = x \bmod b^{k+1} - (q\ m) \bmod b^{k+1}$;
if $(x < 0)$ *then*
$x = x + b^{k+1}$;
while $(x \geq m)$ *do*
$x = x- m$;

2) Montgomery Algorithm

The basic idea of Montgomery's theorem is to make x a multiple of R by adding multiples of m. Instead of computing all of t at once, one can compute one digit t_i at a time, add $t_i mb$ to x, and repeat. This change allows the computation of $m'_0 = m_0^{-1} \bmod b$ instead of m'.

The pseudo code of Montgomery's algorithm follows:

for $(i=0; i < k; i++)$ *do* {
$t_i = (x_i \bullet m'_0) \bmod b$;
$x = x + t_i m b^i$;
}
if $(x \geq m)$ *then* $x = x - m$;

If the time for pre- and post-computation and for m-residue transformation (for Montgomery only) is ignore, the Barrett algorithm is the fastest for argument smaller than 1024 bits, while Montgomery is the fastest for argument greater than 1024 bits. Each algorithm has its own features suitable for a specific field of application. No single algorithm provides a perfect solution to meet all demands; depending on the environment in which computation are to be performed, one algorithm may be preferable over another. For single modular reductions, the classical algorithm seems to be the best choice, as the pre- and post-calculations only involve a very fast and straightforward computation. For small arguments, classical and Barrett algorithms are almost equally fast, with slighter better performance for Barrett. For general modular exponentiation, the exponentiation based on Montgomery's algorithm has the best performance.

III. Optimizations for ECC Operations

Traditional Elliptic Curve cryptography is not optimized one for resource constraint sensor nodes because it adds extra overhead in terms of computation and memory cost. So there is need for optimization, I have considered that problem and optimized the existed elliptic curve cryptography Optimization can be achieved by simplifying the calculations

such as modular multiplication, or by reducing number of steps required for point addition and point doubling. Next section elaborates optimizations which are done to reduce complexity while marinating same security level as ECC.

A. Projective Coordinate Systems

An elliptic curve comprises of the infinity point Ô and the set of points in the affine coordinates (x, y) for x, y a finite field Fq that satisfies the defining equation. Alternatively, a point on an elliptic curve can be represented in a projective coordinate system in the form of (x, y, z). Point addition and point doubling are decisive operations in ECC, which are building blocks for scalar multiplications required by all ECC schemes. These operations in affine coordinate system necessitate modular inversion operations, which are much more expensive than other operations such as modular multiplications, which is not suitable for resource constrained devices. Using a projective coordinate system [7], modular inversions can be moved out with the compensation of a few modular multiplications and squares operation. Due to this, the execution times of point addition and point doubling based on projective coordinate system are faster than those based on affine coordinate system, respectively [7].Optimized-ECC uses two additional optimizations along with projective coordinate representation, which can further minimize both the execution time and the program size. The first one is a mixed point addition algorithm [7], which simply adds a point in projective coordinate and a second point in affine coordinate. This algorithm can be used in scalar multiplications to further reduce the number of modular multiplications and squaring operation, which leads to smaller and faster code. The other one is repeated Doubling [8] for scalar multiplication. If consecutive point doublings are to be performed, the repeated doubling algorithm may be applied to achieve faster performance instead of using doubling formula rapidly. In m consecutive doublings process, this algorithm trades m−1 field additions, m−1 divisions by two, and a multiplication for two field squaring (in comparison with repeated applications of the plain point doubling algorithm) [8]. Although reducing the execution time, the projective coordinate representation needs a larger code size (for implementing more complex formula) and more RAM (for storing additional required variables) than the affine coordinate system.

B. Curve Specific Optimization

A number of elliptic curves specified by NIST [9] and SECG [8] employ pseudo-Mersenne primes. A pseudo-Mersenne prime is of the form $p = 2n - c$, where $c \in 2n$. Reduction modulo a pseudo-Mersenne prime can be performed by a few modular multiplications and additions without any division operation. As a result, the time for modular reduction can be reduced significantly. Thus, using elliptic curves over a pseudo-Mersenne prime can achieve additional performance gain.

IV. Quantitative Overhead Analysis

A. Test Setup

Projects dealing with WSNs use TinyOS as their operating system. TinyOS [11] is an event-driven operating flexible, application-specific operating system for sensor networks. System projected for sensor network nodes that have very limited resources. TinyOS and programs for TinyOS are written in NesC [11]. The NesC programming language is designed specifically for TinyOS and it is based upon the concept of components that are connected or wired together to form a program. MICA2, MICAZ and TelosB from Crossbow platforms for performance evaluation. There are four nodes of both types available for the experiments. Because the MICA2s are easier to work with, due to their simpler connection to a PC, they are used instead of the MICA2DOTs.

The simulator that is used is TOSSIM [8], which stands for TinyOS Simulator. It is included with TinyOS together with a program called TinyViz that can be used to visualize the WSN network running in the simulator and also process debug data from some or all of the nodes. To use it, a TinyOS application needs to be compiled specifically for the simulator. The compiled executable can then be started with command line arguments telling the simulator how many nodes to simulate, what radio model and topology to use and its debugging and visualization settings. The simulations will either start running immediately or if specified, wait for TinyViz to connect to it. TinyViz can then show which node is sending messages to other nodes, who is broadcasting and which LEDs on the nodes are on and off. One drawback of the simulator is that all simulated nodes run the same application. This is a disadvantage when one of the nodes needs to act as a node that is performing an attack. To measure energy consumed by various cryptographic protocol PowerTOSSIM is used. This is extension of TOSSIM and provides an accurate per node estimate of power consumption. In PowerTOSSIM, specific hardware peripherals such as radio, EEPROM, LEDs and CPU are instrumented to obtain a trace of each peripheral's activity during the simulation run time. PowerTOSSIM energy model is based on the Mica2 sensor node platform [8].

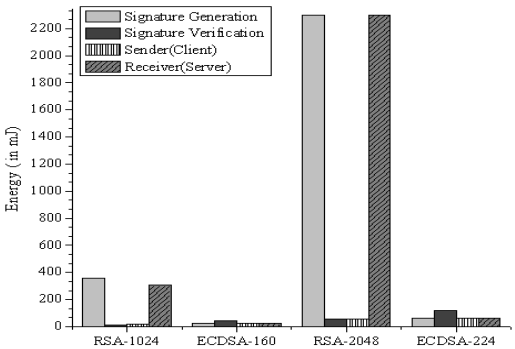

Fig. 1. Energy Consumed (in mille Joules) Digital Signature generation, Verification and in key exchange for RSA and Elliptic curve cryptography.

B. RSA Vs ECC: Performance Comparison

Verification of RSA is cheap but what makes it expensive is the signature generation for authentication. This is because RSA-based key exchange protocol relies on party A to encrypt a randomly generated secret key with party B's public key, and party B decrypting the key using its private key. Now for ECC the signature algorithm is based on Digital Signature Algorithm known as Elliptic Curve Digital Signature Algorithm [ECDSA]. For ECDSA the signature generation and verification both are cheap. The transition

from RSA-1024 to RSA-2048 the energy cost of signing increases by a factor of more than seven, whereas for ECDSA-224 signing is less than three times as expensive as ECDSA-160 signing. With ECC, both parties perform a single ECDH operation to derive the secret key.

Performance measurement for both algorithms can be compared as follows. Hence with the facts, figures and numbers we can clearly say that ECC is the better choice than the most popularly used encryption algorithm for WSN.

C. Optimized-ECC Performance Evaluation

We have implemented optimized-ECC for TelosB and Mica2 platform, it can also be extended for other sensor platforms as well, and for evaluating required time to generate signature and for signature verification we have written a java code using jdk 1.5 and javacomm package. Firstly, I've used two TelosB motes for testing my algorithm, one mote is Alice, and another is Bob. Let us suppose Alice's (mote 1) public key is pre deployed in Bob (mote 2). Alice broadcasts packets with her signature. Bob receive packets and verifies all packets from Alice. Red LED for Alice indicates the signature generation whereas red LED for Bob means, Bob is verifying the signature. If computed signature is correct, Bob will start toggling the green LED, else Bob will turn on all three LEDs. Execute a serial forwarder on your computer, as with the command below, adjusting as needed for your particular configuration.

java net.tinyos.sf.SerialForwarder -comm. serial@COM3:telos &

With the Serial Forwarder backgrounder (or running in its own terminal), also the following has to be executed: java output (see Figure 2)

```
$ java output
Private key:
d: c1c9706b28ba8cd8dee77ffca775433531150805
[ time of ECC.init() is 0.09244466 sec ]
Public key:
x: d5bfa04f395fb40547a844d977b17b75d7573450
y: e28bff59442ceb2509bc02e2f13cc80a1e9a3b76
[ time of public key generation is 0.1465343 sec ]
[ time of ECDSA.init() is 0.17921549 sec ]
content and signature
msg: e2d6b379edcc82163379edc5953c62d6b3700a0b001f
284f81143771fdecc7913d65d5b575fce2d6be6ece83192457b87aebc98
d
signature
r: a52ecbb0450a23d877e1f2266019923e219636c6
s: 92836aeb06c568d81a400ef50b102d72ead142c9
[ time of signature generation is 0.15532987 sec ]
[ time of signature verification is 0.18770833 sec ] (pass)
Average timing result
ECC.init(): 0.09247005
ECDSA.init(): 0.21443835
public key gen: 0.14625478
sign: 0.1851174
verify: 0.22907898
```

Fig. 2. In text form result obtained is as follows for first round.

This will simply report to standard output any messages delivered to the mote attached to PC. One mote which generates the signature and one Telob is directly connected to PC which verifies the signature; basically following simulation demonstrate the running of ECDSA .The same experiment is performed for MICAz platform too.

V. Result Discussion

Using POWER-TOSSIM simulator, Figure 3. Shows the execution time required ECDSA initialization, Signature generation and signature verification, ECIES initialization, Encryption, decryption ECDH initialization, key establishment and Figure 4. Shows Energy consumption (in mJ) for digital signature scheme (ECDSA- Elliptic Curve Digital Signature Algorithm), public key encryption scheme (ECIES- Elliptic Curve Integrated Encryption Scheme) and a key exchange protocol scheme (ECDH - Elliptic Curve Diffie-Hellman). The analysis shows that the TelosB performance is higher than Mica2.

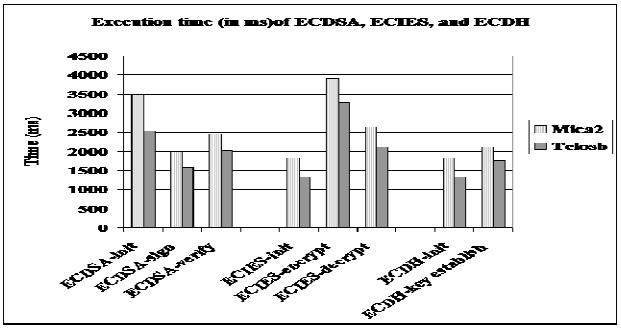

Fig. 3. Eecution Time (ms) for ECDSA, ECIES and ECDH operation on Mica2 and telosB motes.

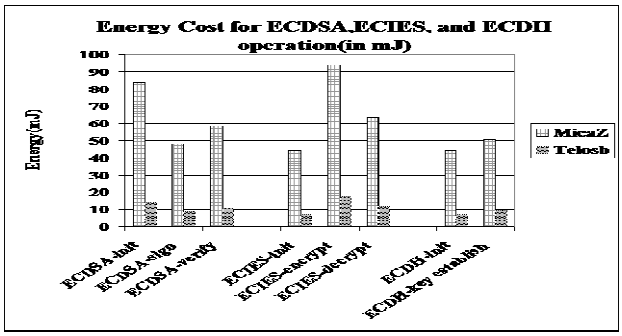

Fig. 4. Energy Cost (mJ) for ECDSA, ECIES and ECDH operation on Mica2 and telosB motes.

VI. Conclusion and Future Scope

We work brought out a light security algorithm i.e. Optimized ECC which is enhancement of traditional Elliptic Curve Cryptography. The idea for Optimized ECC has been taken from Mathematics where the unique property of Elliptic curves have been used and optimization is provided using easy computation , mathematically it is proved that Asymmetric Cryptography can be implemented in these minuscule sensor devices. As observed it drains the battery power but there has to be a tradeoff between the energy utilization and security level. Paper also described and compares the energy consumption as well as running time for TinySec and Optimized ECC. Most of the modern sensors today operate on renewable energy source hence public key cryptography can be implemented in these resource constraint embedded sensor devices. There is scope of resolve DoS attack and more optimization O-ECC toward light & ad hoc network. Further work remains in minimization encryption and decryption operation implementation. Optimized-ECC performance evaluation can also be extended for other sensor platforms.

References

[1] Anoop MS *Elliptic Curve Cryptography–An Implementation Tutorial*:www.tataelxsi.com/whitepapers/ECC_Tut_v1_0.pdf?pdf_id=public_-key_TEL.pdf

[2] Don B. John *Elliptic curve DSA (ECSDA): an enhanced DSA :* 7th conference on USENIX Security Symposium - Volume 7 USENIX Association Berkeley, CA, USA ©1998
[3] S Wang : *Efficient implementation of elliptic curve Diffie-Hellman (ECDH) key*: IEEE COMMUNICATIONS LETTERS, VOL. 12, NO. 2, FEBRUARY 149.2008.
[4] An Liu; Peng Ning :*TinyECC: A Configurable Library for Elliptic Curve Cryptography in Wireless Sensor Networks* : Information Processing in Sensor Networks, IPSN '08. 2008.
[5] Menezes, *Elliptic Curve Public Key Cryptosystems*. Kluwer Academic Publishers, 1993.
[6] *Faster Interleaved Modular Multiplication Based on* Barrett: www.cosic.esat.kuleuven.be/publications/article-1191.pdf
[7] *Performance analysis of Point multiplication methods for Elliptic*: www.rimtengg.com/iscet/proceedings/pdfs/misc/172.pdf
[8] Levis, N. Lee, M. Welsh and D. Culler, "*TOSSIM: Accurate and Scalable Simulation of Entire TinyOS Applications*," Proceedings of the 1st International Conference on Embedded Networked Sensor Systems, pp 126-137. 2003.
[9] Certicom Research. Standards for efficient cryptography SEC 2: Recommended *elliptic curve domain parameters*.:www.secg.org/collateral/sec2_final.pdf, September 2000.
[10] A.J.Menezes, P. C. van Oorschot, and S.A.Vanstone. *Handbook of* Applied *Cryptography*. CRC Press, 1996.
[11] http://www.tinyos.net/tinyos-1.x/doc/tutorial
[12] Certicom. Information on the Certicom ECC challenge, http://www.certicom.com/research/ecc hallenge.html

Arpit works a s Lecturer in IERT, Priyag, Allahabad. He has four and half years experience in Academic area and half year experience in industry as a research associate. Major area of interest is modeling and analyzing systems formally. Automata with Infinite words and Probalistic automata are also areas in with he is working. He is also publisher of three major papers which are published in ACM, IEEE and Spinger.

He has guided my projects of B.Tech students during his career.

His passion is research. He has earned his B.Tech degree in Computer Science Engg from N.I.T Hamirpur(H.P).

Ashwini Kumar work as Lecturer in IERT, Prajyag, Allahabad, India. He has Four year experience in Academic Profession. Major areas of interest are Network, Security and software Formal Method. Currently working on Formal verification of communication network and related its security. He earns the B. Tech CSE Degree from HBTI Kanpur (India) in 2006 and currently engages with various research and projects in IERT.