

CHAPTER 1

1.1 INTRODUCTION

Cryptography has been in use for centuries now, and the earliest ciphers were either used transposition or substitution, and messages were encoded and decoded by hand. However, these schemes satisfied only the basic requirement of confidentiality. In more recent times, with the invention of processing machines, more robust algorithms were required, as the simple ciphers were easy to decode using these machines, and moreover they did not have any of the afore mentioned properties. Secure data communication became a necessity in the 20th century and a lot of research was done in this field by government agencies, during and following the world-wars. The most famous machine of this time, Enigma was an electromechanical device which was used by the German Army.

The first secret key-based cryptographic algorithms worked on the symmetric algorithms. They assumed that both communicating parties shared some secret information, which was unique to them, much like the older One Time Pads. Using this secret information, also called a key, the sender encrypted the data, and the recipient was able to decrypt.

Public key cryptography or conventional key cryptography uses two different keys one for encryption and other for decryption. The main problem of conventional public key cryptography systems is that the key size has to be sufficient large in order to meet the high-level security requirement. This results in lower speed and consumption of more bandwidth. Hence Elliptic Curve Cryptography system comes to existence.

Elliptic curve cryptography (ECC) is an approach to public-key cryptography based on the algebraic structure of elliptic curves over finite fields. ECC requires smaller keys compared to non-ECC cryptography to provide equivalent security.

The technique was first proposed individually by Neal Koblitz and Victor Miller in 1985. Elliptic curves are applicable for encryption, digital signatures, pseudo random generators and other tasks. They are also used in several Integer factorization algorithms that have applications in cryptography.

There are several parameters and algorithm choices which should be considered before implementing ECC system. Several curve domain parameters (field representation, curve type), algorithm for field arithmetic, elliptic curve arithmetic, and protocol arithmetic can be influenced by security factors, platform, constraints, and communications environment.

1.2 LITERATURE SURVEY

The ECC technique was first proposed individually by Neal Koblitz and Victor Miller in 1985. In the late 1990's, ECC was standardized by a number of organizations and it started receiving commercial acceptance. Nowadays, it is mainly used in the resource constrained environments, such as ad-hoc wireless networks and mobile networks[2].

Menezes and Jurisic, in their paper [JM97], said that to achieve reasonable security, a 1024-bit modulus would have to be used in a RSA system, while 160-bit modulus should be sufficient for ECC.

The security of ECC schemes is based on the resolution of an underlying mathematical problem called the Elliptic Curve Discrete Logarithm Problem (ECDLP), hard to solve so far. In ECC, public and private keys are the tool that allows locking and unlocking the cryptographic system. A pair of keys is associated for an elliptic curve E with specific domain parameters D . For cryptographic purposes, elliptic curves are used over finite fields. To generate a key pair, one selects a random integer d from the interval $[1; n - 1]$, which serves as the private key, and computes $Q = dP$ (point multiplication)

which is used as the corresponding public key. The determination of d given E , P and Q is the ECDLP[3].

Encryption and decryption algorithms are based on point multiplications. This operation is considered as the cornerstone of any ECC scheme. Point multiplication relies on the underlying operations of point addition and doubling; these operations form the Elliptic Curve arithmetic and are based on finite-fields arithmetic (addition, multiplication, inversion, etc.). The diagram in the Figure 1.1 illustrates these dependencies.

The main problems with widespread adoption of ECC is simply that it uses a complex and sophisticated form of mathematics to set up secure curves, and that it is not readily understood by information technology professionals responsible for implementing security systems based on public-key cryptography, let alone information systems managers. In contrast, the RSA scheme is not that difficult to understand given normal high-school level mathematics. In this regard it is highly unlikely that an enterprise will dedicate scarce programming resources to creation of appropriate ECC packages. Given the complexity of the mathematics involved to define a curve that ensures system security, the number of people likely to investigate and attack the cipher system is severely reduced[4].

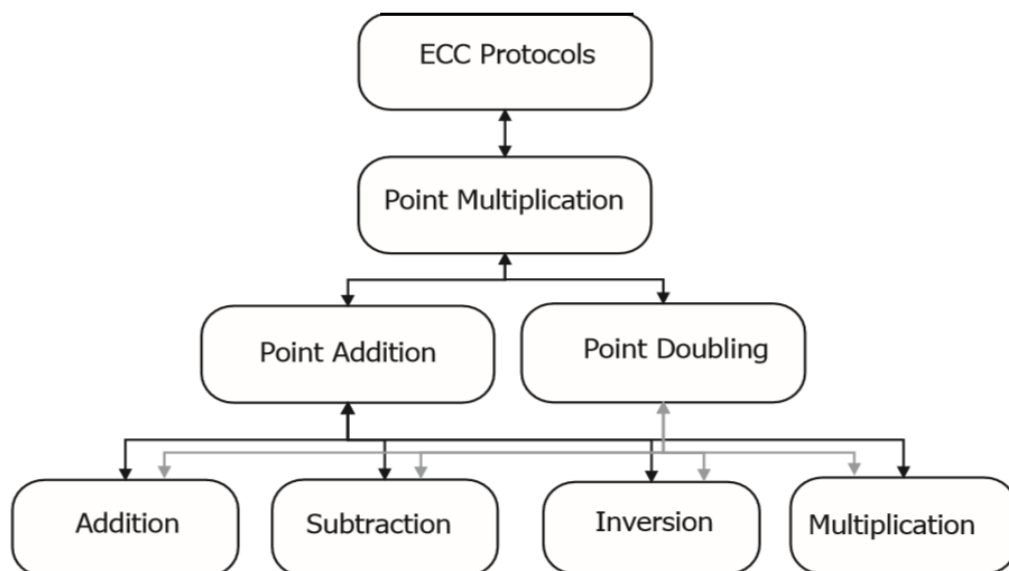


Figure 1.1 Layers of Elliptic Curve Cryptography

The key of an elliptic curve based crypto system takes significantly less memory. The ratio increases rapidly with the increase of security levels. For instance, RSA crypto system with the key length of 1024 bits, is equivalent to an elliptic curve crypto system with the key length of 163 bits.

Key Size (bits)		Generation time (seconds)	
ECC	RSA	ECC	RSA
163	1024	0.08	0.16
233	2240	0.18	7.47
283	3072	0.27	9.89
409	7680	0.64	133.90
571	15360	1.44	679.06

Table 1.1 Comparison of Key Generation of ECC and RSA

The cryptographic operations such as key and digital signature generation are carried out significantly faster for smaller size of keys. For instance, an elliptic curve crypto system with the key length of 233 bits corresponds to RSA crypto system with the key length of 2240 bits. In the first case the key is generated approximately 40 times faster. Due to the smaller key sizes, algorithms of an elliptic curve based crypto systems can be executed on very limited resources[5].

More than a decade after the first ECC standardization the instantiation of public key cryptography is gaining in popularity. Although ECC is still far from the dominant choice for cryptography, the elliptic curve cryptographic landscape shows considerable deployment in 2013. Of the 12 million scanned hosts which support SSH, we found that 10.3% supported ECDSA for authentication and 13.8% supported a form of ECDH for key exchange. 30.2 million TLS servers and found that 7.2% support a form of ECDH. In the Austrian citizen card database, 58% of the 829000 use ECDSA to create digital signatures. All asymmetric cryptography in Bitcoin is based on ECC[6].

1.3 OBJECTIVES

The main objectives are:

- To provide same security as other public key cryptography methods by using smaller size keys.
- To reduce power consumption.
- To increase speed of computation and to utilize bandwidth efficiently.

CHAPTER 2

REQUIREMENT SPECIFICATION

Software Requirement

- Working O.S.: windows any linux platforms
- Language used: any language can be used c,c++,c#,java etc
- Tool :Eclipse
- Optimized 256-bit ECC implementation on 3GHz 64-bit CPU requires about 2 ms per point multiplication.
- Less powerful microprocessors (e.g, on SmartCards or cell phones) even take significantly longer (>10 ms).

Hardware Requirement

- Minimum of 2GB RAM.
- I3 processor.
- High-performance implementations with 256-bit special primes can compute a point multiplication in a few hundred microseconds on reconfigurable hardware.
- Dedicated chips for ECC can compute a point multiplication even in a few ten microseconds.

CHAPTER 3

DESIGN

An elliptic curve is defined by an equation in two variables with coefficients. For cryptography, the variables and coefficients are restricted to elements in a finite field, which results in the definition of a finite abelian group.

An abelian group G , is denoted by $\{G, *\}$, is a set of elements with a binary operation, denoted by $*$, that associates to each ordered pair (a, b) of elements in G an element $(a \# b)$ in G , such that the following axioms are obeyed:

(A1) Closure: If a and b belong to G , then $a * b$ is also in G .

(A2) Associative: $a * (b * c) = (a * b) * c$ for all a, b, c in G .

(A3) Identity element: There is an element e in G such that $a * e = e * a = a$ for all a in G .

(A4) Inverse element: For each a in G there is an element a' in G such that $a * a' = a' * a = e$.

(A5) Commutative: $a * b = b * a$ for all a, b in G .

A number of public-key ciphers are based on the use of an abelian group. For example, Diffie-Hellman key exchange involves multiplying pairs of nonzero integers modulo a prime number q . Keys are generated by exponentiation over the group, with exponentiation defined as repeated multiplication. For example, $a^k \bmod q = (a * a * \dots * a) \bmod q$. To attack Diffie-Hellman, the attacker must determine k given a and a^k ; this is the discrete logarithm problem.

For elliptic curve cryptography, an operation over elliptic curves, called addition, is used. Multiplication is defined by repeated addition. For example, $a * k = (a + a + \dots + a)$ k times where the addition is performed over an elliptic curve. Cryptanalysis involves determining k given a and $(a * k)$.

Elliptic curves are not ellipses. They are so named because they are described by cubic equations, similar to those used for calculating the circumference of an ellipse. In general, cubic equations for elliptic curves take the following form, known as a Weierstrass equation:

$$y^2 + axy + by = x^3 + cx^2 + dx + e$$

where a, b, c, d, e are real numbers and x and y take on values in the real numbers.⁴ For our purpose, it is sufficient to limit ourselves to equations of the form

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

Such equations are said to be cubic, or of degree 3, because the highest exponent they contain is a 3. Also included in the definition of an elliptic curve is a single element denoted O and called the point at infinity or the zero point. Figure 3.1 shows an example elliptic curve for $y^2 = x^3 - 3x + 3$.

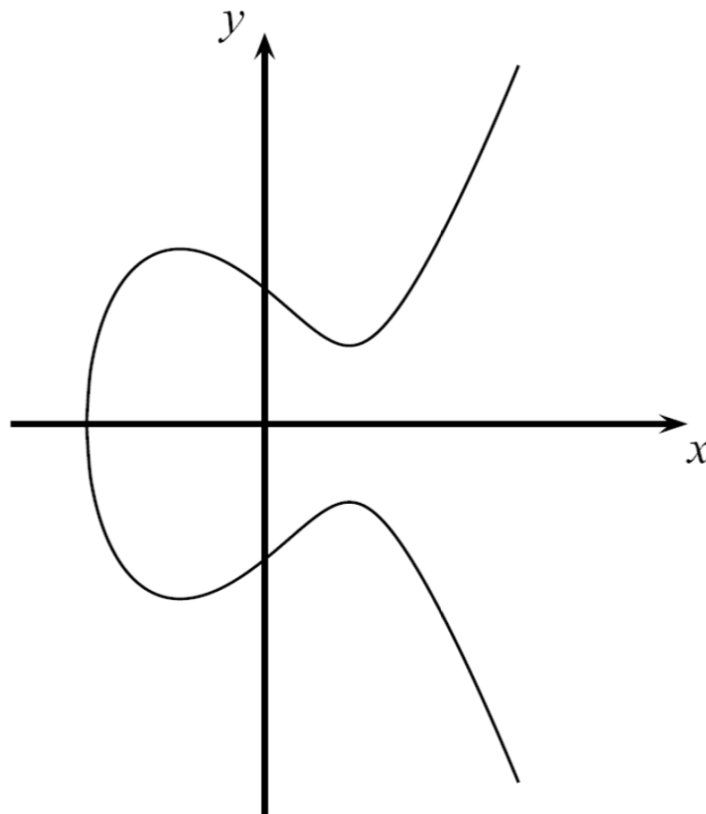


Fig 3.1 Elliptic Curve

Some special considerations are required to convert elliptic curves into a group of points. In any group, a special element is required to allow for the identity operation,

$$\text{i.e., given } P \in E: P + \theta = P = \theta + P$$

This identity point (which is not on the curve) is additionally added to the group definition, this (infinite) identity point is denoted by θ . Figure 3.2 shows an symmetric elliptic curve with a point at infinity denoted by θ .

Elliptic Curve are symmetric along the x-axis. Up to two solutions y and $-y$ exist for each quadratic residue x of the elliptic curve. For each point $P = (x, y)$, the inverse or negative point is defined as $-P = (x, -y)$

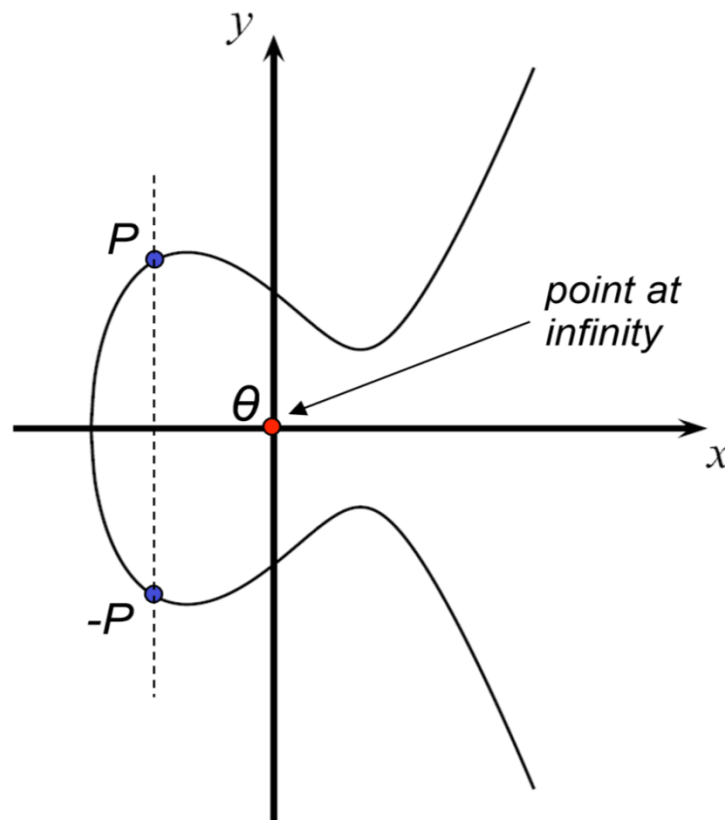


Figure 3.2 Symmetric Elliptic Curve with a Point at Infinity

3.1 Computations on Elliptic Curve

Generating a group of points on elliptic curves based on point addition operation

$$P+Q = R, \text{ i.e., } (x_P, y_P) + (x_Q, y_Q) = (x_R, y_R)$$

Geometric Interpretation of point addition operation

- Draw straight line through P and Q ; if $P=Q$ use tangent line instead.

- Mirror third intersection point of drawn line with the elliptic curve along the x-axis.

Figure 3.3 shows the graphical representation of point addition operation.

Geometric Interpretation of point addition operation

- To the point P on elliptic curve, draw the tangent line to the elliptic curve at P.
- The line intersects the elliptic curve at the point -R. The reflection of the point -R with respect to x-axis gives the point R, which is the results of doubling of point P. i.e., $R=2P$.

Figure 3.4 shows the graphical representation of point doubling operation.

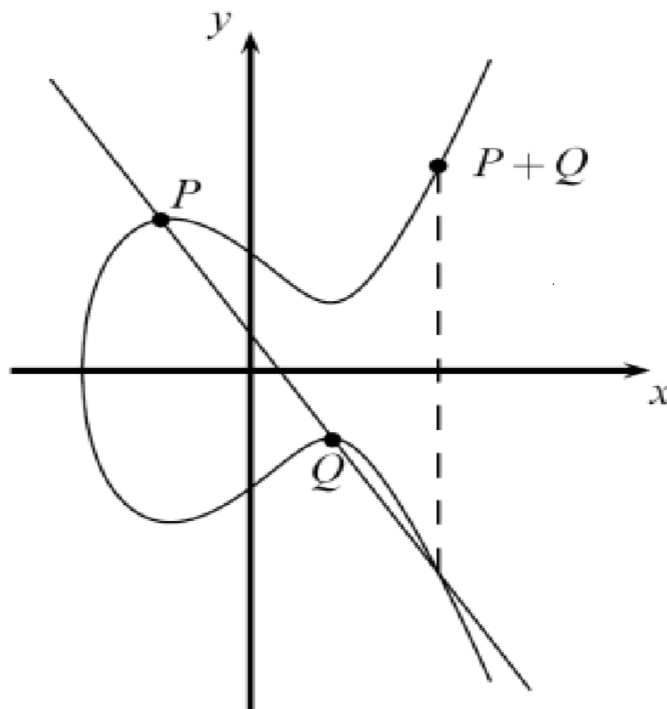


Figure 3.3 Point addition in Elliptic Curve.

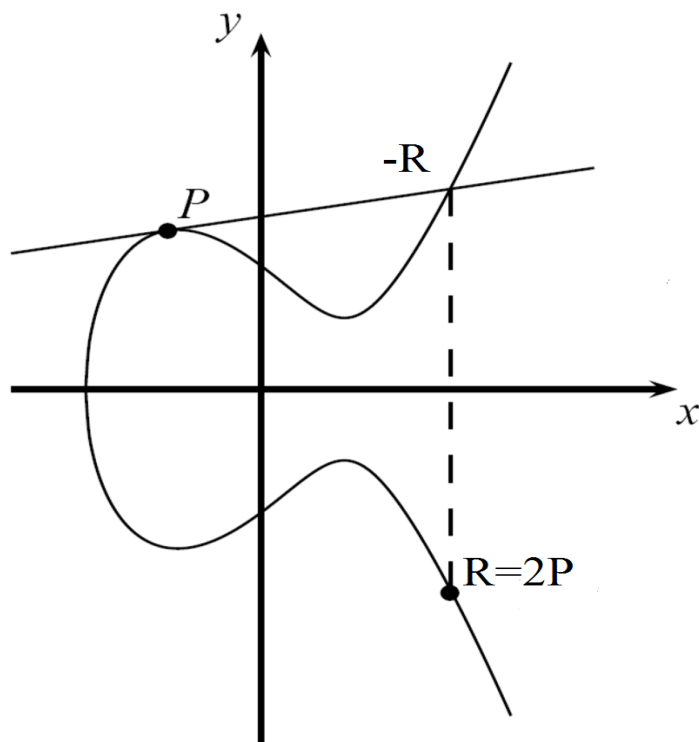


Figure 3.4 Point Doubling in Elliptic Curve.

CHAPTER 4

IMPLEMENTATION

The point multiplication (aka scalar multiplication) is by far the most time-consuming operation of ECC. In point multiplication, a point P is multiplied by an integer k resulting in another point Q on the curve. The multiplication can be performed through a combination of point additions and point doublings.

Double and Add Algorithm

Require $k=(k_{m-1},k_{m-2},\dots,k_0)_2, k_{m-1}=1$

Compute $Q=kP$

$Q=P$

For $i=m-2$ to 0

$Q=2Q$

if $k_i=1$ then

$Q=Q+P$

End if

End for

Return Q

Example: $26P = (11010_2)P = (d_4d_3d_2d_1d_0)_2 P$.

Step		
#0	$P = 1_2P$	initial setting
#1a	$P+P = 2P = 10_2P$	DOUBLE (bit $d_3=1$)
#1b	$2P+P = 3P = 10^2_2P + 1_2P = 11_2P$	ADD (bit $d_3=1$)
#2a	$3P+3P = 6P = 2(11_2P) = 110_2P$	DOUBLE (bit d_2)
#2b		no ADD ($d_2 = 0$)
#3a	$6P+6P = 12P = 2(110_2P) = 1100_2P$	DOUBLE (bit d_1)
#3b	$12P+P = 13P = 1100_2P + 1_2P = 1101_2P$	ADD (bit $d_1=1$)
#4a	$13P+13P = 26P = 2(1101_2P) = 11010_2P$	DOUBLE (bit d_0)
#4b		no ADD ($d_0 = 0$)

Figure 4.1 Example

In Figure 4.1 some examples are given by tracing the double and add algorithm.

Elliptic Curve Diffie-Hellman (ECDH) Key Exchange

Given a prime p , a suitable elliptic curve E and a point $P=(x_P, y_P)$. ECDH operates by providing the two parties sharing a secret key with a public key, which in this case is a point P on elliptic curve E .

The Elliptic Curve Diffie-Hellman Key Exchange is defined by the following protocol:

- Alice performs scalar multiplication using this point P and a scalar multiple a , which is secret key of Alice. $(a.P)$ now becomes publickey of Alice which she can share with the other party.
- On the other end, Bob performs scalar multiplication using point P and a scalar multiple of his choice i.e. b , which is secret key of Bob. $(b.P)$ becomes publickey of Bob which he shares with Alice.
- Alice performs scalar multiplication of public key of Alice $(b.P)$ with her secret key a to get $a.b.P$. Bob also does the same with his secret key b and public key of Alice $(a.P)$ to get the same $a.b.P$. This entity i.e. $a.b.P$ is same for both the parties and is their shared key. Figure 4.2 shows this overall key exchange mechanism in pictorial form.

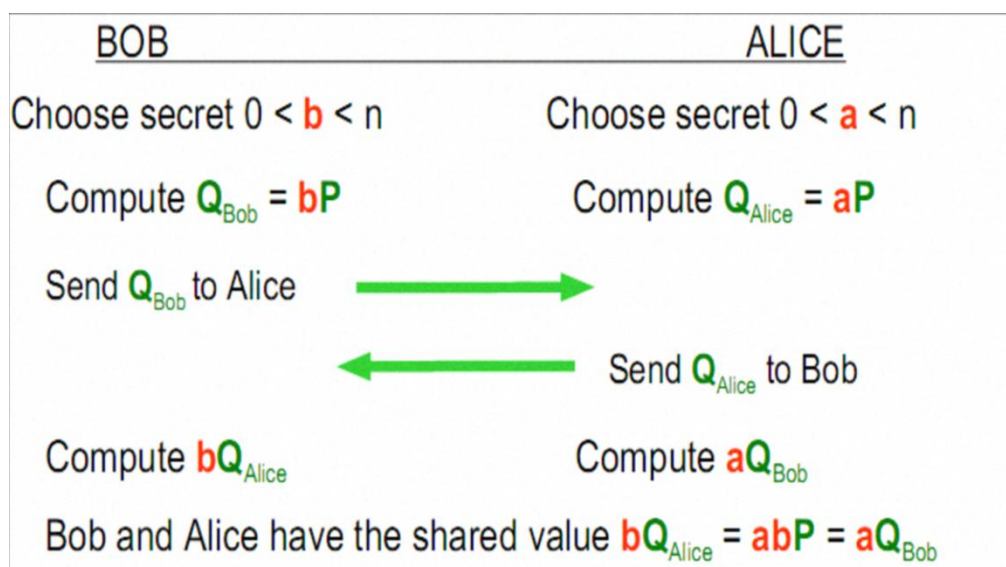


Figure 4.2 Elliptic Curve Deffie Hellman(ECDH) key exchange.

CHAPTER 5

RESULTS AND ANALYSIS

To generate a public key $Q(x,y)$, select a point $P(x_1,y_1)$ on the elliptic curve which acts as private key and multiply it with some scalar say k . Figure 5.1 shows the result of key generation.

```

Java - Ecc_Demo/src/ECCKeyGeneration.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
ECCSignature.java ECCKeyGeneration.java ECCKeyAgreement.java
// The following code is from http://www.academicpub.org/PaperInfo.aspx?PaperID=14496 .
*import java.security.*;

public class ECCKeyGeneration {
    public static void main(String[] args) throws Exception {
        KeyPairGenerator kpg;
        kpg = KeyPairGenerator.getInstance("EC", "SunEC");
        ECGenParameterSpec ecsp;
        ecsp = new ECGenParameterSpec("secp192r1");
        kpg.initialize(ecsp);

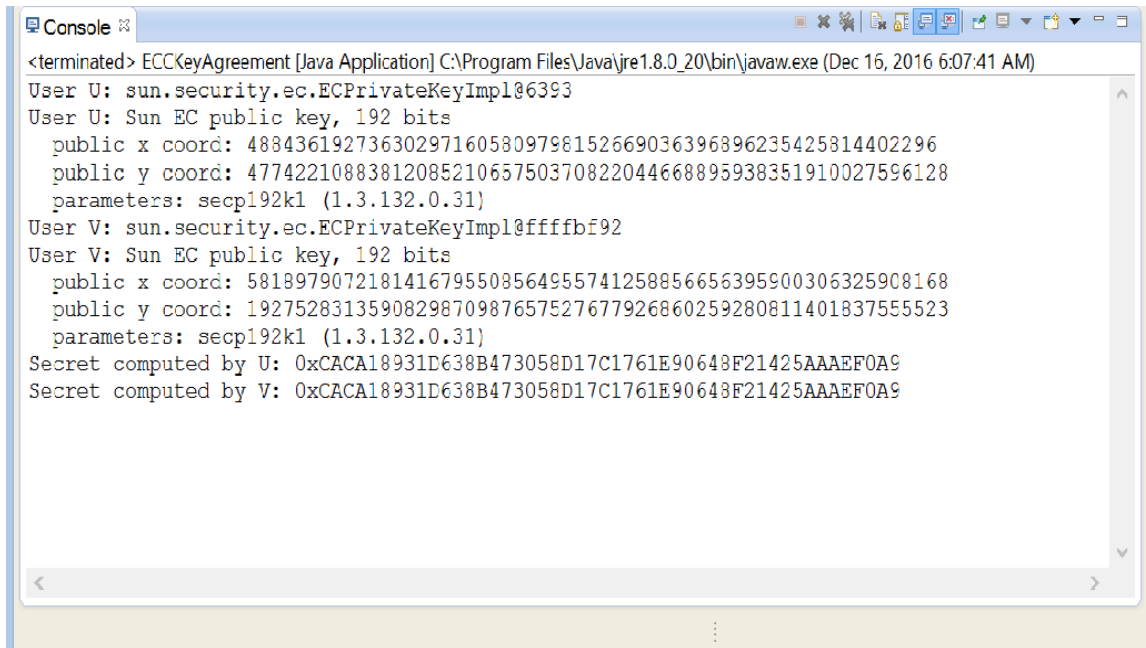
        KeyPair kp = kpg.genKeyPair();
        PrivateKey privKey = kp.getPrivate();
        PublicKey pubKey = kp.getPublic();

        System.out.println(privKey.toString());
        System.out.println(pubKey.toString());
    }
}

Console
<terminated> ECCKeyGeneration [Java Application] C:\Program Files\Java\jre1.8.0_20\bin\javaw.exe (Dec 15, 2016 10:33:54 PM)
sun.security.ec.ECPrivateKeyImpl@fffff23e
Sun EC public key, 192 bits
public x coord: 2985657455595699424154793597525283841687894275342708277947
public y coord: 5199138497160169906948908828843128722352553078988567205494
parameters: secp192r1 [NIST P-192, X9.62 prime192v1] (1.2.840.10045.3.1.1)
    
```

Figure 5.1 Generation of public key.

The generated key can be exchanged using Elliptic curve Diffie hellman key exchange algorithm. The Figure 5.2 shows the snapshot result of key exchange with secret key calculation. the secret key can be calculated once the sender and reciever exchanges there public key, which can be used for further transactions.



```
<terminated> ECCKeyAgreement [Java Application] C:\Program Files\Java\jre1.8.0_20\bin\javaw.exe (Dec 16, 2016 6:07:41 AM)
User U: sun.security.ec.ECPrivateKeyImpl@6393
User U: Sun EC public key, 192 bits
  public x coord: 488436192736302971605809798152669036396896235425814402296
  public y coord: 4774221088381208521065750370822044668895938351910027596128
  parameters: secp192k1 (1.3.132.0.31)
User V: sun.security.ec.ECPrivateKeyImpl@ffffbf92
User V: Sun EC public key, 192 bits
  public x coord: 5818979072181416795508564955741258856656395900306325908168
  public y coord: 1927528313590829870987657527677926860259280811401837555523
  parameters: secp192k1 (1.3.132.0.31)
Secret computed by U: 0xCACA18931D638B473058D17C1761E90648F21425AAAEF0A9
Secret computed by V: 0xCACA18931D638B473058D17C1761E90648F21425AAAEF0A9
```

Figure 5.2 Secret key generation

CHAPTER 6

CONCLUSION

Comparison of the performance of ECC with RSA is done in terms of key sizes for the same level of security, data sizes, encrypted message sizes, and computational power. RSA takes sub exponential time and ECC takes full exponential time. The security of Elliptic Curve Cryptosystem depends on how difficult it is to determine x given xP and P . This is referred to as the elliptic curve logarithm problem.

This implementation takes only fraction of a second. Along with time consumption ECC provides power consumption too. This makes it an ideal choice for portable, mobile and low power applications. It can be a very secure and useful replacement of already being used cryptosystems for key exchange, key agreement and mutual authentication due to its complex algorithm which is difficult for cryptanalysis. Due to the smaller key sizes, algorithms of an elliptic curve based crypto systems can be executed on very limited resources.

As the model suggests, the ECC API is used in the security layer to automatically encrypt/decrypt all data that flows to or from the application layer. This model allows the application to be oblivious to encryption issues by designating that responsibility to the security layer. The security layer in turn will depend on the API to carry out its task. A major advantage of this model is that existing applications do not have to be rewritten to utilize the ECC API, instead they can be executed as they are and still benefit from the new encryption scheme.

REFERENCES

- [1].William Stallings “Cryptography and Network Security” ,principles and practice Sixth edition.
- [2]. Muhammad Yasir Malik " Efficient Implementation of Elliptic Curve Cryptography Using Low-power Digital Signal Processor" Feb. 7-10, 2010 ICACT 2010 ISBN 978-89-5519-146-2
- [3]. Nejmeddine ALIMI, Younes LAHBIB, Mohsen MACHHOUT & Rached TOURKI "On Elliptic Curve Cryptography implementations and evaluation" 978-1-4673-8526-8/16 2016 IEEE.
- [4].William J Caelli, Professor Edward P Dawson and Scott A Rea "PKI, Elliptic Curve Cryptography, and Digital Signatures" Computers & Security, 18 (1999) 47-66.
- [5]. Kristi Magons " Applications and Benefits of Elliptic Curve Cryptography" .
- [6]. Joppe W. Bos¹, J. Alex Halderman, Nadia Heninger, Jonathan Moore, Michael Naehrig, and Eric Wustrow " Elliptic Curve Cryptography in Practice"