ISSN:0975-9646

An Elliptic Curve Cryptography approach for Digital Signature in PDA devices.

Mahendra Singh yadav, Mahendra Kumar Rai

SRIT jabalpur , India

1. ABSTRACT-The problem undertaken for this paper is "An Elliptic Curve Cryptography approach for Digital Signature in PDA devices". Digital transaction have become common place and in some cases inextricably linked to modern life. This technological dependency requires that information be unaltered and confidential. So in this paper, problem is to search a good secure technique, which ensures the confidentiality and privacy of message. Cryptography is one efficient way to ensure that if sent message fall into wrong hands, they cannot read it. It is the art of secret writing. Digital signature allows the verification of the 'origin' of messages. We use the concept of RSA (by Rivest, Shamir and Adleman) and Elliptic curve Algorithm to implement Digital Signature. Our problem is to find equation of polynomial such that it is too complex to design its elliptic curve. An elliptic-curve group for cryptography comes from the multiples of a generating point 'G' a two dimensional point on an elliptic curve over a finite field. In practice, the finite fields used are either integers modulo large primes, or a similar construction using 0/1 polynomials.

2. INTRODUCTION

Elliptic Curve Cryptography (ECC) is a public key cryptography. In public key cryptography each user or the device taking part in the communication generally have a pair of keys, a public key and a private key, and a set of operations associated with the keys to do the cryptographic operations. Only the particular user knows the private key whereas the public key is distributed to all users taking part in the communication. Some public key algorithm may require a set of predefined constants to be known by all the devices taking part in the communication. 'Domain parameters' in ECC is an example of such constants. Public key cryptography, unlike private key cryptography, does not require any shared secret between the communicating parties but it is much slower than the private key cryptography.

The mathematical operations of ECC is defined over the elliptic curve $y^2 = x^3 + ax + b$, where $4a^3 + 27b^2 \neq 0$. Each value of the 'a' and 'b' gives a different elliptic curve. All points (x, y) which satisfies the above equation plus a point at infinity lies on the elliptic curve. The public key is a point in the curve and the private key is a random number. The public key is obtained by multiplying the private key with the generator point G in the curve. The generator point G, the curve parameters 'a' and 'b', together with few more constants constitutes the domain parameter of ECC. One main advantage of ECC is its small key size. A 160-bit key in ECC is considered to be as secured as 1024-bit key in RSA.

There are three families of public-key cryptography in common use today. The most widely used systems are those based on integer factorization. In particular, the RSA cryptographic system is perhaps the most popular public-key algorithm. It is used in most web browsers (for SSL), email packages (for S/MIME) as well as within the Entrust family of products. Systems based on the discrete logarithm problem are also very popular as they can provide support for both digital signatures (with DSA) and key agreement (with the Diffie-Hellman algorithm). Traditionally Entrust has supported both of these families of cryptographic algorithms. This family is based on arithmetic using elliptic curves. **Elliptic curve cryptography (ECC)** is a relatively new family of public-key algorithms that can provide shorter key lengths and, depending upon the environment and application in which it is used, improved performance over systems based on integer factorization and discrete logarithms. This paper will describe about the Elliptic Curve Cryptography, discuss its security and performance advantages and describe Entrusts support of this important type of Cryptography.

3. ELLIPTIC CURVES ARITHMETIC

3.1 FIELD ARITHMETIC

ECC uses modular arithmetic or polynomial arithmetic for its operations depending on the field chosen. The arithmetic involves big numbers in the range of 100s of bits. This section gives a brief overview for these two finite field operations.

3.2 MODULAR ARITHMETIC

Modular arithmetic over a number p involves arithmetic between numbers 0 and p – 1. If the number happens to be out of this range in any of the operation the result is wrapped around in to the range 0 and p – **ADDITION**

Let p = 23, a = 15, b = 20

a + b (mod p) = 15 + 20 (mod 23) = 35 mod 23 = 12

Since the result of a + b = 35 which is out of the range [0 22], The result is wrapped around in to the range [0 22] by subtracting 35 with 23 till the result is in range [0 22]. a mod b is thus explained as remainder of division a/b.

SUBTRACTION

Let p = 23, a = 15, b = 20

a - b (mod p) = 15 - 20 (mod 23) = -5 mod 23 = 18

Since the result of a - b = -5 which is negative and out of the

range [0 22], The result is wrapped around in to the range [0 22] by adding -5 with 23 till the result is in range [0 22].

MULTIPLICATION

Let p = 23, a = 15, b = 20

a * b (mod p) = 15 * 20 (mod 23) = 300 mod 23 = 1

Since the result of a * b = 300 which is out of the range [0 22], The result is wrapped around in to the range [0 22] by subtracting 300 with 23 till the result is in range [0 22].

DIVISION

The division a/b (mod p) is defined as a * b^{-1} (mod p). b^{-1} is the multiplicative inverse of b over p.

MULTIPLICATIVE INVERSE

Multiplicative inverse of number b with respect to mod p is defined as a number b^{-1} such that $b*b^{-1}$ (mod p) = 1. Multiplicative inverse exists only if b and n are relatively prime. The algorithm such as extended Euclidean algorithm can be used to find the multiplicative inverse of a number efficiently. Finding multiplicative inverse is a costly operation.

FINDING X MOD Y

x mod y is the remainder of the division x/y. Finding x mod y by repeatedly subtracting y with x till the result is in range [0 y^{-1}] is a costly operation. Methods such as Barrett Reduction can be used to find modulus of a number in efficient manner.

3.3 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 (am-1 ... a1 a0) can be expressed as polynomial am-1xm-1 + am-2xm-2 + ... + a2x2 + a1x + a0 where ai = 0

1. For e.g., a 4 bit number 1101 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 a reduction polynomial.

In binary polynomial 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 operation on the coefficient.

All the operations below are defined in field F24 are on irreducible polynomial $f(x) = x^4 + x + 1$. Since m = 4 the operation involves polynomial of degree 3 or lesser.

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

B = 0110

A + B = 1011 which is an XOR operation between A and B. This is true in all cases. Hence polynomial addition can be achieved by simple XOR of two numbers.

i.e. A + B = A XOR B.

SUBTRACTION

Addition and subtraction are same operation in F2m. Consider two polynomial $A = x^3 + x^2 + 1$ and $B = x^2 + x$. On polynomial subtraction A – B gives $x^3 – x + 1$. Taking mod 2 over coefficients $A - B = x^3 + x + 1$

On binary representation

A = 1101

B = 0110

A - B = 1011 which is an XOR operation between A and B. This is true in all cases. Hence polynomial subtraction can be achieved by simple XOR of two numbers.

i.e. A - B = A XOR B

MULTIPLICATION

Consider two polynomial $A = x^3 + x^2 + 1$ and $B = x^2 + x$. On polynomial multiplication A * B gives $x^5 + x^3 + x^2 + x$. Coefficient are reduced to mod 2. Since m = 4 the results are to be reduces to a degree less than 4 by irreducible polynomial x4 + x + 1.

i.e. $x^5 + x^3 + x^2 + x$ (mod f(x))

$= (x4 + x + 1)x + x^5 + x^3 + x^2 + x$

$= 2x^5 + x^3 + 2x^2 + 2x$

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

On binary representation

A = 1101

B = 0110

A * B = 1000

DIVISION

The division a/b(mod f(x)) is defined as a * b^{-1} (mod f(x)). b^{-1} is the multiplicative inverse of b over f(x).

MULTIPLICATIVE INVERSE

Multiplicative inverse of number b with respect to irreducible polynomial f(x) is defined as a number b^{-1} such that $b*b^{-1}$ (mod f(x)) = 1. The algorithm such as extended Euclidean algorithm can be used to find the multiplicative inverse of a polynomial efficiently. Finding multiplicative inverse is a costly operation.

3.4 IRREDUCIBLE POLYNOMIAL

Irreducible polynomial is an analogue to modulus p in modular arithmetic. Irreducible polynomial is a polynomial of degree m that cannot be expressed as the product of two polynomials of lesser degree. 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. An example is shown in multiplication section above. In many standard implementation of elliptic curve operation, for making polynomial reduction more efficient the irreducible polynomial is chosen to be trinomial (polynomial containing 3 terms) or pentanomial (polynomial containing 5 terms).

4. ALGEBRA AND NUMBER THEORY

Algebra and Number Theory[27] are the mathematical foundation of Modern Cryptography. Numerous cryptographic algorithms are designed. They are also the corner stone of (provable) security of cryptographic schemes.

We use the following notations. A prime number *p* is called a safe prime if $p = 2p0 + 1$, such that *p0* is also a prime number. An integer *n* is called an RSA modulus if *n* is a product of two primes of equal size. An integer *n* is called a safe-prime product. There is a unique field of order p^n for every prime *p* and every positive integer *n*, up to isomorphism.

In detail, the finite fields are classified as follows :-

- The **order**, or number of elements, of a finite field is of the form p^n, where *p* is a prime number called the **characteristic** of the field, and *n* is a positive integer.
- For every prime number *p* and positive integer *n*, there exists a finite field with p^n elements.
- Any two finite fields with the same number of elements are isomorphic. That is, under some renaming of the elements of one of these, both its addition and multiplication tables become identical to the corresponding tables of the other one.

This classification justifies using a naming scheme for finite fields that specifies only the order of the field. One notation for a finite field is $\mathbf{F}_{p^n}$. Another notation is $\mathbf{GF}(p^n)$, where the letters "GF" stand for "Galois field".

Groups and modular arithmetic in Zn.

Mathematical "groups" play a decisive role in number theory and cryptography. We only talk of groups if, for a defined set and a defined relation (an operation such as addition or multiplication), the following properties are fulfilled:

- The set is closed
- A neutral element exists
- An inverse element exists for each element
- The associative law applies.
- The abbreviated mathematical notation is (G, +) or (G, *).

Definition. Zn: Zn comprises all numbers from 0 to $n - 1$: $Zn = \{0, 1, 2, \cdots, n - 2, n - 1\}$.Zn is an often used finite group of the natural numbers. It is sometimes also called the remainder set R modulo n.

For example, 32-bit computers (standard PCs) only directly work with whole numbers in a finite set, that is the value range $0, 1, 2, \cdots, 232 - 1$.

This value range is equivalent to the set Z232 .

4.1 GROUPS

First recall the definition of a group (a cyclic group in particular) and some other related notions. A group is a set G together with an associative binary operation * on elements of G such that G contains an identity element for * and every element has an inverse under *. If * is commutative, the group is called abelian or commutative. Often, a group is denoted by (G, *) or simply by G. A group G is called finite if |G| is finite. The number of elements of a finite group is called its order.

4.2 THE ABELIAN GROUP

Given two points P,Q in E(Fp), there is a third point, denoted by P+Q on E(Fp), and the following relations hold for all P,Q,R in E(Fp)

- $P + Q = Q + P$ (commutatively)
- $(P + Q) + R = P + (Q + R)$ (associability)
- $P + O = O + P = P$ (existence of an identity element)
- there exists $(-P)$ such that $-P + P = P + (-P) = O$ (existence of inverses)

Addition in a group

If we define the operation mod+ on such a set where $a \bmod + b := (a + b) \pmod n$, then the set Zn together with the relation mod+ is a group because the following properties of a group are valid for all elements in Zn:

- a mod + b is an element of Zn (the set is closed),
- $(a \bmod + b) \bmod + c \equiv a \bmod + (b \bmod + c)$ (mod+ is associative),
- the neutral element is 0.

Each element a 2 E Zn has an inverse for this operation, namely $n - a$ (because $a \bmod + (n - a) \equiv a + (n - a) \pmod n \equiv n \equiv 0 \pmod n$).

Since the operation is commutative, i.e. $(a \bmod + b) = (b \bmod + a)$, this structure is actually a "commutative group".

Multiplication in a group

If we define the operation mod* on the set Zn where $a \bmod * b := (a * b) \pmod n$, then Zn together with this operation is usually not a group because not all properties are fulfilled for each n.

Examples:

a) In Z_{15}, for example, the element 5 does not have an inverse. That is to say, there is no a with $5 * a * 1 \pmod{15}$. Each modulo product with 5 on this set gives 5, 10 or 0.

b) In $Z_{55} \setminus \{0\}$, for example, the elements 5 and 11 do not have multiplicative inverses. That is to say, there is no a $2 Z_{55}$ such that $5 * a * 1 \pmod{55}$ and no a such that $11 * a * 1 \pmod{55}$. This is because 5 and 11 are not relatively prime to 55. Each modulo product with 5 on this set gives 5, 10, 15, . . . , 50 or 0. Each modulo product with 11 on this set gives 11, 22, 33, 44 or 0. On the other hand, there are subsets of Zn that form a group with the operation mod*. If we choose all elements in Zn that are relatively prime to n, then this set forms a group with the operation mod*. We call this set Z*n.

4.3 FINITE FIELDS

The elliptic curve operations defined above are on real numbers. Operations over the real numbers are slow and inaccurate due to round-off error. Cryptographic operations need to be faster and accurate. To make operations on elliptic curve accurate and more efficient, the curve cryptography is defined over two finite fields.

- **Prime field Fp and**
- **Binary field F2m**

The field is chosen with finitely large number of points suited for cryptographic operations. The operations in these sections are defined on affine coordinate system. Affine coordinate system is the normal coordinate system that we are familiar with in which each point in the coordinate system is represented by the vector (x, y).

Graphically finite field can be as shown:-

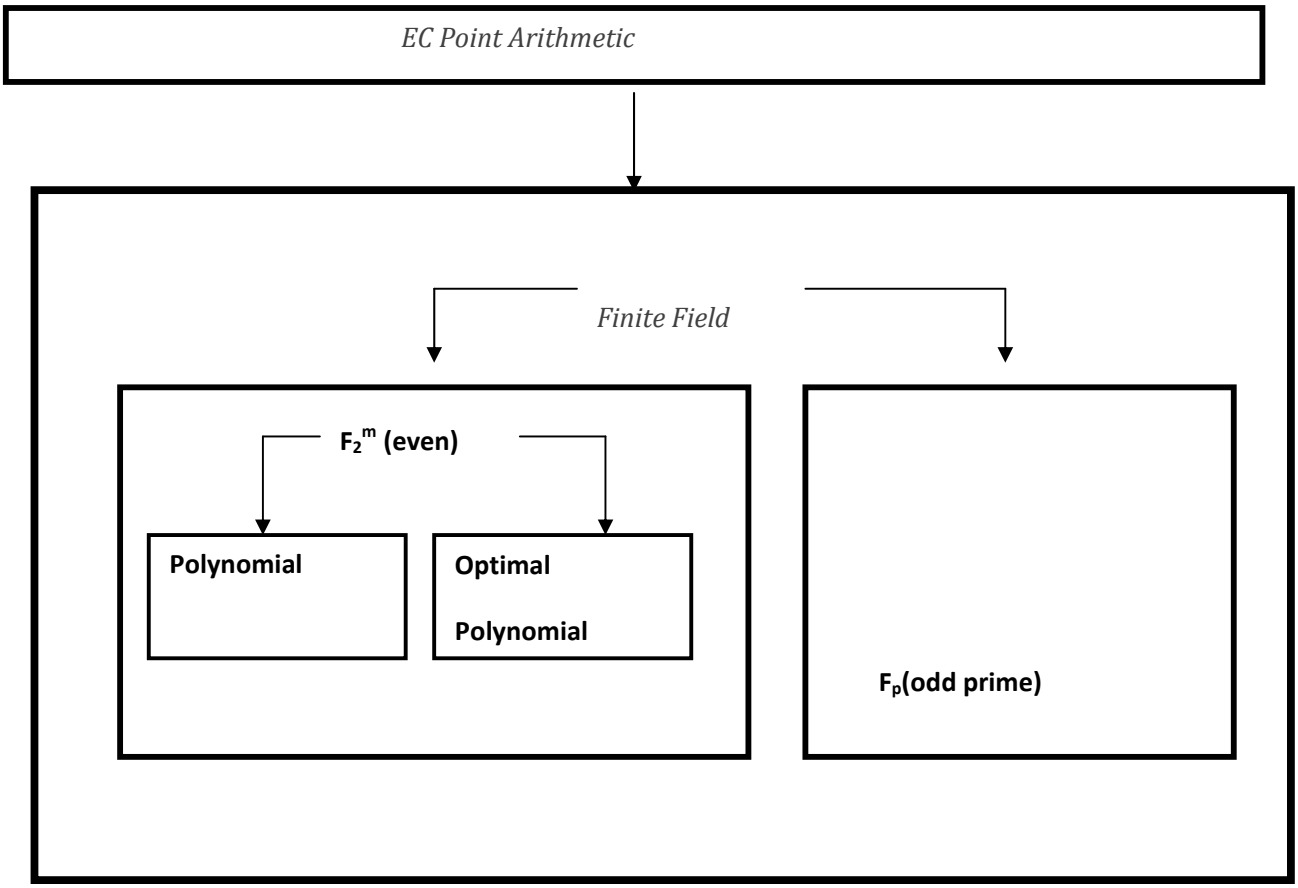

Figure 9 – Finite field

4.4 EC ON PRIME FIELD F_P

The equation of the elliptic curve on a prime field Fp[30] is $y^2 \bmod p = x^3 + ax + b \bmod p$, where $4a^3 + 27b^2 \bmod p \neq 0$. Here the elements of the finite field are integers between 0 and p – 1. All the operations such as addition, substation, division, multiplication involves integers between 0 and p – 1. This is modular arithmetic and is defined in details. The prime number p is chosen such that there is finitely large number of points on the elliptic curve to make the cryptosystem secure. The graph for this elliptic curve equation is not a smooth curve. Hence the geometrical explanation of point addition and doubling as in real numbers will not work here. However, the algebraic rules for point addition and point doubling can be adapted for elliptic curves over Fp.

The way that the elliptic curve operations are defined is what gives ECC its higher security at smaller key sizes.

An elliptic curve is defined in a standard, two dimensional x,y Cartesian coordinate system by an equation of the form:

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

The graphs turns out to be gently looping lines of various forms.

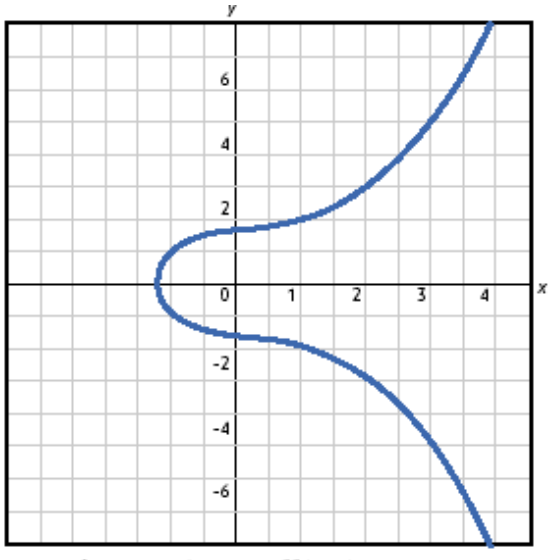

Figure 10 - An elliptic curve

In elliptic curve cryptosystems, the elliptic curve is used to define the members of the set over which the group is calculated, as well as the operations between them, which

define how math works in the group. It's done as follows: imagine a graph labeled along both axes with the numbers of a large prime field. That is to say: a square graph, p x p in size, where p is a very large prime number. Fp is the field of integers modulo p, and consists of all the integers from 0 to p-1.

Now the prime numbers actually employed in practical ECC implementations are quite large, so it's difficult to visualize this graph if you use the real kinds of numbers used. But as an exercise, you can imagine a more comprehensible prime — such as 17. So you'd be looking at graph 17x17 units in size. Now if you define an elliptic curve — an equation of the form given above — so that there are points (x, y) on the curve that satisfy the condition that both x and y are members of the prime field, you have implicitly created a group from the set of integer points on the curve; it is a subset of all the points in the p by p matrix created when you drew the graph — specifically the ones the curve passes directly through.

Note that unlike the groups used in Diffie Hellman, the elements of the set aren't integers, but points. But the system that will result is still going to be, in most senses, the same, familiar arithmetic system as those discussed above. It contains a set of elements (points, in this case), and when you add one point to another, or subtract one from another, there are rules that say what point in the set you wind up at when you do so — just as for the integers in the groups used in Diffie Hellman.

4.5 POINT ADDITION

Point addition is the addition of two points P and Q on an elliptic curve to obtain another point L on the same elliptic curve.

4.6 GEOMETRICAL EXPLANATION

Adding Points P and Q

Point addition(when $Q \neq -P$) Point addition(when $Q \neq -P$)

Consider two points P and Q on an elliptic curve as shown in figure11 (a). If $Q \neq -P$ then a line drawn through the points P and Q will intersect the elliptic curve at exactly one more point –R. The reflection of the point –R with respect to x-axis gives the point R, which is the result of addition of points P and Q.

Thus on an elliptic curve R = P + Q.

If Q = -P the line through this point intersect at a point at infinity O. Hence P + (-P) = O.

This is shown in figure 11(b). O is the additive identity of the elliptic curve group. A negative of a point is the reflection of that point with respect to x-axis.

5.1 ANALYTICAL EXPLANATION

Consider two distinct points J and K such that J = (xJ, yJ) and K = (xK, yK)

Let L = J + K where L = (xL, yL), then

xL = s2 - xJ – xK

yL = -yJ + s (xJ – xL)

s = (yJ – yK)/(xJ – xK), s is the slope of the line through J and K.

If K = -J i.e. K = (xJ, -yJ) then J + K = O. where O is the point at infinity.

If K = J then J + K = 2J then point doubling equations are used.

Also J + K = K + J

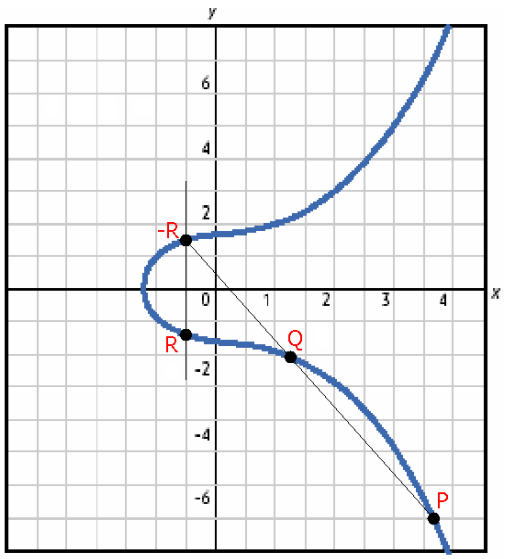

Figure 11(a)

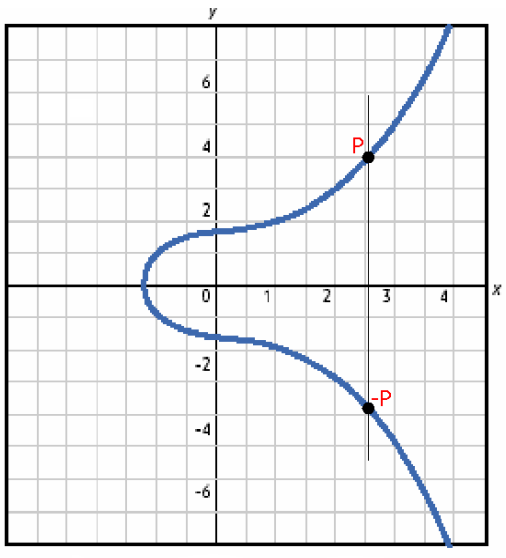

Figure 11(b)

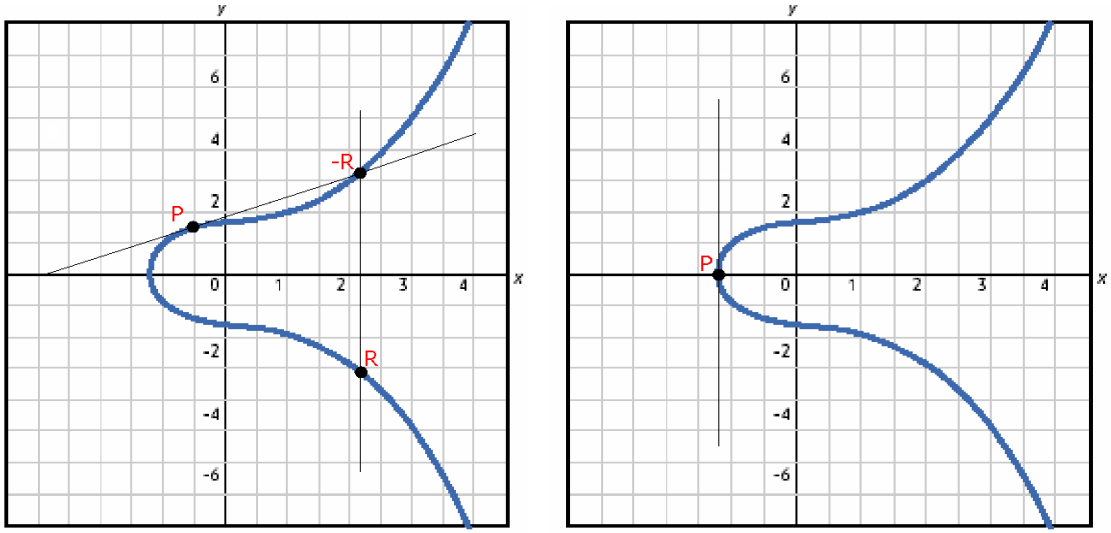

Figure 12(b)

5.1.1 Point doubling

Point doubling is the addition of a point P on the elliptic curve to itself to obtain another point R on the same elliptic curve. To double a point P to get R, i.e. to find R = 2P, consider a point P on an elliptic curve as shown in figure12 (a). If y coordinate of the point P is not zero then the tangent line at P will intersect the elliptic curve at exactly one more point –R. The reflection of the point –R with respect to x-axis gives the point L, which is the result of doubling the point P. Thus R = 2P.

If y coordinate of the point P is zero then the tangent at this point intersects at a point at infinity O. Hence 2P = O when yP = 0. This is shown in figure12 (b).

Point doubling (when Py ≠ 0)Point doubling (when Py =0)

5.1.2 ANALYTICAL EXPLANATION

Consider a point J such that J = (xJ, yJ), where yJ ≠ 0

Let L = 2J where L = (xL, yL), Then

xL = s2 – 2xJ

yL = -yJ + s(xJ - xL)

If yJ = 0 then 2J = O, where O is the point at infinity

5.1.3 POINT SUBTRACTION

Consider two distinct points J and K such that J = (xJ, yJ) and K = (xK, yK)

Then J - K = J + (-K) where -K = (xk, -yk mod p)

Point subtraction is used in certain implementation of point multiplication.

5.2 EC ON BINARY FIELD F_2M

The equation of the elliptic curve on a binary field F_2m is $y^2 + xy = x^3 + ax^2 + b$, where $b \neq 0$. Here the elements of the finite field are integers of length at most 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. All the operation such as addition, substation, division, multiplication involves polynomials of degree m – 1 or lesser.. The m is chosen such that there is finitely large number of points on the elliptic curve to make the cryptosystem secure. SEC specifies curves with m ranging between 113-571 bits . The graph for this equation is not a smooth curve. Hence the geometrical explanation of point addition and doubling as in real numbers will not work here. However, the algebraic rules for point addition and point doubling can be adapted for elliptic curves over F_2m.

5.2.1 Point Addition

Consider two distinct points J and K such that J = (xJ, yJ) and K = (xK, yK)

Let L = J + K where L = (xL, yL), then

xL = s2 + s + xJ + xK + a

yL = s (xJ + xL) + xL + yJ

s = (yJ + yK)/(xJ + xK), s is the slope of the line through J and K.

If K = -J i.e. K = (xJ, xJ + yJ) then J + K = O. where O is the point at infinity.

If K = J then J + K = 2J then point doubling equations are used.

Also J + K = K + J

5.2.2 Point Subtraction

Consider two distinct points J and K such that J = (xJ, yJ) and K = (xK, yK)

Then J - K = J + (-K) where -K = (xk, xk + yk)

Point subtraction is used in certain implementation of point multiplication.

5.2.3 Point Doubling

Consider a point J such that J = (xJ, yJ), where xJ ≠ 0

Let L = 2J where L = (xL, yL), Then

xL = s2 + s + a

yL = xJ2 + (s + 1)*xL

s = xJ + yJ/ xJ, s is the tangent at point J and a is one of the parameters chosen with the elliptic curve If xJ = 0 then 2J = O, where O is the point at infinity.

5.2.4 Point Multiplication

The dominant operation in ECC cryptographic schemes is point multiplication. This is the operation which is the key to the use of elliptic curves for asymmetric cryptography — the critical operation which is itself fairly simple, but whose inverse (the elliptic curve discrete logarithm problem — defined below) is very difficult. ECC arranges itself so that when you wish to perform an operation the cryptosystem should make easy — encrypting a message with the public key, decrypting it with the private key — the operation you are performing is point multiplication.

Point multiplication is simply calculating kP, where k is an integer and P is a point on the elliptic curve defined in the prime field. In terms of the addition operation we defined above, and the corresponding diagram, you can see how the following would look: **take a point, add it to itself (doubling).** Then take the result, and the original point, and add them together again, using the chord and tangent rule. Then take that result, and the original point again, and use the chord and tangent rule yet again. And so on — doing one doubling, and k-2 chord and tangent additions, until you've added P to itself k-1 times, giving kP.

Now if the only way of doing this were in fact to repeat those precise operations — finding the points P, 2P, 3P, and so on up to kP — elliptic curves would be useless for cryptography, since the operation which searches for k given only P and kP (see the elliptic curve discrete logarithm problem, below) would be no harder than doing this. You could thus search for k from kP as quickly as you could calculate kP directly given p and k.

However, there are shortcuts for point multiplication. Given the known shape of the curve, there are in fact several algorithms available which run in considerably less time than would such a stepwise operation. Which of them you choose to use depends on a number of factors - including which calculations you might be able to do ahead of time (which is practical for some cryptographic protocol purposes, in which P is known ahead of time), how much RAM you can set aside for lookup tables, and that sort of thing.

None of the operations is exactly what you'd call trivial. But all of them are vastly easier than doing it by stepwise addition — easier by many orders of magnitude. And they also run in near constant time for a given field size, regardless of what k and P you feed them as input. Some of the fastest working are in the NIST P192 curve, defined using the prime $2^{192}-2^{64}+1$. Here, you are able to get kP in the equivalent of 38 addition and 192 doubling operations, when counting only those which are usable even when P isn't known ahead of time. You can do better still, in fact, in hardware implementations, in fields of order 2^m. In these fields-the fields called binary fields, or characteristic two finite fields hardware implementations that take advantages of opportunities for parallel processing, multiplication has been accelerated. Now these still aren't trivial operations. But the important thing is this: compared to what the attacker has to do to get k back from kP, it's nothing. Which brings us to the inverse operation.

5.3 ELLIPTIC CURVE CRYPTOGRAPHY (DISCRETE LOGARITHM PROBLEM)

The security of ECC depends on the difficulty of Elliptic Curve Discrete Logarithm Problem. Let P and Q be two points on an elliptic curve such that kP = Q, where k is a scalar. Given P and Q, it is computationally infeasible to obtain k, if k is sufficiently large. k is the discrete logarithm of Q to the base P. Hence the main operation involved in ECC is point multiplication. i.e. multiplication of a scalar k with any point P on the curve to obtain another point Q on the curve.

The inverse operation to point multiplication — finding a log in a group defined on an elliptic curve over a prime field — is defined as follows: **given points Q and P, find the integer k such that Q=kP**.

This is the elliptic curve discrete logarithm problem — and this is the inverse operation in the cryptosystem — the one you effectively have to perform to get the plaintext back from the cipher text, given only the public key.

Now naively the obvious, certain way of finding k would be to perform repeated addition — operations — stepping through P, 2P, 3P, and so on, until you find kP. You'd start by doubling P, then adding P to 2P finding 3P, then 3P to P finding 4P and so on. This is the brute force method. The trouble with this is if you use a large enough prime field, the number of possible values for k becomes inconveniently large. So inconveniently large that it's quite practical to create a sufficiently large prime field that searching through the possible values of k would take all the processor time currently available on the planet thousands of years. Though there is a bit more to the story we have to get to now, to distinguish between how difficult it is to break ECC versus how difficult it is to break Diffie Hellman and RSA.

Example of Point multiplication:-

In point multiplication a point P on the elliptic curve is multiplied with a scalar k using elliptic curve equation to obtain another point Q on the same elliptic curve. i.e. kP=Q

Point multiplication is achieved by two basic elliptic curve operations

- Point addition, adding two points J and K to obtain another point L i.e., L = J + K.
- Point doubling, adding a point J to itself to obtain another point L i.e. L = 2J.

Here is a simple example of point multiplication.

Let P be a point on an elliptic curve. Let k be a scalar that is multiplied with the point P to obtain another point Q on the curve. i.e. to find Q = kP.

If k = 23 then kP = 23.P = 2(2(2(2P) + P) + P) + P.

Thus point multiplication uses point addition and point doubling repeatedly to find the result. The above method is called ‘double and add’ method for point multiplication.

There are other efficient methods for point multiplication such as NAF (Non – Adjacent Form) and wNAF (windowed NAF) method for point multiplication.

5.3.1 Elliptic Curve Domain parameters

Apart from the curve parameters a and b, there are other parameters that must be agreed by both parties involved

in secured and trusted communication using ECC. These are domain parameters. The domain parameters for prime fields and binary fields are described below. The generation of domain parameters is out of scope of this paper. Generally the protocols implementing the ECC specify the domain parameters to be used.

5.3.2 Domain parameters for EC over field Fp

The domain parameters for Elliptic curve over Fp are p, a, b, G, n and h. p is the prime number defined for finite field Fp . a and b are the parameters defining the curve $y^2 \bmod p = x^3 + ax + b \bmod p$. G is the generator point (xG, yG), a point on the elliptic curve chosen for cryptographic operations and n is the order of the elliptic curve. The scalar for point multiplication is chosen as a number between 0 and n – 1. h is the cofactor where h = #E(Fp)/n. #E(Fp) is the number of points on an elliptic curve.

5.3.3 Domain parameters for EC over field F2m

The domain parameters for elliptic curve over F_2m are m, f(x), a, b, G, n and h.m is an integer defined for finite field F_2m. The elements of the finite field F_2m are integers of length at most m bits. f(x) is the irreducible polynomial of degree m used for elliptic curve operations. a and b are the parameters defining the curve $y^2 + xy = x^3 + ax^2 + b$. G is the generator point (xG, yG), a point on the elliptic curve chosen for cryptographic operations. n is the order of the elliptic curve. The scalar for point multiplication is chosen as a number between 0 and n – 1. h is the cofactor

where h = #E(F2m)/n. #E(F2m) is the number of points on an elliptic curve.

Example:-(Elliptic curve over F23) Let p=23 and consider an elliptic curve E=>

$Y^2 = x^3 + x + 4$ defined over F23.

6. IMPLEMENTATION ISSUES

Generation of Elliptic Curve on a finite set of Integers

Consider an elliptic curve:- $y^2 = x^3 + ax + b$ (**mod p**) such that $4a^3 + 27b^2 \neq 0$

Let domain parameters are a = 2, b = 3, & p = 5 then the curve is-

$y^2 = x^3 + 2x + 3$ (**mod 5**)

Now to find no of points on elliptic curve is:-

- x = 0 => $y^2 = 3$ => no solution (mod 5)
- x = 1 => $y^2 = 6 = 1$ => y = 1,4 (mod 5)
- x = 2 => $y^2 = 15 = 0$ => y = 0 (mod 5)
- x = 3 => $y^2 = 36 = 1$ => y = 1,4 (mod 5)
- x = 4 => $y^2 = 75 = 0$ => y = 0 (mod 5)

Then points on the elliptic curve are:-

(1,1) (1,4) (2,0) (3,1) (3,4) (4,0) and the point at infinity.

Point Counting

The order of E (Fp) is denoted as # E(Fp).

Determining #E (Fp) is an important problem, called point counting.

Hesse'e Theorem

$P + 1 - 2\sqrt{p} \leq \#E(Fp) \leq p + 1 + 2\sqrt{p}$

6.1 Elliptic Curve & Finite Field

Elliptic curve calculations are usually defined over finite field The finite field is prime field GF(P) The elements are {0,1,2,…,p-1} all operations are modulo p

- The finite field is a binary polynomial field GF(2m)
- The elements are binary polynomials all operations are modulo 2

x = am-1Xm-1 + am-2Xm-2 + … + a1X + a0 ; ai = {0,1}

Defining the curve over Binary Field will speed up the calculations

6.2 Elliptic Curve Scalar Multiplication

- Scalar multiplication is the dominant computation part of ECC
- It computes k×P for a given point P and integer k.
- Q = k×P = (P + P + … + P) ((k-1) addition)
- There are different methods for speeding up this process, The most common
- one is the Binary Method (also called Double and Add Method)

6.3 ECDSA (Signature Generation & Verification)

The EC algorithms are specified above. An overview of EC cryptographic algorithms for key agreement and digital signature are explained below.

6.4 ECDSA - Elliptic Curve Digital Signature Algorithm

Signature algorithm is used for authenticating a device or a message sent by the device. For example consider two devices A and B. To authenticate a message sent by A, the device A signs the message using its private key. The device A sends the message and the signature to the device B. This signature can be verified only by using the public key of device A. Since

the device B knows A's public key, it can verify whether the message is indeed send by A or not. ECDSA is a variant of the Digital Signature Algorithm (DSA) that operates on elliptic curve groups. For sending a signed message from A to B, both have to agree up on Elliptic Curve domain parameters. The domain parameters are defined in section 9. Sender 'A' have a key pair consisting of a private key dA (a randomly selected integer less than n, where n is the order of the curve, an elliptic curve domain parameter) and a public key QA = dA * G (G is the generator point, an elliptic curve domain parameter). An overview of ECDSA process is defined below.

6.4.1 Signature Generation

For signing a message m by sender A, using A's private key d_A

1. Calculate e = HASH (m), where HASH is a cryptographic hash function, such as SHA-1
2. Select a random integer k from [1,n − 1]
3. Calculate r = x1 (mod n), where (x1, y1) = k * G. If r = 0, go to step 2
4. Calculate $s = k^{-1}(e + d_A r) \pmod n$. If s = 0, go to step 2
5. The signature is the pair (r, s)

6.4.2 Signature Verification

For B to authenticate A's signature, B must have A's public key QA

1. Verify that r and s are integers in [1,n − 1]. If not, the signature is invalid
2. Calculate e = HASH (m), where HASH is the same function used in the signature generation.
3. Calculate $w = s^{-1} \pmod n$
4. Calculate u1 = ew (mod n) and u2 = rw (mod n)
5. Calculate (x1, y1) = u1G + u2QA
6. The signature is valid if x1 = r(mod n), invalid otherwise

Size The size of an ECC public key, ECDSA signature and an ECIES encryption will be compared with those produced by an RSA system.

Public Key Size

An RSA public key consists of an ordered pair *(n,e)* where *n* is a composite number, called the modulus, and *e* is the public exponent. In a 1024-bit RSA system, *n* will have 1024 bits. A common value for the public exponent is *e*=216+1. This is the value that Entrust uses. Thus, an Entrust RSA public key would require 128 bytes for the modulus and 3 bytes for the public exponent. The total size is then 131 bytes. An ECC public key consists of a point on the elliptic curve. Each point is represented by an ordered pair of elements *(x,y)* each with 192 bits. For a 192-bit elliptic curve, the public key would then be represented by two 24-byte values, giving a total key size of 48 bytes.1As can be seen from the numbers above, ECC does provide a significant reduction in public key size. This reduction can be crucial in many severely constrained environments where large public keys are not possible. However, in a PKI using X.509 certificates, the effect of using the smaller public keys is minimal. A typical size for an X.509 certificate would be about 1K (~1000 bytes). Thus, changing a user's public key from RSA or DSA to ECC would reduce his/her certificate size by less than 10%.Another important point to keep in mind is that each ECC public key is only valid in the context of certain parameters. These parameters must also be specified and transferred with integrity to the public key recipient (e.g. within an X.509 certificate). While there do exist certain curves which can be represented using short identifiers, in the general case, it will require an additional five 192-bit (24-byte) quantities to specify these parameters. Thus, it could take up to 110 additional bytes. RSA does not require any parameters be transferred with the public key.

Signature Size

An RSA signature consists of a single 1024 bit value. Thus, it can be represented in 128 bytes. An ECDSA signature consists of two 192-bit values. Thus, it can be represented using two 24-byte values, for a total signature size of 48 bytes.2 1 A method does exist to reduce the size of ECC public keys by almost a factor of 2. This method, called **point compression**, is a proprietary technique that is not available to all implementers of ECC, thus to ensure interoperability it is not recommended. For this reason, the size estimates given assume no point compression has been performed. If point compression was used, a public key could be represented using one 192-bit value and one additional bit. This would then require (24+1=) 25 bytes. Again, the reduction in signature size is substantial and may be important for many constrained environments. However, as with public key size, the difference represents less than 10% of the size of a public key certificate. For larger signed messages, the difference would represent an even smaller percentage of the overall message.

Encryption Size

This section will compare the difference in size in transporting a 128 bit symmetric key using RSA and ECIES. This is the typical scenario when files are encrypted, for example. The encryption algorithm ECIES is specified in the ANSI X9.63 draft . A 128 bit symmetric key encrypted using RSA will consist of one 1024 bit value. Thus, it can be represented using 128 bytes. A 128 bit symmetric key encrypted using ECIES will consist of an elliptic curve point, a 128-bit value and a 160-bit value. The elliptic curve point consists of two 192-bit values, so it can be represented using two 24-byte values, or 48 bytes.3 The 128-bit value can be represented using 16 bytes and the 160-bit value can be represented using 20 bytes. Thus the encrypted symmetric key requires 84 bytes. While ECIES does indeed produce smaller encrypted values than RSA, the difference is not as dramatic as for public keys and signature values. When considering that the symmetric key will then usually be used to encrypt much larger files, the advantage may become inconsequential.

6.4 Comparison with RSA

This section compares ECC public key sizes, signature and encryption lengths, and speed with those of RSA. Typical usage scenarios will be used to describe the effect these have on various implementations. The ECC system under consideration will use an odd characteristic 192-bit elliptic curve, which is the default used by the Entrust product line. The RSA system will use 1024-bit keys, which is also the default in the Entrust product line.

6.5.1 Run-time Comparisons

To test and compare the performance characteristics of the RSA and ECDSA signature algorithms, we independently tested each of the three main components: key generation, signature generation and signature verification. Since ECC offers security equivalent to RSA using much smaller key sizes, the performances were tested according to the following table, suggested from .

Symmetric	ECC	RSA
80	163	1024
112	233	2240
128	283	3072
192	409	7680
256	571	15360

Table 2 -Comparable key sizes (in bits)

6.5.2 Tests were performed on an Intel P4 2.0 GHz machine with 512MB of RAM.

The message used for signing is a 100 KB text file

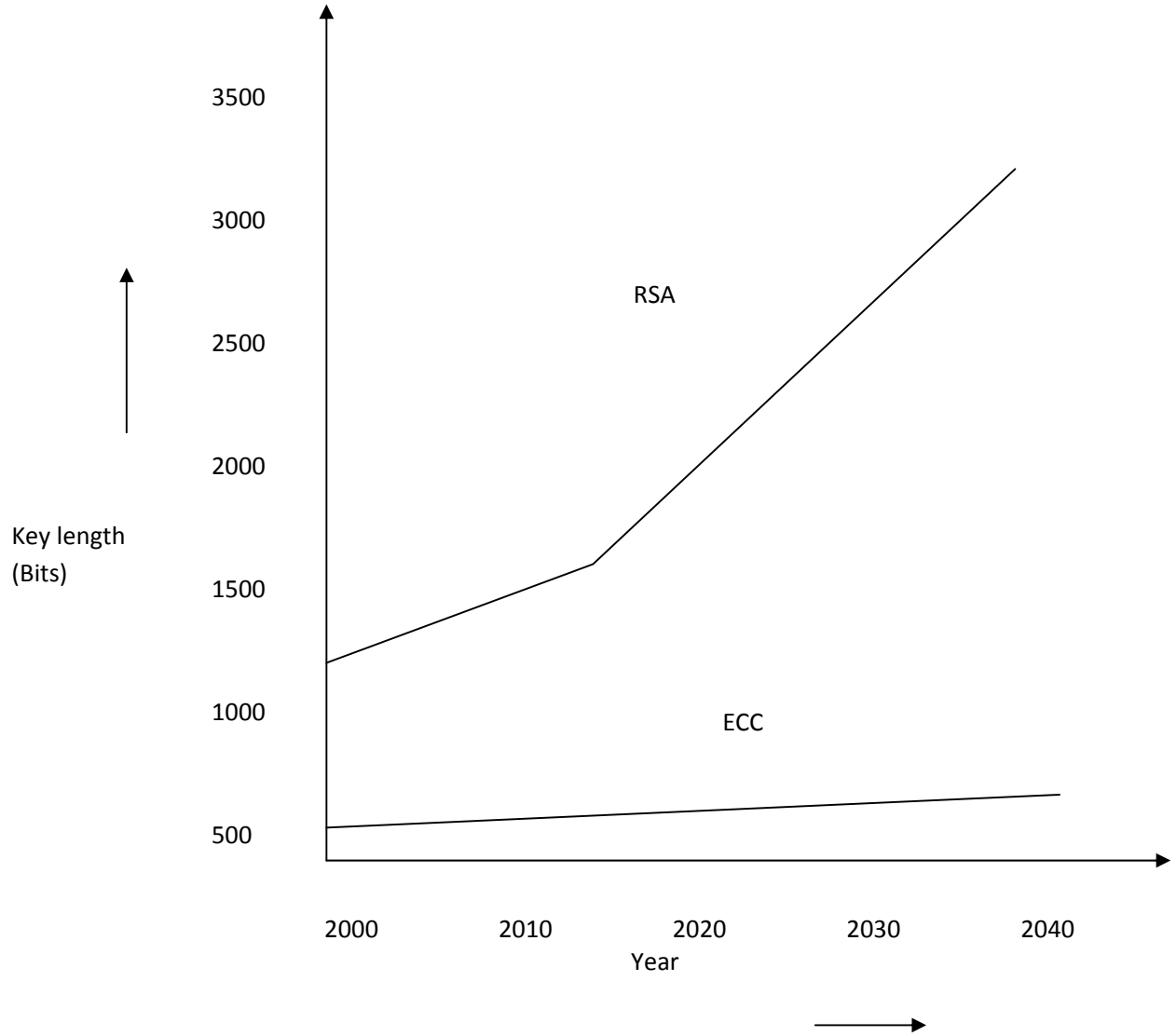

Figure 13 - Comparison b/w ECC & RSA

6.6 **Security of ECC**

One of the advantages of using elliptic curve based cryptographic systems instead of integer factorization or discrete logarithm based methods is that they provide similar security levels using smaller key lengths. Why is this? As mentioned in the previous section, the security of any public-key based cryptography is based upon the difficulty of solving certain mathematical problems. Thus, we can determine the amount of effort that would be required to break one of these public-key systems by looking at the effort required to solve these hard problems, using the best algorithms, software and hardware which are known. It should be noted that in the future new solutions to any of these problems might be discovered that drastically change the amount of effort required to solve them. The analysis below is based on the best methods known today.

Most people consider the integer factorization and discrete logarithm problems to have approximately equivalent security. Both of these problems have undergone intensive review and study by many of the world's top mathematicians and cryptographers. This can give us a sense of comfort that these problems are, in fact, difficult to solve. Actually, the best method known to solve each of these problems is the Number Field Sieve (NFS). The NFS is what is known as a **sub-exponential time method**. This means that the problem can be considered hard to solve, but not as hard as problems that only allow fully exponential solutions. It is generally accepted that, based on the difficulty of solving the integer factorization problem and discrete logarithm problem, RSA, DSA and Diffie-Hellman keys should be at least 1024 bits long and that for very long-term security (20 years or more) 2048 bit keys should be used. Recently a large-scale effort was able to factor a 512-bit integer, thus showing that keys of this size are vulnerable to attack by large, sophisticated adversaries.

On the other hand, solving the ECDLP is generally considered to be a much more difficult problem than factoring integers or solving the discrete logarithm problem. Because of the structure that is inherent within an elliptic curve, the types of solutions to these problems do not seem to apply to the ECDLP. The best method known to solve the ECDLP is an elliptic curve version of an attack developed by Entrust researchers for the discrete logarithm problem, known as the parallel collision search method. This method is **fully exponential**, which means that that the ECDLP can be considered among the hardest types of problems to solve, using the best methods known today. One of the consequences of the ECDLP only having a fully exponential solution is that for every two additional bits of key used, attacking that key requires twice as much effort. Thus, attacking a 193-bit elliptic curve public key requires twice as much effort as attacking a 191-bit key. Because it is relatively new, the ECDLP has not received as much attention from mathematicians and cryptographers as the integer factorization and discrete logarithm problem. Although, within the past few years, that has begun to change and a great deal of effort has been made at attempting to solve this problem. Since a great deal of research is still ongoing, it is difficult to directly compare the security levels provided by ECC with those provided by RSA, DSA and Diffie-Hellman, for example. However, it seems reasonable that for security equivalent to an RSA key with 1024 bits, one should use an elliptic curve with about 170 bits and that for security equivalent to an RSA key with 2048 bits, one should use an elliptic curve with about 230 bits. The above discussion on the difficulty of attacking an ECC public key assumes that certain weak cases have been avoided when constructing the elliptic curve parameters. There are certain elliptic curves that are known to produce cryptographic systems with a substantially lower security level than the general case described above.

7.CONCLUSION

And this, in the end, is the reason ECC is a stronger option than the RSA and discrete logarithm systems for the future. And this, in the end, is why ECC is such an excellent choice for doing asymmetric cryptography in portable, necessarily constrained devices right now.

As an example: as of this writing, a popular, recommended RSA key size for most applications is 2,048 bits. For equivalent security using ECC, you need a key of only 224 bits. The difference becomes more and more pronounced as security levels increase (and, as a corollary, as hardware gets faster, and the recommended key sizes must be increased). A 384 -bit ECC key matches a 7680-bit RSA key for security.

The smaller ECC keys mean the cryptographic operations that must be performed by the communicating devices can be squeezed into considerably smaller hardware, that software applications may complete cryptographic operations with fewer processor cycles, and operations can be performed that much faster, while still guaranteeing equivalent security. This means, in turn, less heat, less power consumption, less real estate consumed on the printed circuit board, and software applications that run more rapidly and make lower memory demands. Leading in turn to more portable devices which run longer, and produce less heat.

In short, if you're trying to make your devices smaller—and if you need to do asymmetric cryptography, you need ECC. If you're trying to make them run longer on the same battery, and produce less heat, and you need asymmetric cryptography, you need ECC. And if you want an asymmetric cryptosystem that scales for the future, you want ECC. And if you just want the most elegant, most efficient asymmetric cryptosystem going, you want ECC. If you just want the most elegant, most efficient asymmetric cryptosystem going, you want ECC.

For efficient implementation of ECC, it is important for the point multiplication algorithm and the underlying field arithmetic to be efficient. There are different methods for efficient implementation point multiplication and field arithmetic suited for different hardware configurations. Implementation of ECC using projective coordinates has shown considerable improvement in efficiency compared to the affine coordinate implementation. This improvement in efficiency is due to the elimination of multiplicative inverse operation in point addition and doubling that would otherwise cost considerable processor cycles.

If the irreducible polynomial in binary field implementation is chosen to be trinomial or pentanomial the implementation of ECC on binary field can be made efficient than the prime field implementation. In SEC specified domain parameters, the irreducible polynomials are either trinomial or pentanomial. These chosen polynomials cause the polynomial reduction in binary field to run much faster than the modular reduction in prime field.

How ECC is the next generation of public key cryptography

Asymmetric cryptography is a marvelous technology. Its uses are many and varied. And when you need it, you need it. For many situations in distributed network environments, asymmetric cryptography is a must during communications. If you're taming key distribution issues with a public key infrastructure (PKI), you're using asymmetric cryptography. If you're designing or employing any kind of network protocol or application requiring secure communications, to come up with a practical solution, you're going to have to use asymmetric cryptography.

Asymmetric cryptography has, in fact, proved so useful for securing communications that it has become pervasive in modern life. Every time you buy something on the Internet, if the vendor is using a secure server, you're using asymmetric cryptography to secure the transaction.

But Asymmetric cryptography is demanding and complex, by its very nature. The hard problems in number theory — the key to the algorithms' functionality — are all intrinsically difficult enough that the processor cycles you must throw at doing it, and/or the chip space you must dedicate to the

implementation, inevitably far outstrip the resources you must dedicate for doing symmetric cryptography.

That's why if you need asymmetric cryptography, you should be considering elliptic curve cryptography (ECC).

- ECC offers considerably greater security for a given key size — something we'll explain at greater length later in this paper.
- 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.
- 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 if you're looking for the cryptosystem that will give you the most security per bit, you want ECC.

This thesis describes elliptic curve cryptography in greater depth — how it works, and why it offers these advantages. It will begin by discussing the larger subject of asymmetric cryptography in general.

REFERENCES

[1] Prof.Vivek Katiyar,A Survey on Elliptic Curve Cryptography for Pervasive Computing Environment, *International Journal of Computer Applications (0975 – 8887) Volume 11– No.10, December 2010*

[2] Prof.Tarun Narayan Shankar, CRYPTOGRAPHY WITH ELLIPTIC CURVES, *International Journal Of Computer Science And Applications Vol. 2, No. 1, April / May 2009*

[3] Prof. Dr. Ersan AKYILDIZ "SCALAR MULTIPLICATION ON ELLIPTIC CURVES" M.Sc., Department of Cryptography YAYLA Supervisor: August 2006.

[4] Don Johnson et al Alfred Menezes and Scott Vanstone "ECC, Future Resiliency and High Security Systems" March 30, 1999 ,Dept. of Combinatorics & Optimization, University of Waterloo,Canada ,mail to:- djohnson@certicom.com

[5] Frankfurt "CrypTool Script Mathematics and Cryptography" The author, 1998-2003 July 17, 2003

[6] Robert Zuccherato " Elliptic Curve Cryptography Support in Entrust "Author: Date: May 9, 2000

[7] Joe Hurd "Elliptic Curve Cryptography A case study in formalization using a higher order logic theorem prover" Course Notes, Oxford University joe.hurd@comlab.ox.ac.uk, 4–5 August 2005

[8] CZESŁAW KO´S CIELNY "A NEW APPROACH TO THE ELGAMAL ENCRYPTION SCHEME" Academy of Management of Legnica, Faculty of Computer Science , Reymonta 21, 59–220 Legnica, Poland e-mail: C.Koscielny@wsm.edu.pl

[9] Don B. Johnson, Certicom "ECC A Future High Security Systems" Revised July 6 , 1999 to correct typo in RSA key generation description and clarify low exponent RSA discussion , e-mail :- djohnson@certicom.com

[10] Nicholas Jansma and Brandon Arrendondo "Performance Comparison of Elliptic Curve and RSA for Digital Signatures" April 28 2004, njansma@engin.umich.edu, barrendo@engin.umich.edu

[11] Christian P¨uhringer cip "High Speed Elliptic Curve Cryptography Processor for GF(p)" 8.7.2005, e-mail: cip@gmx.at

[12] Sharat Narayan "New Generation Cryptosystems using Elliptic Curve Cryptography"

[13] Aleksandar Juri & Alfred J. Menezesy "Elliptic Curves and Cryptography"March 23, 2005, Journal of Cryptology, 3 (1991), 63-79.

[14] Victor Miller [l2] and Neal Koblitz " Elliptic curve cryptography" Hewlett-Packard Laboratories, Palo Alto, CA 94304, U.S.A., 1999, Metsovo, Greece, June 27 - July 1

[15] Schinianakis et al D.M. Kakarountas and A.P.; Stouraitis "A new approach to elliptic curve cryptography" Electrotechnical Conference, 2006. MELECON 2006. IEEE Mediterranean Volume , Issue , 16-19 May 2006.

[16] V. S. Dimitrov *et al* L. Imbert and P. K. Mishra "Fast Elliptic Curve Point Multiplication using Double-Base Chains" University of Calgary, University drive NW Calgary, AB, T2N 1N4, Canada 2 CNRS, LIRMM, UMR 5506 ,161 rue Ada, 34392 Montpellier cedex 5, France.

[17] M. Abdalla, M. Bellare, and P. Rogaway. DHAES: "Elliptic Curve Cryptography" Certicom Research. 1998. secg-talk@lists.certicom.com,http://www-cse.ucsd.edu/users/mihir/

[18] IEEE P1363. *Standard Specifications for Public-Key Cryptography*. Institute of Electrical and Electronics Engineers, 2000.

[19] Dr. Andreas Steffen "The Elliptic Curve Cryptosystem" in the year of 2002, Zürcher Hochschule Winterthur

[20] IEEE P1363A. *Standard Specifications for Public-Key Cryptography: Additional Techniques*. May,2000. Working Draft.

[21] ISO/IEC 14888-3. *Information technology - Security techniques - Digital signatures with appendix- Part 3: Certificate-based mechanisms*.

[22] ISO/IEC 15946-1. *Information technology - Security techniques - Cryptographic techniques based on elliptic curves - Part 1: General*. 1998. Working draft.

[23] ANSI X9.63-199x: *Public Key Cryptography for the Financial Services Industry: Key Agreement and Key Transport Using Elliptic Curve Cryptography*. October, 1999. Working Draft.

[24] ISO/IEC 15946-2. *Information technology - Security techniques - Cryptographic techniques based on elliptic curves - Part 2: Digital signatures*. 1998. Working draft.

[25] M. Abdalla, M. Bellare, and P. Rogaway. DHAES: An encryption scheme based on the Diffie- Hellman problem. 1998. Full version of [11]. Available from: http://www-cse.ucsd.edu/users/mihir/

[26] T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, IT-31: pages 469–472, 1985.

27] Stallings, W. *Cryptography and Network Security*. Prentice Hall, 2003.

[28] ANSI X9.62-1998: *Public Key Cryptography for the Financial Services Industry: the Elliptic Curve Digital Signature Algorithm (ECDSA)*. American Bankers association, 1999.

[29] G. Lay and H. Zimmer. Constructing elliptic curves with given group order over large finite fields. *Algorithmic Number Theory*, pages 250–263, 1994.

[30] Schneier, B. "Elliptic Curve Public Key Cryptography". Cryptogram ENewsletter. November 15, 1999 <http://www.counterpane.com/