Comparative Study of Various Key Exchanging Algorithms

[1]Navita Saini, [2] P.R. Suri, [3]Gurpreet Singh, [4]Sushil Pensia

[1, 2, 3, 4,] Computer Science and Application Department, Kurukshetra University, Kurukshetra

Email: navita_saini2005@yahoo.co.in

***Abstract*— The concept of public key Cryptosystem evolved from an attempt to attack two of the most difficult problems associative with symmetric encryption. The initial problem is that of key distribution. As we have seen, key distribution under symmetric encryption requires either (1) that two communication already share a key, which somehow is distributed to them; or (2) the use of key distribution centre. Finally, there is a feeling that key distribution is trivial when using public key encryption, compared to the rather cumbersome handshaking involved with key distribution canters for symmetric encryption. This paper provides an overview of various public key algorithms and a comparison between them is made on the basis of Key size, security and Cost. First we discuss the RSA algorithm, Diffie Hellman Key Exchange Algorithm and EC Cryptography – (ECDH – Elliptic Curve Diffie Hellman) and a comparison between them is made on the basis of Key size, security and Cost.**

KEYWORDS— RSA ALGORITHM, DIFFIE-HELLMAN KEY EXCHANGE ALGORITHM, ELLIPTIC CURVE CRYPTOGRAPHY (ECC*)*

I. INTRODUCTION

Public-key cryptography [1] refers to a commonly used set of methods for transforming a message into a form that can be read only by the intended recipient. This cryptographic approach involves the use of asymmetric key algorithms that is, the non-message information (the public key) needed to change the message to a protected form is different from the information required to reverse the process (the private key).

Thus, unlike symmetric key algorithms [7], a public key algorithm does not involve a secure initial exchange of one or more secret keys between the sender and receiver. The Standard RSA, which is considered to be one of the best Cryptographic Systems built.

II. RSA ALGORITHM

RSA [2] is a public-key cryptosystem defined by Rivest, Shamir, and Adleman in 1977. Any of the two related keys can be used for encryption, with the other is used for reverse process i.e. decryption. The RSA algorithm involves three basic steps: key generation, encryption and decryption

A. *Key Generation*

- Generate two large prime numbers, p and q
- Let $n = pq$
- Let $m = (p-1)(q-1)$
- Choose a small number e, co prime to m
- Find d, such that $de \% m = 1$
- Distribute e and n as the public key. Keep d and n as the secret key.

B. *Encryption*

- $C = P^e \% n$

C. *Decryption*

- $P = C^d \% n$

$x \% y$ means the remainder of x divided by y

In the general case where RSA is used to exchange symmetric keys [8], key encapsulation provides a simpler alternative to padding. Instead of generating a random symmetric key, padding it and then encrypting the padded version through RSA, a random integer m between 1 and n-1 is generated and encrypted directly using RSA. Both the sender and receiver produce identical symmetric keys by applying the same key derivation function to m. Key encapsulation mechanisms (KEMs) are a class of encryption techniques intended to secure symmetric cryptographic key material for transmission using asymmetric (public-key) algorithms. In practice, public key systems are awkward to use in transmitting long messages. Instead they are frequently used to exchange symmetric keys, which are relatively short. The symmetric key is then used to encrypt the large message.

The conventional approach to sending a symmetric key with public key systems is to first generate a random symmetric key and then encrypt it using the selected public key algorithm. The recipient then decrypts the public key message to recover the symmetric key. As the symmetric key is usually small in size, padding is required for full security and proofs of security for padding schemes are often less than complete. KEMs shorten the process by generating a random element in the finite group underlying the public key system and deriving the symmetric key by hashing that element, eliminating the necessitate for padding.

Suppose Alice has transmitted her public key (n,e) to Bob, while keeping her private key secret, as usual. Bob then wishes to transmit symmetric key M to Alice. M might be a 128 or 256-bit AES key, for example. Note that the public key n is usually 1024-bits or even longer, thus much larger than classic symmetric keys. If e is small enough that $M < n^{1/e}$, then the encryption can be rapidly broken using ordinary integer arithmetic. To avoid such potential flaws, Bob first turns M into a larger integer 0 < m < n by using an agreedupon reversible procedure known as a padding scheme, such as OAEP. He then computes the cipher text c equivalent to:

$$c \equiv m^e \pmod{n}.$$

Alice be able to recover m from c by using her private key exponent d by using the following computation:

$$m \equiv c^d \pmod{n}.$$

Given m, she recovers the original message M by reversing the padding scheme. With KEM the method is simplified as follows:

Instead of generating a random symmetric key M, Bob first generates a random m, 0 < m < n. He derives his symmetric key M by M = KDF (m), where KDF is a key derivation function, such as a cryptographic hash. He then computes the cipher text c corresponding to m:

$$c \equiv m^e \pmod{n}.$$

Alice then recovers m from c by using her private key exponent d by the same method as above:

$$m \equiv c^d \pmod{n}.$$

Given m, she is able to recover the symmetric key M by M = KDF (m). The KEM eliminates the difficulty of the padding scheme and the proofs needed to show the padding is secure p. Note that while M can be considered from m in the KEM approach, the reverse is not feasible; assuming the key derivation function is one-way. An attacker who by some means recovers M cannot obtain the plaintext m. using the padding approach, he can. Thus KEM is supposed to encapsulate the key. Note that if the same m is sent to e or more recipients in an encrypted way and the receivers share the same exponent e, but different p, q, and n, then one can decrypt the original clear text message via the Chinese remainder theorem. Thus key encapsulation must not be used directly to send the same symmetric key to multiple recipients. In its place, the common symmetric key can be encrypted via separate symmetric keys (Key Encryption Keys) for each recipient and the encrypted keys then sent individually. Similar techniques are presented for Diffie-Hellman encryption and other public key methods.

A). Is RSA secure?

Nobody knows. An apparent attack on RSA is to factor pq into p and q. Unfortunately nobody has the slightest idea how to prove that factorization is inherently slow. It is easy to formalize what we mean by ``RSA is/isn't strong"; Note that there may even be a `shortcut' to breaking RSA other than factoring. It is obviously adequate but so far not provably essential. That is, the security of the system depends on two significant assumptions:

(1) factoring is required to break the system, and

(2) Factoring is `inherently computationally intractable', or, alternatively, `factoring is hard' and `any approach that can be used to break the system is at least as hard as factoring'.

Previously even professional cryptographers have made mistakes in estimating and depending on the intractability of different computational problems for secure cryptographic properties. For example, a system called a `Knapsack cipher' was in vogue in the literature for years until it was demonstrated that the instances typically generated could be efficiently broken, and the whole area of research fell out of favour.

III. Diffie-Hellman Key Exchange Algorithm

Many cryptographic applications, such as encryption, depend on secret keys. These must be exchanged in advance and this must happen in a secret way, as the name suggests. Let's say Alice and Bob want to exchange a few words via a voice-over-IP telephone. As data is broadcasted across the unsafe Internet, a symmetric algorithm is used for encryption. That is one where both parties apply the same secret key. Let's also assume that this key is a (large) number. Evil Eve would also want to know the secret key so that she is able to decrypt Alice and Bob's communication data. How can Alice and Bob have the same opinion on a key without Eve knowing what it is? They could meet up in a area without Eve and agree on a number. But what if Alice was in India and Bob was in Brazil? In a scenario like this a technique called Diffie-Hellman key exchange [9] is often used. It permits the agreement on a shared secret key via an insecure network (that is one where we have eavesdroppers like Eve) such as the Internet. Before explaining how it works it is necessary to briefly introduce one mathematical term, the primitive root. Let p be a prime. From now on we do all calculations modulo p. This is called modular arithmetic and has similar properties to the ``normal" arithmetic (addition and multiplication). Basically, after every operation we divide the result by p and keep on working with the remainder. Definition: An integer g € {1… p-1} is called a primitive root mod p if for every A € {1,2,….,p-1} there is an integer a such that A ≡ ga mod p.

A) Algorithm Steps:

- Alice and Bob agree on a prime number p and a natural number g such that g is a primitive root mod p. These numbers may be public.
- Alice chooses a secret natural number a, computes A = g^a mod p, and sends A to Bob.
- Bob chooses a secret natural number b, computes B = g^b mod p, and sends B to Alice.
- Alice computes the key $K_A = B^a$ mod p.
- Bob computes the key $K_B = A^b$ mod p.
- The two keys match because $K_A \equiv B^a \equiv (g^b)^a \equiv g^{ba} \equiv g^{ab} \equiv (g^a)^b \equiv A^b \equiv K_B$ mod p.

The number g should be a primitive root mod p because this ensures that there are many different possibilities for the key. In the example above with p=5, choosing g=4 (not a primitive root) gives only the values 1 or 4 for the key, whereas choosing g=2 (a primitive root) yields the values K=1,2,3,4 as possible keys. The more possible values there are, the harder it becomes for Eve to guess the actual key, which is what we want. In practice g is not always compulsory to be a primitive root, but it must be selected in such a way that there are adequately many different values for the key. Such a sensible choice requires additional investigation, which is why theorists often include the primitive root requirement.

B) Security of the Diffie-Hellman Key Exchange

Now consider what information evil Eve can gain during the process of a Diffie-Hellman key exchange [10]. Let's assume she has access to all the communication between the two parties. Hence she knows the values of g, p, A and B, but not those of a or b since a is only known to Alice and b is only known to Bob and they are never exchanged. Eve wants to know the key, i.e. the value g^{ab} mod p. Eve can achieve this by finding a or b and then working out g^{ba} mod p. The task of finding a or b is a well-known problem, called the discrete logarithm problem.

C) *Discrete Logarithm Problem (DLP)*

Given a prime p, a base g and a number $A \equiv g^a$ mod p, find the value of a. The number a is then called the discrete logarithm to base g of A mod p. essentially, the difficulty is to find a suitable exponent a. Calculating logarithms is easy over the real numbers, but a hard problem when using modular arithmetic. Note that the DLP [9] always has a solution if g is a primitive root b mod p, meaning that the powers of g take on all values from 1 to p-1 mod p. In other words for every number A there exists that $g^a \equiv A$ mod p. Still it may be very difficult to find this solution as we will explain soon. If Eve can solve the DLP, she can obtain the secret key and thus break the Diffie-Hellman key exchange protocol. Therefore it is often said that the security of a Diffie-Hellman key exchange depends on the hardness of the DLP. So the most important question to answer at this point is: How difficult is solving the discrete logarithm problem? You may have noticed that solving the DLP may not be the only way of breaking the Diffie-Hellman key exchange protocol. It is not explicitly required that Eve find out a or b, she really just has to find the key $K \equiv g^{ab}$ mod p somehow. If she can do this without the knowledge of a or b, that's fine. This problem is commonly known as the Diffie-Hellman problem (DHP) and may be easier to solve than the DLP. However, the best currently known way of solving the DHP is by solving the DLP. Scientists have been studying these two prominent problems for decades without any significant progress, which is why it is commonly assumed that they are equivalent, and research has been concentrating on the more general DLP. In those years of study, scientists have come to the conclusion that solving the DLP is very difficult. Try solving it by generating a key in the above example and finding the value of a only using your knowledge of p, g and A (be sure you uncheck both boxes). You will notice that as the numbers get larger, the problem becomes almost impossible to solve by hand. The truth is that it is also very difficult to solve using a computer. We will get a grasp of exactly how difficult it is by examining some algorithms which can solve the discrete logarithm problem. A common method of determining the difficulty of a problem is by studying how efficient the algorithms are that can solve it. The efficiency (or complexity) of an algorithm is in turn measured by its running time (i.e. the number of operations it performs) relative to the length in bits of its input. There are several classes of algorithms, three of which we discuss here. The running time of a polynomial-time algorithm can be expressed as a polynomial in the length of its input (e.g. n^2 if n is the length of the input). Polynomial-time algorithms are generally considered efficient or feasible. On the contrary, exponential algorithms have a running time that can only be expressed by a term exponential in the length of the input (e.g. n^2 for an input of length n). Exponential algorithms are considered inefficient or infeasible. Sub-exponential algorithms are neither polynomial-time nor exponential, but somewhere in between. Problems that can only be solved by exponential algorithms are considered very hard. Unfortunately, there is usually no way of proving that there is no sub-exponential or polynomial-time algorithm that solves a certain problem. It is rather an assumption that scientists make when they have studied a problem for years without finding a better algorithm. However, we should always be prepared for the unlikely case that an ingenious person presents a better than the currently known best algorithm. Nevertheless, most people feel save using a cryptographic system if its security relies on a well-studied hard problem in this sense. And no matter how paranoid we are, it's the best we can do at the moment.

IV. Elliptic Curve Cryptography (ECC)

A. *Introduction*

Elliptic Curve Cryptography (ECC) [4] is a public key cryptography. In public key cryptography each user 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 identified 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 need any shared secret between the communicating parties but it is much slower than the private key cryptography.

The mathematical operations of ECC [12] is defined over the elliptic curve $y2 = x3 + ax + b$, where $4a3 + 27b2 \neq 0$. Each value of the 'a' and 'b' provides a different elliptic curve. All points (x, y) which suits 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 acquired by multiplying the private key with the generator point G in the curve. The generator point G, the curve parameters 'a' and 'b', collectively with few more constants represents 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 safe as 1024-bit key in RSA.

B. *Discrete Logarithm Problem*

The security of ECC depends on the difficulty of Elliptic Curve Discrete Logarithm Problem. Suppose 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 find k, if k is sufficiently large. k is the discrete logarithm of Q to the base P. Hence the main operation concerned in ECC is point multiplication. i.e. multiplication of a scalar k with any point P on the curve to find another point Q on the curve.

C. Finite Fields

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

- Prime field Fp and
- Binary field F2 m

The field is chosen with finitely large amount of points suited for cryptographic operations. The operations on finite fields 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 characterized by the vector (x, y).

D) EC on Prime field Fp

The equation of the elliptic curve on a prime field [12] Fp is $y2 \bmod p = x3 + ax + b \bmod p$, where $4a3 + 27b2 \bmod p \neq 0$. Now the elements of the finite field are integers between 0 and $p - 1$. All the operations such as addition, substraction, division, multiplication involves integers between 0 and $p - 1$. This is known as modular arithmetic. The prime number p is preferred in such a way that there is finitely large number of points on the elliptic curve to make the cryptosystem secure. SEC gives curves with p ranging between 112-521 bits. The graph for this elliptic curve equation is not a smooth curve. Therefore the geometrical explanation of point addition and doubling as in real numbers will not work here. On the other hand, the algebraic rules for point addition and point doubling can be adapted for elliptic curves over Fp.

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 $xL = s2 - xJ - xK \bmod p$ $yL = -yJ + s(xJ - xL) \bmod p$ $s = (yJ - yK)/(xJ - xK) \bmod p$, s is the slope of the line through J and K. If $K = -J$ i.e. $K = (xJ, -J \bmod p)$ 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 $J + K = K + J$

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 $-K = (xk, -yk \bmod p)$ Point subtraction is used in certain implementation of point multiplication such as NAF [1].

3) Point Doubling

Consider a point J such that $J = (xJ, yJ)$, where $yJ \neq 0$ Let $L = 2J$ where $L = (xL, yL)$, Then $xL = s2 - 2xJ \bmod p$

$yL = -yJ + s(xJ - xL) \bmod p$

$s = (3xJ\ 2 + a) / (2yJ) \bmod p$, s is the tangent at point J and a is one of the parameters chosen with the elliptic curve If $yJ = 0$ then $2J = O$, where O is the point at infinity.

E) EC on Binary field F2 m

The equation of the elliptic curve on a binary field [11] F2 m is $y2 + xy = x3 + ax2 + b$, where $b \neq 0$. At this point, 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 polynomial arithmetic is defined in session 10.2. 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 [4]. 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 F2 m.

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 $J + K = K + J$

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 such as NAF.

3) Point Doubling

Consider a point J such that $J = (xJ, yJ)$, where $xJ \neq 0$

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

$xL = s2 + s + a$

$yL = xJ$

$2 + (s + 1)*xL$

$s = xJ + yJ/ xJ$, 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.

F) 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 [14]. The generation of domain parameters is out of scope of this paper. There are several standard domain parameters defined by SEC. Generally the protocols implementing the ECC specify the domain parameters to be used.

1) 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 y2 mod p= x3 + ax + b mod p. 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(Fp)/n. #E(Fp) is the number of points on an elliptic curve.

2) *Domain parameters for EC over field F2 m*

The domain parameters used for elliptic curve over F2 m are m, f(x), a, b, G, n and h. m is an integer intended for finite field F2 m. The elements of the finite field F2 m 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 y2 + xy = x3 + ax2 + b. G is the generator point (xG, yG), a point on the elliptic curve selected 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(F2 m)/n. #E(F2 m) is the number of points on an elliptic curve.

V. EC CRYPTOGRAPHY – (ECDH – ELLIPTIC CURVE DIFFIE HELLMAN)

ECDH [14] is a key agreement protocol that permits two parties to establish a shared secret key that can be used for private key algorithms. Mutually parties exchange some public information to each other. Using this public data and their own private data these parties computes the shared secret key. Any third party, who doesn’t have right to use to the private details of each device, will not be able to determine the shared secret from the available public information. An outline of ECDH process is defined below. For producing a shared secret between A and B using ECDH, both have to agree up on Elliptic Curve domain parameters. Both end have a key pair consisting of a private key d (a randomly selected integer less than n, where n is the order of the curve, an elliptic curve domain parameter) and a public key Q = d * G (G is the generator point, an elliptic curve domain parameter). Suppose (dA, QA) be the private key - public key pair of A and (dB, QB) be the private key - public key pair of B.

1. The end A calculates K = (xK, yK) = dA * QB
2. The end B calculates L = (xL, yL) = dB * QA
3. Since dAQB = dAdBG = dBdAG = dBQA. As a result K = L and hence xK = xL
4. Hence the shared secret is xK

Since it is practically impossible to find the private key dA or dB from the public key K or L, it’s not possible to find the shared secret for a third party.

VI. WHAT'S THE DIFFERENCE BETWEEN THE RSA AND DIFFIE-HELLMAN SCHEMES?

Diffie and Hellman proposed a system that necessitates the dynamic exchange of keys for every sender-receiver pair. This two-way key negotiation is helpful in further complicating attacks, but requires additional communications overhead. The RSA system reduces communications overhead with the capability to have static, unchanging keys for each receiver that are `advertised' by a recognized `trusted authority' (the hierarchical model) or distributed in an informal `web of trust'.

VII. COMPARING ECC TO RSA AND DIFFIE-HELLMAN

ECC's effectiveness and security is considered strong enough that the US National Security Agency (NSA) incorporated it, while excluding RSA, from its Suite B cryptography recommendations. Suite B is a set of algorithms that the NSA recommends for use in defensive both classified and unclassified US government information and systems.

One of the ways judgments are made about the correct key size for a public key system is to look at the power of the conventional (symmetric) encryption algorithms that the public key algorithm will be used to key or authenticate. The subsequent table gives the key sizes recommended by the National Institute of Standards and Technology (NIST) to shield keys used in conventional encryption algorithms like the DES and AES together with the key sizes for RSA, Diffie-Hellman and elliptic curves that are required to offer equivalent security.

Symmetric Key Size (bits)	**RSA and Diffie-Hellman Key Size (bits)**	**Elliptic Curve Key Size (bits)**
56	512	112
80	1024	160
112	2048	224
128	3072	256
192	7680	384
256	15360	521

TABLE 1: COMPARABLE KEY SIZES IN TERMS OF COMPUTATIONAL EFFORT FOR CRYPTANALYSIS

As symmetric key sizes increase, the required key sizes for RSA and Diffie-Hellman increase at a more rapidly rate than the required key sizes for elliptic curve cryptosystems. An elliptic curve system proposes more security per bit increase in key size than either RSA or Diffie-Hellman public key systems [6]. Elliptic curve cryptosystems are also more computationally efficient than the first generation public key systems, RSA and Diffie-Hellman. Even though elliptic curve arithmetic is slightly more difficult per bit than either RSA or DH arithmetic, the added strength per bit more than makes up for any additional compute time. The subsequent table shows the ratio of DH computation versus EC computation for every key size listed in Table 1.

Security Level (bits)	Ratio of DH Cost : EC Cost
80	3:1
112	6:1
128	10:1
192	32:1
256	64:1

TABLE 2: RELATIVE COMPUTATION COSTS OF DIFFIE-HELLMAN AND ELLIPTIC CURVES

Elliptic curve cryptography support is still in its early life but its use will only grow in the coming years.

Through the result of tables given below, it can be seen that ECC is superior to RSA in terms of the key size and cost. When all parameters are chosen securely, ECC cryptosystems provide the same functionality and security features as the widely used RSA cryptosystem. Still, they require much shorter key lengths for getting a similar security level. For example, a 160-bit ECC-key (meaning that the underlying field is Fp for a 160-bit prime p) attains about the same level of security as a 1024-bit RSA key. In the future, when computers become more powerful and necessary key lengths become larger, this advantage grows disproportion ally bigger. As the ECC key sizes are so much shorter than comparable RSA keys, the length of both the public key and private key is much shorter in elliptic curve cryptosystems. This results into faster processing times, and lesser demands on memory and bandwidth; some studies have found that ECC is faster than RSA for signing and decryption, but slower for signature verification and encryption [5].

ECC Key Size	RSA Key Size	Key-Size Ratio
163	1,024	1:6
256	3,072	1:12
384	7,680	1:20
512	15,360	1:30

Table 3: Key sizes in bits

Recommended key lengths for RSA, ECC-systems and symmetric algorithms until 2040 are shown in the graphs below

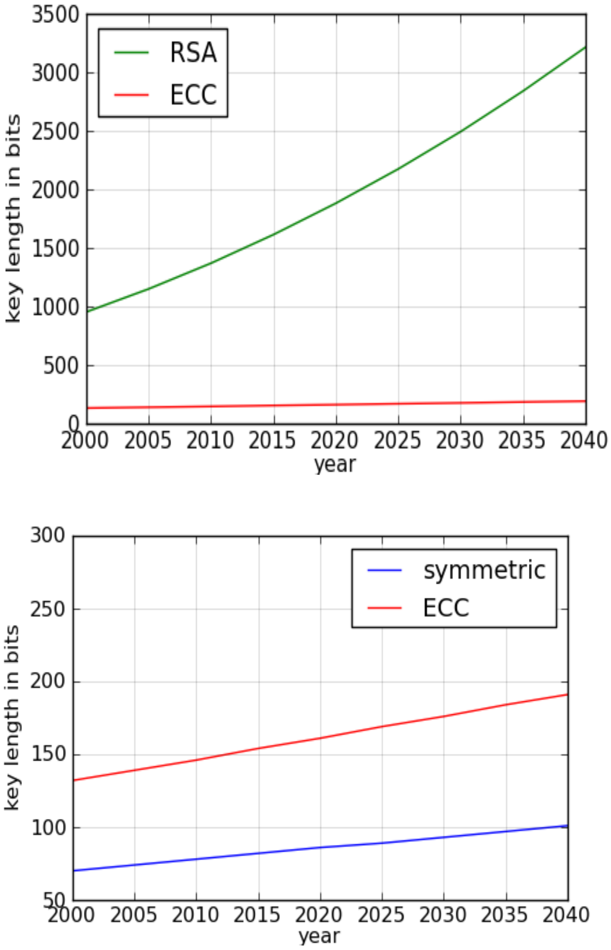

FIGURE 1: RECOMMENDED KEY LENGTHS FOR RSA, ECC-SYSTEMS AND SYMMETRIC ALGORITHMS UNTIL 2040 [17].

VIII. RECOMMENDED KEY LENGTHS

The reason for this advantage of ECC lies in the fact that the best algorithms for solving the ECDLP are exponential, while the best factorization algorithm runs in sub-exponential-time. So the parameters for a cryptosystem based on the difficulty of factoring large integers, such as RSA, must be considerably larger. These details result in substantial advantages of ECC over conventional public-key algorithms. ECC cryptosystems provide high security using comparatively short key lengths, which allows for faster computations and shorter signatures and thus results in a more efficient utilization of resources like processing power, bandwidth, storage capacity, and power consumption. This is especially applicable for smart card applications, where such resources are inadequate, as well as in high security environment.

IX. CONCLUSIONS

"Security is not the only striking feature of elliptic curve cryptography. Elliptic curve cryptosystems also are more computationally efficient than the initial generation public key systems, RSA and Diffie-Hellman". In this paper, we discussed the basis of RSA Cryptography, Diffie Hellman Key Exchange Algorithms and ECC cryptography and a comparison between them is made on the basis of Key size, security and Cost. These Algorithms that were previously explained are compared in terms of protection against common attacks, fastness in Encryption/ Decryption step and security as they are explained in only mathematical and theoretical basis; no experiment was conducted; although some experimental results were referred to.

X. REFERENCES

[1] William Stallings,"Cryptography and Network Security, Principles and Practices, Fourth Edition", Pearson Education, 2006.

[2] R.L.Rivest, A.Shamir, and L.Adelman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems", Comm., Of ACM Vol.21, no.2, pp. 122-126, 1978.

[3] Hellman, M."An Overview of Public Key Cryptography" IEEE Communication Magazine, November 1999.

[4] Anoop MS "Elliptic Curve Cryptography-An Implementation Guide", anoopms@tataelxsi.co.in.

[5] Gary C. Kessler "An Overview of Cryptography" March 2011

[6] W. Diffie and M. E. Hellmann, "*New Directions in Cryptography*," *IEEE Transactions on Information Theory*, Vol. IT-22, pp. 644-654, Nov. 1976

[7] A. J. Menezes; P. C. van Oorschot; S. A. Vanstone "Handbook of Applied Cryptography", ISBN 0-8493-8523-7, 1997. http://www.cacr.math.uwaterloo.ca/hac/.

[8] Barmawi, A.M. Takada, S. Doi, N. "Augmented encrypted key exchange using RSA encryption", pp.-490,vol No.-2, PIMRC '97., The 8th IEEE International Symposium on, August 2002.

[9] Maurer, U. Wolf, S. "Diffie-Hellman, decision Diffie-Hellman, and discrete logarithms" . pp. 327, Information Theory, 1998. Proceedings. 1998 IEEE International Symposium on ,August 2002

[10] Nan Li "Research on Diffie-Hellman key exchange protocol", pp. V4-634, Vol No.4, Computer Engineering and Technology (ICCET), 2010 2nd International Conference on , June 2010 .

[11] Darrel Hankerson, Julio Lopez Hernandez, Alfred Menezes, *Software Implementation of Elliptic Curve Cryptography over Binary Fields*, 2000, Available at http://citeseer.ist.psu.edu/hankerson00software.html

[12] M. Brown, D. Hankerson, J. Lopez, A. Menezes, *Software Implementation of the NIST Elliptic Curves Over Prime Fields*, 2001, Available at http://citeseer.ist.psu.edu/brown01software.html

[13] Certicom, Standards for Efficient Cryptography, *SEC 1: Elliptic* Curve *Cryptography,Version 1.0*, September 2000, Available at http://www.secg.org/download/aid- 385/sec1_final.pdf

[14] Certicom, Standards for Efficient Cryptography, *SEC 2: Recommended Elliptic Curve Domain Parameters*, *Version 1.0*, September 2000, Available at http://www.secg.org/download/aid-386/sec2_final.pdf

[15] Certicom, http://www.certicom.com/index.php?action=ecc_tutorial,home

[16] Alfred J. Menezes, Paul C. van Oorschot and Scott A. Vanstone, *Handbook of Applied Cryptography*, CRC Press, 1996

[17] http://www.key distribution\Intro-EC-in-Cryptography__6__Conclusion_and_Further_Topics -- Sage.mht