**1. Introduction**

In this age of universal electronic connectivity, of viruses and hackers, of electronic eavesdropping and electronic fraud, there is indeed no time at which security does not matter. This is where cryptography comes in [12].

From the ancient days of classical encryption techniques like substitution, transposition, rotor machines and steganography, the field of cryptography has evolved to the extent that nowadays cryptography considers the study and practices of authentication, digital signatures, integrity checking, and key management, etc.

A cryptographic system or a cryptosystem can be broadly classified on the basis of the number of keys used as a **symmetric key cryptosystem** and an **asymmetric key cryptosystem** (also called **public key cryptosystem**). In symmetric key cryptosystems, only one key, the same key, is used for both encryption and decryption whereas in asymmetric key cryptosystems, two different keys are used, one for encryption and another for decryption. Symmetric key cryptosystems required that key to be distributed to the communicating parties in advance i.e. there existed problem with the key distribution which was solved by using public key cryptosystems.

Public key cryptosystems are classified into three classes of mathematical problems using a trapdoor one-way function:

* Integer factorization algorithms(e.g. RSA)
* Discrete Logarithms(e.g. Diffie-Hellman)
* Elliptic Curves

The RSA cryptosystem invented by Ron Rivest, Adi Shamir and Len Adlemann in 1978 is the most widely known and widely used public key cryptosystem in the world today [8]. RSA involves a public key and a private key. The public key is used for encrypting messages and is made available to everyone while the private key is kept secret and is used for decrypting messages encrypted with the public key. RSA works on the principle that it is relatively easy to find large primes whereas it is infeasible to factorize a product of two large primes. Generally, easy is defined to mean a problem that can be solved in polynomial time as a function of input length. Thus, if the length of the input is n bits, then the time to compute the function is proportional to na, where a is a fixed constant. Such algorithms belong to the class P. A problem is infeasible if the effort to solve it grows faster than polynomial time as a function of input size. For example, if the length of the input is n bits and the time to compute the function is proportional to 2n, the problem is considered infeasible [15].

Public key cryptosystems based on discrete logarithm problem are of the form

y = gx mod p

where g is a primitive root of prime p and 0≤x≤p-1.

Given g, x, and p, it is a straightforward matter to calculate y while given y, g, and p, it is, infeasible to calculate x for large primes, i.e., it is relatively easy to calculate exponentials modulo a prime; it is very difficult to calculate discrete logarithms. For large primes, the latter task is considered infeasible [12].

Both of these approaches require the use of large primes in order to use public key cryptography effectively which demands considerable computing resources. However, on limited platforms, such as handheld devices, the problem is exacerbated and one may ask if it is possible at all to implement public-key cryptography efficiently on handheld devices, i.e. where the time needed to perform encryption or decryption is sufficiently small to avoid having a negative impact on the user experience while retaining the security of the cryptosystem [1].

Another approach to public key cryptography is elliptic curve cryptography which is based on elliptic curve discrete logarithm problem. The elliptic curve approach uses small keys in comparison to RSA approach of large keys providing equal level of security from cryptanalysis point of view [14].

**2. Problem Definition**

The original RSA cryptosystem was proposed in 1978 by Rivest, Shamir and Adelman [1] and consists of three parts:

• **Key generation:** Given an integer n, generate two distinct primes p and q of (n/2)-bits each and compute N = p\*q and φ(N) = (p-1)\*(q-1). Choose a random integer 1 < e < φ(N) such that gcd(e,φ(N)) = 1. Next, compute the uniquely defined integer 1 < d < φ(N) satisfying e\*d ≡ 1 (mod φ(N)). The public key is <N,e> and the private key <N,d>.

• **Encryption:** To encrypt a message X with the public key <N,e>, transform the message X to an integer M in {0,…,N-1} and compute the ciphertext C = Me mod N.

• **Decryption:** To decrypt the ciphertext C with the private key <N,d>, compute M = Cd mod N and employ the reverse transformation to obtain the message X from M.

Key generation is only performed occasionally so the efficiency of that part is less important than the two other parts, encryption and decryption. Their efficiency is determined by:

1) the transformation of the message X to the integer M, and back, and

2) the modular exponentiations Me mod N and Cd mod N.

The transformations can be performed with standard algorithms, e.g. from the Public Key Cryptography Standards (PKCS) published by RSA Security [3]. Thus modular exponentiation is the single most important component of RSA in regards to its efficiency.

Practical approach to perform modular exponentiation is the repeated square-and-multiply algorithm [6]. In this case, we have large N so as to counter the factorization attacks till known for the RSA modulus [13]. The public key exponent e is small and generally chosen to be e = 216 + 1, so that encryption is efficient. The problem lies in decryption since private key exponent d cannot be chosen that way and in the worst case |d| nearly has the same magnitude as |N|. According to repeated square-and- multiply algorithm its runtime complexity is O(tv2) where t is the bitlength of the exponent and v is the bitlength of the modulus thus yielding an O(n3) complexity in case of decryption in RSA[10].

Thus the problem with RSA is in optimizing its decryption time so that RSA can be implemented successfully in devices with limited computing resources such as mobile phones, PDAs etc. Taking these considerations into account some variants of RSA like CRT RSA, Multi-prime RSA, Multi-power RSA, Rebalanced RSA and R-prime RSA are analyzed in this research so as to reduce the decryption time for limited platforms and a new approach to public key cryptography which is elliptic curve cryptography (ECC) is also taken for research so as to implement it on a handheld device and compare its efficiency with RSA variants.

**3. Objectives**

To perform comparative analysis of RSA variants and ECC on the basis of encryption/decryption time speedups gained so as to achieve the best public key cryptographic model for handheld devices.

**4. Research Methodology and Literature Review**

1. **Literature Review**

A lot of research work has been done in optimizing the decryption time of RSA with its variants but still elliptic curve cryptography is just an emerging cryptosystem and very few research works have been done on it especially if we take the case of handheld devices.

**4.1.1. Fast Variants of RSA**

**4.1.1.1 CRT RSA**

CRT RSA is the one of the RSA variant for speeding up decryption. The idea behind CRT RSA is to split the costly decryption into two smaller and faster modular exponentiations using the Chinese Remainder Theorem, hence the acronym CRT RSA. It was first described by Couvreur and Quisquater in 1982 [1].

According to the Chinese Remainder Theorem, for a system of r Congruences,

x ≡ a1 (mod n1), …, x ≡ ar (mod nr), where n1,…,nr are relatively prime integers and a1,…,ar

are ordinary integers has a unique solution modulo N = n1\*n2\*…\*nr.

This solution can be written as

x = (a1\*N1\*y1 + … + ar\*Nr\*yr) mod N,

where Ni = N/ni and yi = Ni-1 mod ni for 1 ≤ i ≤ r [4].

CRT RSA uses the Chinese Remainder Theorem the following way:

**• Key generation:** Generate e and d the same way as in the original RSA. Next, compute dp = d mod p-1 and dq = d mod q-1. The public key is <N,e> and the private key <p,q,dp,dq>.

**• Encryption:** Encryption is the same as for the original RSA, C = Me mod N.

**• Decryption:** Decryption is split into the following computations:

First, compute Mp = Cdp mod p and Mq = Cdq mod q.

Then, using the Chinese Remainder Theorem, find M as:

M = (Mp\*q\*(q-1 mod p) + Mq\*p\*(p-1 mod q)) mod N.

If we ignore the contribution from the sum function of the Chinese Remainder Theorem, decryption using CRT RSA requires two times O((n/2)3) since the bitlength of both the exponents and the moduli are n/2 and a single modular exponentiation has an upper bound of O(tv2) where t is the bitlength of the exponent and v is the bitlength of the modulus. Compared to the O(n3) decryption of the original RSA, CRT RSA improves decryption time with a factor

n3 / (2 \* (n/2)3) = 42.

**4.1.1.2 Multi-Prime RSA**

By adding more primes to the generation of N, decryption can be split into an arbitrary number of smaller exponentiations instead of just two. This is the theme of Multi-Prime RSA. This variant of the original RSA was first described by Collins et al. in 1997 [1]:

• **Key generation:** Given two integers n and r ≥ 3, generate r different primes p1,…, pr each (n/r)-bits long. Set N = p1\*p2\*…\*pr and φ(N) = (p1-1)\*(p2-1)\*…\*(pr-1).

Compute e and d as in the original RSA.

Next, compute di = d mod pi-1 for 1 ≤ i ≤ r. The public key is <N,e> and the private key is <p1,…,pr,d1,…,dr>.

• **Encryption:** Encryption is the same as for the original RSA, C = Me mod N.

• **Decryption:** Decryption is split into r exponentiations, Mi = Cdi mod pi, for 1 ≤ i ≤ r. Using the Chinese Remainder Theorem, M is found as

M = (M1\*N1\*y1 + …+ Mr\*Nr\*yr) mod N where Ni = N/pi and yi = Ni-1 mod pi for 1 ≤ i ≤ r.

If we ignore the contribution of the Chinese Remainder Theorem, decryption in Multi-Prime RSA requires r times O((n/r)3). Compared to the O(n3) decryption of the

original RSA, Multi-Prime RSA improves decryption time with a factor

n3 / (r \* (n/r)3) = r2.

Since the individual primes need to have a certain size to guard against factorization attacks, so the size of r is limited and thus the actual improvement in decryption is also checked. Hinek provided the following guidelines [8]:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| n | 1024 | 2048 | 4096 | 8192 |
| max r | 3 | 3 | 4 | 4 |

**4.1.1.3 Multi-Power RSA**

In Multi-Prime RSA, the modulus N consists of r different primes whereas the modulus in Multi-Power RSA has the structure N = pr-1q for r ≥ 3. This different structured modulus gives rise to a more efficient decryption than Multi-Prime RSA. Multi-Power RSA was first described by Takagi in 1998 [5]:

• **Key generation:** Given two integers n and r ≥ 3, generate two different primes p and q each (n/r)-bits long. Set N = pr-1q. Compute e as in the original RSA and then compute d satisfying e\*d ≡ 1 (mod (p-1)\*(q-1)). Finally, compute dp = d mod p-1 and dq = d mod q-1. The public key is <N,e> and the private key <p,q,dp,dq>.

• **Encryption:** Encryption is the same as for the original RSA, C = Me mod N.

• **Decryption:** To decrypt the ciphertext using the private key <p,q,dp,dq>[7]:

Step 1: Compute Mp = Cdp mod p and Mq = Cdq mod q. Thus Mpe = C mod p and Mqe = C mod q.

Step 2: Using Hensel Lifting construct an Mp1 such that (Mp1)e = C mod pr-1.

Step 3: Using CRT, compute a M ε ZN such that M = Mp1 mod pb – 1 and M = Mq mod q. Then M = Cd mod N is a proper decryption of C.

Ignoring the contribution from the Chinese Remainder Theorem and the extra arithmetic from the Hensel Lifting, this means that decryption in Multi-Power RSA requires two times O((n/r)3). Compared to the O(n3) decryption of the original RSA, Multi-Power RSA improves decryption time with a factor

n3 / (2 \* (n/r)3) = r3/2.

With respect to the security of Multi-Power RSA, the same guidelines as for Multi-Prime RSA apply limiting the size of r and thus the actual improvement in practice.

**4.1.1.4 Rebalanced RSA**

In the original RSA, encryption is more efficient than decryption because e is small and d is large. So a straightforward way to optimize decryption is to “switch” the exponents, i.e. make e large and d small. But small values of d open up for Wiener's Low Decryption Exponent Attack [9]. Instead Wiener proposed in 1990 the variant Rebalanced RSA [2] that retains the size of d but makes dp = d mod p-1 and dq = d mod q-1 small (at the expense of a larger e):

**• Key generation:** Given integers n and w, generate two different primes p and q each (n/2)-bits long such that gcd(p-1,q-1) = 2. Set N = p\*q and φ(N) = (p-1)\*(q-1).

Compute two w-bit integers dp and dq satisfying gcd(dp,p-1) = gcd(dq,q-1) = 1 and dp ≡ dq (mod 2). Find a d such that d = dp mod p-1 and d = dq mod q-1. Compute e = d-1 mod φ(N). The public key is <N,e> and the private key is <p,q,dp,dq>.

**• Encryption:** Encryption is the same as for the original RSA, C = Me mod N, but with a much larger e (on the order of N).

**• Decryption:** Decryption is the same as for CRT RSA but with smaller dp and dq – each w-bit long in Rebalanced RSA versus (n/2)-bit in CRT RSA. Usually, w ≥ 160 and n/2 ≥ 512 [11]. Ignoring the contribution of the Chinese Remainder Theorem, decryption in Rebalanced RSA requires two times O(w(n/2)2). Compared to the O(n3) decryption of the

original RSA, Rebalanced RSA improves decryption time with a factor

n3 / (2w · (n/2)2) = 2n/w.

With respect to the security of Rebalanced RSA, it is recommended to set w ≥ 160 thereby limiting the actual improvement of decryption in practice [11]. Also the speed-up in decryption comes at the cost of a much slower encryption (since e is on the order of N). This means that encryption in Rebalanced RSA is as slow as decryption in the original RSA.

An example of this scenario can be provided by applications running on handheld devices (PDAs), which generally possess limited computational resources. In communications with servers (or even with notebooks or desktop computers), we can

leave the task of decryption (fast) for the small device, and the encryption(slow) for the computers with more computational resources.

**4.1.1.5 R-Prime RSA**

In Rebalanced RSA, decryption is the same as in CRT RSA. Since Multi-Prime RSA is a generalization of CRT RSA, why not generalize Rebalanced RSA to use Multi-Prime RSA

in its decryption as well. This is the idea behind R-Prime RSA. It was first described by Paixao in 2003 [2]:

**• Key generation:** Given n and w, generate r ≥ 3 different primes p1,…,pr each (n/r)-bits long such that gcd(p1-1,…,pr-1) = 2. Set N = p1\*…\*pr and φ(N) = (p1-1)\*…\*(pr-1). Compute r w-bit integers dp1,…, dpr satisfying gcd(dp1,p1-1) = … = gcd(dpr,pr-1) = 1 and dp1 ≡ … ≡ dpr (mod 2). Find a d such that d = dp1 mod p1-1,…, d = dpr mod pr-1. Compute e = d-1 mod φ(N). The public key is <N,e> and the private key is <p1,…,pr,dp1,…,dpr>.

**• Encryption:** Encryption is the same as for the original RSA, C = Me mod N, but with a much larger e (as was the case with Rebalanced RSA).

**• Decryption:** Decryption is the same as for Multi-Prime RSA, i.e., decryption is split into r modular exponentiations Mi = Cdpi mod pi for 1 ≤ i ≤ r after which the Chinese Remainder Theorem is applied. The difference lies in the length of dpi (denoted di in Multi-

Prime RSA). In R-Prime RSA, these values are w-bit each whereas in Multi-Prime RSA, they are n/r each.

This means that decryption in R-Prime RSA requires r times O(w(n/r)2) (ignoring the Chinese Remainder Theorem). Compared to the O(n3) decryption of the original RSA, R-Prime RSA improves decryption time with a factor

n3 / (r\*w · (n/r)2) = nr/w.

The same security considerations as Rebalanced RSA and Multi-Prime RSA apply to R-Prime RSA, i.e. it is recommended to set w ≥ 160 and limit the value of r with respect to n. As in Rebalanced RSA, the speed-up in decryption means a much slower encryption. Since Multi-Power RSA is faster than Multi-Prime RSA, why not use Multi-Power RSA in R-Prime RSA? The reason is the technique Hensel Lifting that is used in decryption in Multi-Power RSA. Hensel Lifting makes use of a number of exponentiations modulo e which is not a problem in Multi-Power RSA since e is small. But in R-Prime RSA, e is on the order of N making Hensel Lifting (and consequently, Multi-Power RSA) inefficient.

**4.1.2. Elliptic Curve Cryptography**

Let GF(q) be a finite field with q elements, where q = pm. The number p is prime and is called the characteristic of GF(q). If m = 1, we have GF(q) = Zp, the set of integers modulo p.

An elliptic curve over Zp for prime p is defined by the cubic equation

y2 ≡ (x3 + ax + b) mod p

where coefficients a and b and the variables x and y are all elements of Zp such that 4a3 + 27b2 ≠ 0.

The set of integers (x,y) satisfying the above equation together with a point at infinity O forms a set Ep(a,b). Based on the set Ep(a,b) we can define a finite abelian group as follows:

For all points P,Q ε Ep(a,b) with O serving as the identity element:

(1) Identity: P + O = P

(2) Negative: If P = (xP,yP) then –P = (xP,-yP) and P + (-P) = O

(3) Point Addition: If P = (xP,yP) and Q = (xQ,yQ) with P ≠ -Q, then R = P + Q = (xR,yR) is determined by following rules:

xR = (λ2 – xP – xQ) mod p

yR = (λ(xP - xR) - yP) mod p

where λ = ( (yQ - yP) / (xQ - xP) ) mod p if P ≠ Q

and λ = ( (3xP2 + a) / 2yP ) mod p if P = Q

(4) Point Multiplication: Multiplication is defined as repeated addition; e.g. nP = P + P + …+ P (n times) [13, 14]

Using these facts we will find a base point G (or a generator point if the order of the group is prime) on the elliptic curve whose order is a large value n. The order n of point G on the elliptic curve is the smallest positive integer n such that nG = O and the private keys to be used by the communicating parties are integers less than n and their respective public keys are G multiplied by private keys.

To perform encryption, we encode the plaintext message m to be sent as an (x,y) point Pm and it is this point that will be encrypted as ciphertext and subsequently decrypted.

Suppose that user A selects a private key nA and generates a public key PA = nA \* G.

To encrypt and send a message Pm to B, A chooses a random positive integer k and produces the ciphertext Cm consisting of the pair of points:

Cm = {kG, Pm + kPB}

Note that A has used B's public key PB. To decrypt the ciphertext, B multiplies the first point in the pair by B's secret key and subtracts the result from the second point:

Pm + kPB - nB(kG) = Pm + k(nBG) - nB(kG) = Pm

A has masked the message Pm by adding kPB to it. Nobody but A knows the value of k, so even though PB is a public key, nobody can remove the mask kPB. However, A also includes a "clue," which is enough to remove the mask if one knows the private key nB. For an attacker to recover the message, the attacker would have to compute k given G and kG, which is a hard problem.

The elliptic curve approach uses advantage of easiness of computing Q given k and P in the equation Q = kP where Q,P ε Ep(a,b) and k<p while hardness of computing k given Q and P. This is called **elliptic curve discrete logarithm problem** (ECDLP).

Currently the best algorithms known to solve the ECDLP have fully exponential running time, in contrast to the subexponential-time algorithms known for the integer factorization problem. This means that a desired security level can be attained with significantly smaller keys in elliptic curve systems than is possible with their RSA counterparts. The advantages that can be gained from smaller key sizes include speed and efficient use of computing power, bandwidth, and storage.[14]

1. **Data Collection**

Suitable sample programs that implements above mentioned RSA variants and ECC will be run and numeric data will be taken for encryption and decryption.

1. **Testing and Verification**

The time taken by RSA variants and ECC for encryption and decryption will be evaluated. For testing the simulation environment will be J2ME. The time complexity of each algorithm derived from theoretical point of view will be compared with time complexity given by experiment for verification.

**5. Expected Output**

After the performance comparison of RSA variants and ECC in terms of encryption/decryption time, the best public key cryptographic model will be the final output of this research.

**6. Working Schedule**

|  |  |  |
| --- | --- | --- |
| **S.N.** | **Activities** | **Duration (Days)** |
| **1** | **Study of Papers** | **30\*** |
| **2** | **Topic Selection** | **10\*** |
| **3** | **Generation of Conceptual Framework** | **15\*** |
| **4** | **Implementation** | **5** |
| **5** | **Proposal Writing** | **5\*** |
| **6** | **Data Collection and Analysis** | **3** |
| **7** | **Thesis Documentation** | **10** |
| **8** | **Final Presentation** | **1** |

**\***The tasks are completed.

**7. References**

[1] K. Hansen, T. Larsen and K. Olsen. “On the Efficiency of Fast RSA Variants in Modern

Mobile Phones.” International Journal of Computer Science and Information Security,

Vol. 6, No. 3, 2009.

[2] C. A. M. Paixao. “An Efficient Variant of the RSA Cryptosystem.” University of Sao Paulo, Brasil, 2003.

[3] RSA Laboratories. “PKCS #1 v2.1: RSA Cryptography Standard.” RSA laboratories, 2002.

[4] http://data.at.preempted.net/INDEX/articles/CRT.pdf

[5] T. Takagi. “Fast RSA-type Cryptosystem Modulo pkq.” Proceedings of Crypto 98, Vol. 1462, pp. 318-326, 1998.

[6] Md. Ali-Al-Mamun, Md. M. Islam, S.M. M. Romman and A.H. S. U. Ahmad. “Performance Evaluation of Several Efficient RSA Variants.” International Journal of Computer Science and Network Security, Vol.8, No.7, July 2008.

[7] Z. Zhong and Z. Xia. “On the Variants and Speed Methods of RSA.” Department of Computer Science, School of Computer, Wuhan University, P.R. China.

[8] M. J. Hinek. “On the Security of Multi-Prime RSA.” University of Waterloo, Canada, 2006.

[9] M. J. Wiener. “Cryptanalysis of Short RSA Secret Exponents.” Aug. 3, 1989.

[10] A. Menezes, P. van Oorschot and S. Vanstone. “Handbook of Applied Cryptography.” 1st Edition, CRC Press, 1997.

[11] D. Boneh and H. Shacham. “Fast Variants of RSA.” RSA Laboratories, 2002.

[12] W. Stallings. “Cryptography and Network Security” Fourth Edition, 2009.

[13] H.C.A. van Tilborg. “Fundamentals of Cryptology”.

[14] R. Soram and M. Khomdram. “Juxtaposition of RSA and Elliptic Curve Cryptosystem.” International Journal of Computer Science and Network Security, Vol.9 No.9, September 2009.

[15] T.H. Cormen, C.E. Leiserson, R.L. Rivest, C. Stein. “Introduction to Algorithms.” Second Edition, Prentice-Hall India, 2007.