A SOFTWARE COMPARISON OF RSA AND ECC

Vivek B. Kute
Lecturer. CSE Department, SVPCET, Nagpur 9975549138
vivek_kute@rediffmail.com

P. R. Paradhi
Lecturer, CSE Department, RKNEC, Nagpur 9823049376
praful_pardhi@rediffmail.com

G. R. Bamnote
Sr. Lecturer, CSE Department, PRMIT, Badnera, 9421741516
grbamnote@rediffmail.com

ABSTRACT

To meet a user's needs cryptographic algorithm needs to be selected on the basis of attributes like security and performance. One of the tasks of a cryptosystem designer is to weigh the advantages and disadvantages and select the algorithmic tools that best address the problem to be solved.

RSA is the most popular public-key cryptosystem today but long-term trends such as the proliferation of smaller, simpler devices and increasing security needs will make continued reliance on RSA more challenging over time. Hence Elliptic Curve Cryptography (ECC) is a suitable alternative.

This paper focuses on performance attribute of public key cryptosystems. The algorithms studied and compared are RSA, ECC. We have implemented these algorithms in Java in order to perform software tests so that we may gain insight into the relative performance of each algorithm and its associative parameters.

Software based tests are performed to yield an overall analysis of key generation, message encryption and decryption. Implementations are in Java and executable in the Windows environment. Each algorithm is tested for key generation and encryption/decryption of ordinary but large files.

Categories and Subject Descriptors

Network Security

General Terms

Algorithms, Security.

Keywords

Cryptographic Algorithms, Elliptic curve cryptography, RSA, Encryption / Decryption, Public Key Cryptography.

1 INTRODUCTION

Public key cryptography is based on the creation of mathematical puzzles that are difficult to solve without certain knowledge about how they were created. The creator keeps that knowledge secret (the private key) and publishes the puzzle (the public key).

A Class of puzzle involves solving the equation $a^b = c$ for b when a and c are known. Such equations involving real or complex numbers are easily solved using logarithms. However, in a large finite group, finding solutions to such equations is quite difficult and is known as the discrete logarithm problem.[1,2,5,8,11]

This paper focuses on implementation aspects of 2 public key cryptography algorithms: RSA and ECC and evaluate the algorithms with respect to time required for key generation , encryption and decryption.

Ron Rivest, Adi Shamir and Leon Adleman invented the RSA system in 1977. It is based on the assumption that it is easy to multiply two prime numbers, but difficult to divide the result again into the two prime numbers.

Neil Koblitz invented elliptic curve cryptography in 1987.Elliptic curve cryptography works with points on a curve. The security of this type of public key cryptography depends on the elliptic curve discrete logarithm problem. The main advantage of elliptic curve cryptography is that the keys can be much smaller. Recommended key sizes are in the order of 160 bits rather than 1024 bits for RSA.[1,11]

2 PUBLIC KEY ALGORITHMS

In this section we discus issues about the two public key algorithms RSA and ECC.

2.1 A Public Key Algorithm: RSA

In cryptology, **RSA** is an algorithm for public-key encryption. It was the first algorithm known to be suitable for signing as well as encryption, and one of the first great advances in public key cryptography. RSA is still widely used in electronic commerce protocols, and is believed to be secure given sufficiently long keys and the use of up-to-date implementations.

Operation: RSA involves a public and private key. The public key can be known to everyone and is used for encrypting messages. Messages encrypted with the public key can only be decrypted using the private key. The **public key** consists of the modulus n and the public (or encryption) exponent e. The **private key** consists of the modulus n and the private (or decryption) exponent d which must be kept secret.

Encrypting messages: Alice transmits her public key (n & e) to Bob and keeps the private key secret. Bob then wishes to send message **M** to Alice. He first turns **M** into a number m < n by using an agreed-upon reversible protocol known as a padding scheme. He then computes the ciphertext c corresponding to:

$$c = m^e \bmod n_c$$

This can be done quickly using the method of exponentiation by squaring. Bob then transmits c to Alice.

Decrypting messages:Alice can recover m from c by using her private key d in the following procedure**:**

$$m = c^d \bmod n$$

Given m, she can recover the original message **M**.

The decryption procedure works because first

$$c^d \equiv (m^e)^d \equiv m^{ed} \pmod{n}.$$

2.2 A public Key Algorithm: ECC

Elliptic curve cryptography (ECC) is an approach to public-key cryptography based on the algebraic structure of elliptic curves over finite fields. Elliptic curves are also used in several integer factorization algorithms that have applications in cryptography, such as, for instance, Lenstra elliptic curve factorization, but this use of elliptic curves is *not* usually referred to as "elliptic curve cryptography."

Public key cryptography is based on the creation of mathematical puzzles that are difficult to solve without certain knowledge about how they were created. The creator keeps that knowledge secret (the private key) and publishes the puzzle (the public key). The puzzle can then be used to scramble a message in a way that only the creator can unscramble. Early public key systems, such as the RSA algorithm, used products of two large prime numbers as the puzzle: a user picks two large random primes as her private key, and publishes their product as her public key. The difficulty of factoring ensures that no one else can derive the private key (i.e., the two prime factors) from the public one. However, due to recent progress in factoring, RSA public keys must now be thousands of bits long to provide adequate security.

Another class of puzzle involves solving the equation $a^b = c$ for b when a and c are known. Such equations involving real or complex numbers are easily solved using logarithms. However, in a large finite group, finding solutions to such equations is quite difficult and is known as the discrete logarithm problem. [1,2,5,8,11]

An *elliptic curve* is a plane curve defined by an equation of the form

$$y^2 = x^3 + a\,x + b.$$

The set of points on such a curve (i.e., all solutions of the equation together with a point at infinity) can be shown to form an abelian group (with the point at infinity as identity element). If the coordinates x and y are chosen from a large finite field, the solutions form a finite abelian group. The discrete logarithm problem on such elliptic curve groups is believed to be more difficult than the corresponding problem in (the multiplicative group of nonzero elements of) the underlying finite field. Thus keys in elliptic curve cryptography can be chosen to be much shorter for a comparable level of security.

3 IMPLEMENTATION

Its not difficult to find a cryptographic algorithms. Varity of options are available. The developer will have to analyze the algorithm for its suitability to his system. For this he has to implement cryptographic algorithms and select the efficient one. So in this project we have implemented the standard cryptographic algorithms and compared them.

The algorithms we have implemented are RSA and ECC. Software based tests are performed to yield an overall analysis of key generation, message encryption and decryption. Implementations are in Java and executable in the Windows environment. Each algorithm is tested for performing the following functions using ordinary but large files.

- Generation of private and public keys
- Encryption using public/private keys
- Decryption using public/private keys

3.1 The Algorithm RSA

- Each entity creates an RSA public Key and corresponding private Key.
- Each entity has to select two large prime numbers and will create public key and private key. The selected two prime numbers should be large and of same size.
- Then we will calculate their product and choose a random number. This random number and the product will be the public key.
- Compute a unique number using Extended Euclidean Algorithm. This number and the random number should be congruent to modulo product of prime numbers. This unique number will be the private key.

Now public key and private keys are generated. Each entity will publish the public key to all and will keep the private key secret. If any entity A wants to send message to entity B, then A have to use B's public key to encrypt the message and send the encrypted message to B. B in turn will use its private key to decrypt the encrypted message. Any entity wanting to send something to other entity it have to encrypt the message using other entities public key which the receiver entity can only decrypt using its private key.

3.1.1 PROCESS OF ENCRYPTION AND DECRYPTION

- The message will be some text or image data. We need to find its binary representation and divide the message into blocks so that the binary digits should be in the range from 0 to the product of prime numbers minus 1.Now each such binary block of message will be considered m and the entity B will apply the encryption algorithm on it. So each such block will be encrypted and a file of encrypted message will be created which will be sent to entity A.
- Entity A takes the encrypted file finds its binary representation and divides it into blocks. It applies the decryption algorithm on each block. Ultimately it creates the decrypted file, which will be nothing but the actual message.

3.2 The Algorithm ECC

- At first we will take a curve in the form Y^2=X^3+aX+b. where a and b are curve parameters
- We then choose a prime number.
- Using point adding and point doubling we compute the points on the curve.
- Select a generating point out of those points whose order should be large.
- Then take a random number less than order of generating point as a private number for each entity. This will be a secret key.
- This entity will then generate its public key by multiplying the generating number with the secret number and will publish the point**.**

3.2.1 PROCESS OF ENCRYPTION AND DECRYPTION

The first task in this system is to encode the plaintext message m to be sent as an x-y point Pm. It will be the point Pm that will be encrypted as a cipher text and subsequently decrypted. As with the key exchange system, an encryption/decryption system requires a point G and an elliptic group Ep(a,b) as parameters.

- Each user A selects a private key nA and generates a public key PA= nA X G.
- To encrypt and send a message Pm to B , A chooses a random positive integer x and prduces the cipher text Cm consisting to the pair of points
- Cm = {xG, Pm + xPB}. (A uses B's public key PB to encrypt the message.)
- To decrypt the cipher text , B multiplies the first point in the pair by B's secret key and subtracts the result from the second point :

$$Pm + xPB – nB(xG) = Pm + x(nBG) – nB(xG) = Pm$$

A has masked the message Pm by adding xPB to it. Nobody but A knows the value of x , so even though PB is a public key, nobody can remove the mask xPB. 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 have to compute x given G and xG which is hard.

4 IMPLEMENTATION RESULTS

We have tested the implementation of RSA and ECC separately. The tests are done for comparing the time required for key generation, encryption and decryption in both algorithms. The results are shown by following tables and graphs.

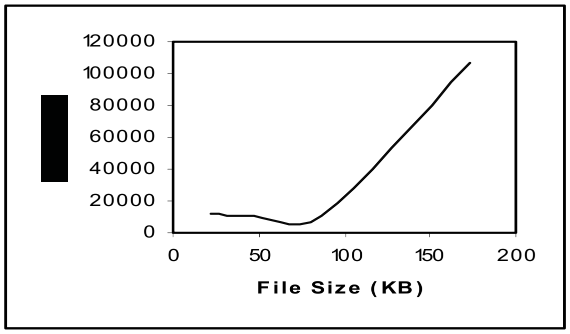
Tables and graphs for RSA :

Table. 1. Key generation time

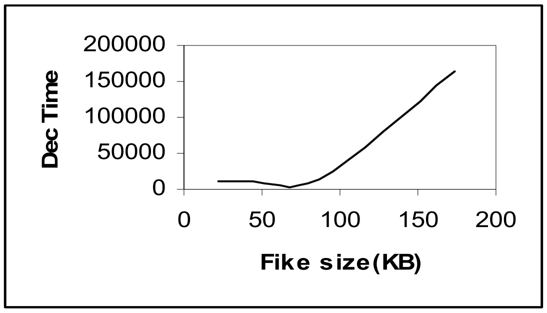
Prime	Time (in Milliseconds)	
	Public Key generation	Private Key Generation
8	47	16
16	31	15

Table 2.Enc/Dec using 8 bit Prime Number

File Size (in KB)	Time (in Milliseconds)	
	Encryptionn	Decryption
22	11907	12438
87	10828	14219
174	107062	163600



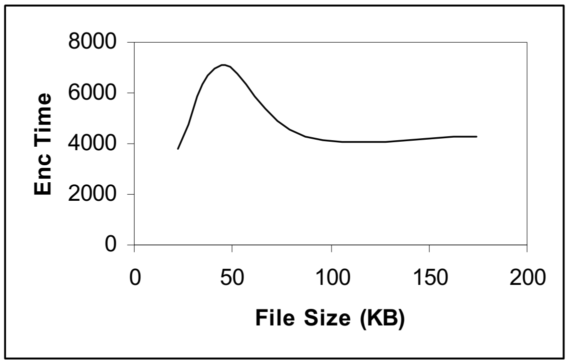
Enc time vs File size



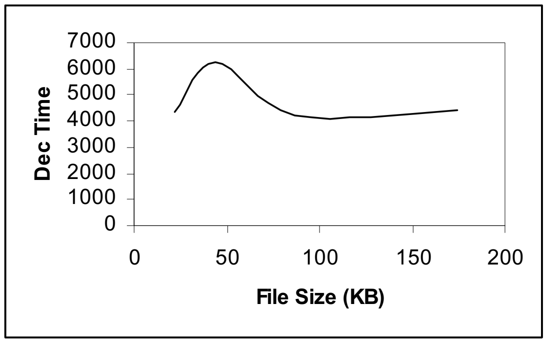
Dec time vs File size

Table 3.Enc/Dec using 16 bit Prime Number

File Size (in KB)	Time (in Milliseconds)	
	Encryption	Decryption
22	3782	4328
87	4297	4188
174	4265	4390



Enc time vs File size



Dec time vs File size

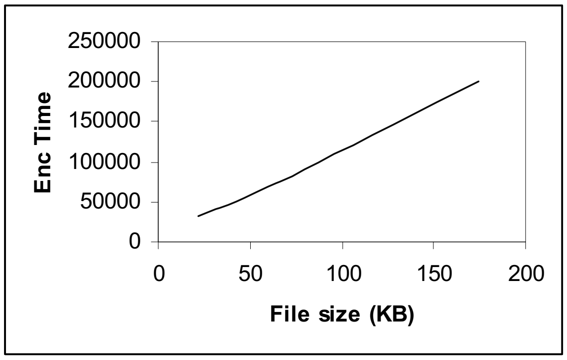
Tables and graphs for ECC :

Table. 4. Key generation time

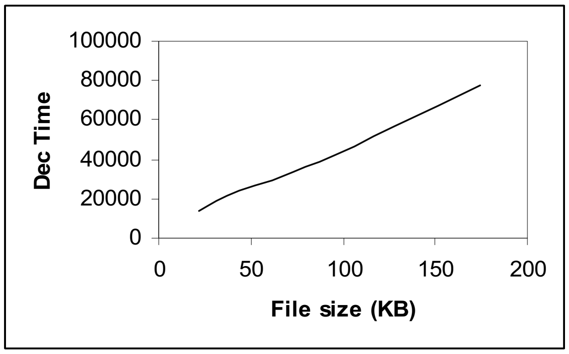
Prime Number (size in bits)	Time (in Milliseconds)		
	Curve Point Generation	A's key Generation	A's key Generation
8	79		
16	3378766	123203	122922

Table 5.Enc/Dec using 8 bit Prime Number

File Size (in KB)	Time (in Milliseconds)	
	Encryption	Decryption
22	32937	13406
87	99922	38734
174	199484	77250



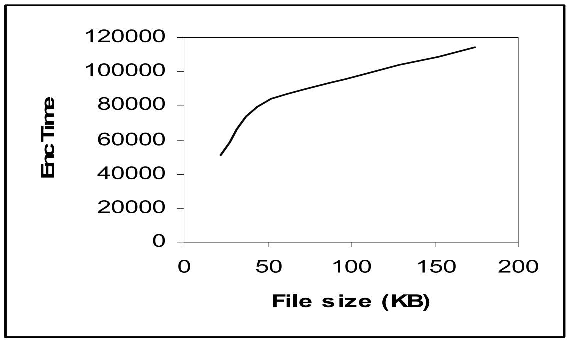
Enc time vs File size



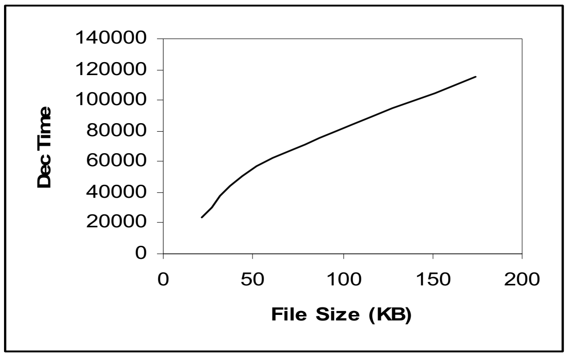
Dec time vs File size

Table 6.Enc/Dec using 16 bit Prime Number

File Size (in KB)	Time (in Milliseconds)	
	Encryption	Decryption
22	51391	24187
87	93741	75402
174	113947	115137



Enc time vs File size



Dec time vs File size

From above tables we reach to the following conclusion

Table 7 Concluding Table

	RSA	**ECC**
Key generation	FAST	LESS FAST
Encryption	VERY FAST	LESS FAST
Decryption	LESS FAST	FAST

5 CONCLUSIONS

In today's digital world there is a tremendous growth in the usage of the Internet. A day will come when almost all the work can be made possible with Internet. Behind one software generator there are hundreds of hackers. So, a fraction of second will be enough to destroy the world Internet. Hence we require strong algorithms, which secure the Internet works.

From the implementation of RSA and ECC algorithms we now can conclude that operations in RSA are comparatively faster than ECC. In RSA key generation and encryption are faster whereas decryption is slower. On the other hand in ECC key generation and encryption are slower whereas the decryption is faster.

From this conclusion RSA is faster but it is said that security wise ECC is stronger than RSA.

6 REFERENCES

[1] Menzer, Alfred et al , Handbook of Applied Cryptography(2/e ,CRC press 1996).

[2] William Stallings, Cryptography and Network Security(4/e, Pearson Education).

[3] Bruce Schneier, Applied Cryptograph y(2/e, John Willey and Sons, INC.,1996).

[4] William Stallings, Network Security Essentials, Applications and Standards (2/e , Pearson Education)

[5] Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone, Handbook of Applied Cryptography

[6] Koblitz, N. (1987), Elliptic Curve Cryptosystems, Mathematics of Computation, (Vol. 48)

[7] Stallings W. (1999), Cryptography and Network Security: Principles and Practice (2nd ed., Prentice)

[8] Atul Kahate, Cryptography and Network Security (Tata McGraw-Hill)

[9] Behrouz A Forouuzan , Data Communication and Networking (4/e, Tata McGraw-Hill)

[10] Kostas zotos, Andreas litke, Cryptography and Encryption (department of applied mathematics University of Macedonia)

[11] http://www.iusmentis.com/technology/encryption/elliptic-curves/#Introduction,"Introduction"

[12] http://en.wikipedia.org/wiki/Cryptography,"History of Cryptography"

[13] Phil Zimmermann , Introduction to Cryptography chapter 1 "Basics of Cryptography"

[14] http://en.wikipedia.org/wiki/Elliptic_curve_cryptography

[15] http://en.wikipedia.org/wiki/RSA

[16] Robert Zuccherato , Elliptic Curve Cryptography Support in Entrust (v-1,May 9 2000)

[17] High-Speed RSA Implementation. Technical report, RSA Laboratories TR201, November 1994.

[18] Lenka and Jozef, Practical Cryptogrphy-The Key Size Problem :PGP after years 21 Dec 2001

[19] RSA Scheme With MRF And ECC For Data Encryption Chaur-Chin Chen

[20] http://www.certicom.com/index.php?action=ecc,