## Implementation of Enhanced RSA Algorithm

Group15

Srinivasa Raghava Kumar Gangishetty

Alankritha Mudiraj Barla

Prudhvi Sudhana

Veerendra Kumar Pardhu Balisetty

***Abstract***

The intent of the paper is to increase the efficiency of RSA decryption by reducing the exponent in the modular exponentiation, whilst not disrupting the security. RSA (Rivest – Shamir – Adelman) being one of the most widely used public key systems computes modular exponentiation that slows down the process especially for decrypting the cipher text, more computation capacity and time is consumed.

***How are we planning on achieving this?***

The speeding up of RSA could be studied in two variants

-By focusing better on the encryption such that the exponents of the decryption become smaller (RSA – S2 System)

-By reducing the size of the moduli

***RSA – S2 System***

It involves assuming the encryption logic as server and decryption as weak client so that the computational burden (lessen the moduli and exponents of the decipher text) is shifted from decryption to encryption yielding to encrypt assistant RSA **(EARSA)** .EARSA involves key generation, encryption and decryption. This method improves the efficiency by solving large number of exponentiations with reduced moduli and external exponents.

***Security and Parameter Selection***

The choice of primes selected has major impact in security provision, this is because the obtained key generators make sure that it gets exhaustive for the hacker to encode the search space. These variance results in improved efficiency of RSA algorithm.

***Problem with Current RSA Algorithm***

The RSA Algorithm is one of the most important public key encryption technique. The major functionality in RSA is Modular exponentiation. During compelling environments, like if the processor is a single core where parallel computing is very difficult, performing modular exponentiation for large numbers is tedious. This not only leads to processor cessation but also reduce the efficiency of the system. Some related problems include:

1. Speed-During Encryption of plain text

For large data, RSA algorithm compromises with time complexity since it can be very slow for encrypting such data by a single processor.

1. Security

In order to verify the reliability of the public keys during encryption, a third party component is required to validate. Data transfer can be compromised by the external certificate (modifying the RSA public encryption.). To support this claim- we can discuss two ways as to how the encryption can be cracked. One is guessing the correct key, since public key encryption is based on obtaining large number by multiplying two numbers hidden in public and private key encryption guess the number which factors the true value makes the work easy.

1. Speed-During Decryption of cipher text.

As mentioned above RSA decryption is very slow during restricting environments. From RSA algorithm, we know that C=memod N and M= cd mod N. For decryption, exponentiation with d requires numerous multiplications and squarings. Also for each squaring we have to calculate the modulo of N. These modulo operations for large numbers result in a very high time complexity which is not even constant. Moreover, by choosing a smaller d value also doesn’t solve this issue. Preferring to select a smaller d may result in failing security measures as one can easily find e value using brute force methods.

***How Problem relates to the course***

Let’s consider the stepwise approach to RSA algorithm learnt in the course.

1. Encryption

256 RSA (considering the weakest possible data)

M=12345678901234567890

e=65537

Take two large prime numbers

p=255972651020913583852708738755558492779

q=315961372360286283221530569994994089667

We know N=p\*q

N=80877470103268491690225776687889776985485516372419877405296906209811698014593.

To obtain the cipher text, the formula is

C=memod N

Here we have considered only 256 bit RSA. However, in reality larger bit values are taken into action. For example, if we consider the case of 2048 bit RSA , the size each of the value mentioned above is multiplied by 8 times. Imagine how the time complexity will be affected.

C=37526865766319703630750491158429044860161927049301804846356525193422406944367

1. Decryption

Phi(n)=(p-1)(q-1)

M= Cdmod N

d should chose in such a way that ((e\*d)mod phi(n)) is equal to 1.

M=12345678901234567890

Exponentiation by d – involves squaring 256 times ,128 (the number of 1s in d in binary excluding the first 1) multiplications. Also, for each squaring modulus of N is calculated.

These modulo operations for large numbers result in a very high time complexity which is not even constant. Moreover, by choosing a smaller d value also doesn’t solve this issue. Preferring to select a smaller d may result in failing security measures as one can easily find e value using brute force methods.

***History of the problems, how, why, where it arose***

RSA cryptosystem algorithm encrypts and decrypts securely as the large prime numbers involved in the process make it unimaginably difficult to temper the message. Despite being one of the secured cryptosystems it has cons associated with it, one big disadvantage is computation of modular exponentiation. RSA algorithm is slow when large data is encrypted by the same as the computation time is huge for modular exponentiation of large primes.

To further undermine the security, if hackers or eavesdropper over the internet can generate session keys as that of an e-commerce website, its just a matter of time before they gain access to user’s credit card details.

It requires a third-party verification and public keys rely on them. Data that is transferred by RSA algorithm could be compromised through middlemen who might temper with the public key system.

Searching the message

Although, RSA decryption done by factoring the primes consumes unreasonable amount of time, it isn’t stated as impossible to decrypt. Security could be compromised if hackers figure out other ways to crack the code apart from factoring of primes. In cases where the encrypted message is small its simpler to decode until a match of ciphertext is found.

Attack by guessing d

Another possible attack is ciphertext attack where the attacker knows both the plaintext and ciphertext, they just must discover the private key d. Once the value of d is known, its easy to factorize n.

Cycle attack

The cycle attack is another case where the ciphertext is encrypted repeatedly until the original text appears, number of re-cycles would decipher any crypted text. Though this process is slow the process is hastened up factorizing the modulus. An improvement to here would be using exponent that’s coprime to (p-1).(q-1) and is of form 216 + 1.

Smaller exponents

Most of RSA systems use e=3 for faster performance, that results in making it far easier process than expected for the attackers. If the same message is encrypted 3 times with same exponent and different moduli the message can be retrieved.

Padding Oracle attack

This is a high -level idea where the message recipient must check if the padding of the message is correct and an error is thrown if this check fails. This information alone is enough to decrypt the message. These attacks are vulnerable as attackers can recover pre-master secrets of TLS sessions using them. One such attack occurred in 1998 on PKCS # v 1.5.

Though RSA survived over 20 years, it has not been proven that breaking the algorithm and factoring large numbers are equivalent, but similarly it isn’t proven that factoring isn’t equivalent to breaking the RSA algorithm.

***Algorithm for Key generation:***

1. Make a set of prime numbers PR, which has ‘n’ prime numbers.

2. Choose any four prime numbers A, B, C and D from the set PR.

3. Calculate L (product of prime numbers) – L = A\* B \* C \* D

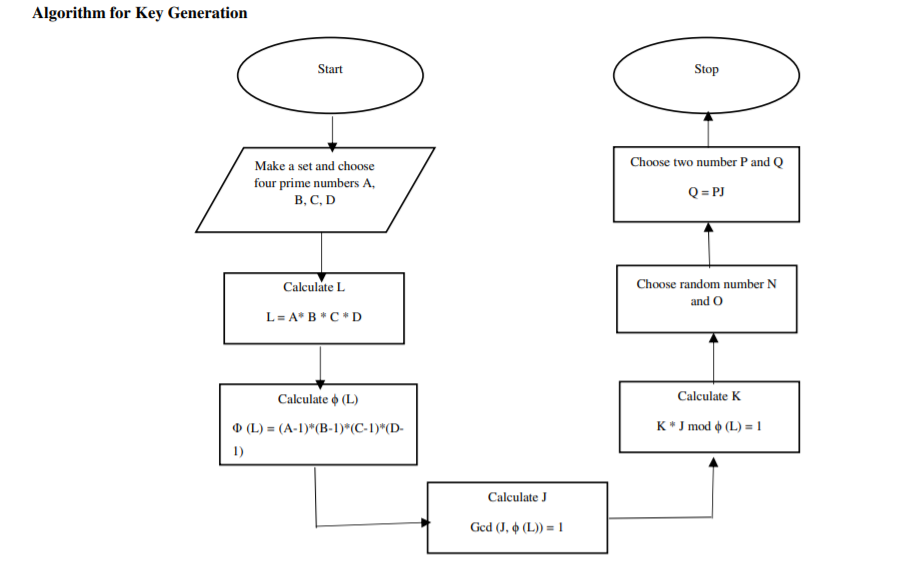
4. Calculate ϕ (L) – ϕ (L) = (A-1) \* (B-1) \* (C-1) \* (D-1)

5. Calculate J ( public key ), such that – Gcd (J, ϕ (L)) = 1

6. Calculate K ( private key ), such that – K \* J mod ϕ (L) = 1

7. Choose random number N and O, O should not be relative prime to ϕ (L)

8. Choose two numbers P and Q, such that Q = PJ

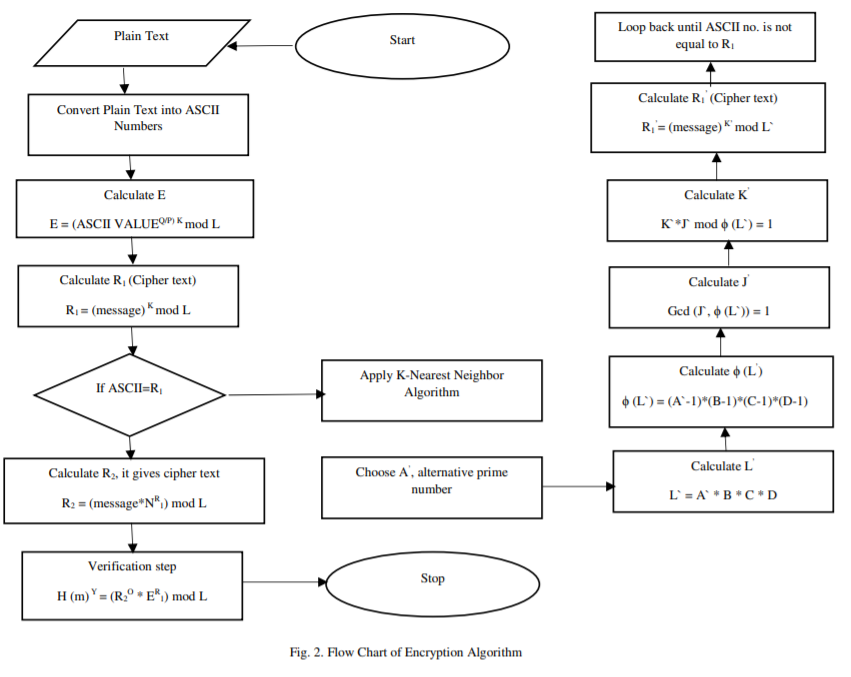


***Algorithm for Encryption***:

* The process of encryption encrypts the message character by character
* Convert the message that has to be encrypting into their respective ASCII values.
* Calculate ‘E’ for each ASCII value, such that – E = (ASCII VALUE Q/P) K mod L
* Calculate R1, as it encrypts the message and gives back cipher text of given plain text - R1 = (message) K mod L
* If the ASCII values and values of R1 comes same then apply K- Nearest Neighbor algorithm
  + Choose alternative prime A` from the set PR.
  + Calculate L` (product of prime numbers) – L` = A` \* B \* C \* D
  + Calculate ϕ (L`) – ϕ (L`) = (A`-1) \* (B-1) \* (C-1) \* (D-1)
  + Calculate J` (public key), such that – Gcd (J`, ϕ (L`)) = 1
  + Calculate K` (private key), such that – K` \* J` mod ϕ (L`) = 1
  + Calculate R1 ` , it encrypts the message and gives back cipher text R1 ` = (message) K `mod L`
  + Loop back the whole process until the ASCII value is not equal to R1 value.
* After that calculate R2

R2 = (message \* NR 1) mod L 7

* Verification – H(m) Y= (R2 O \* ER 1) mod L

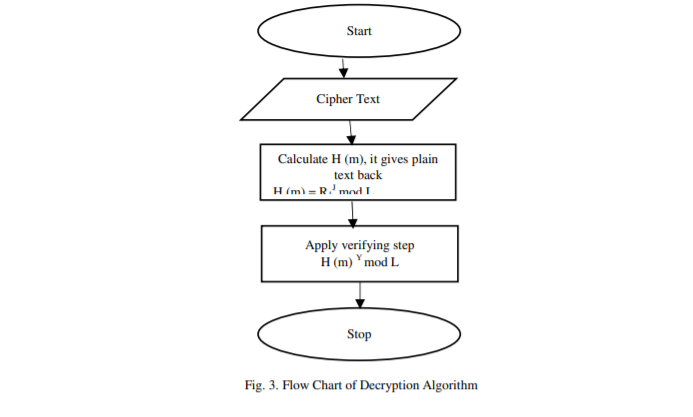


***Algorithm for decryption***:

1. Calculate plain text back again from cipher text

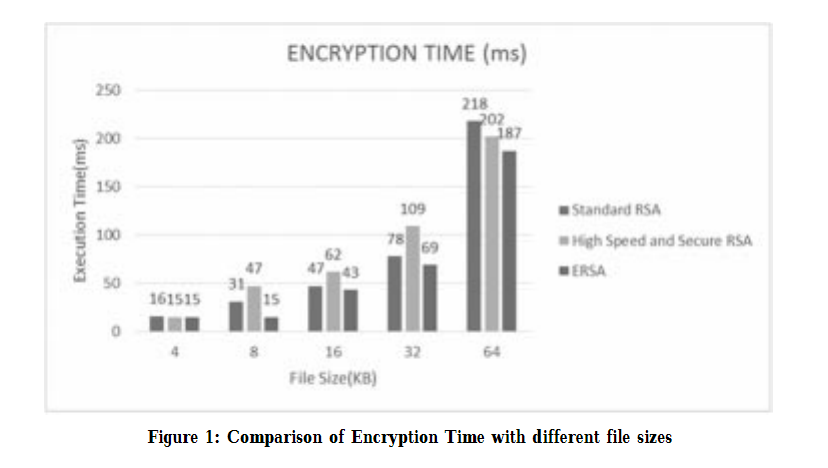
H (m) = R1 J mod L

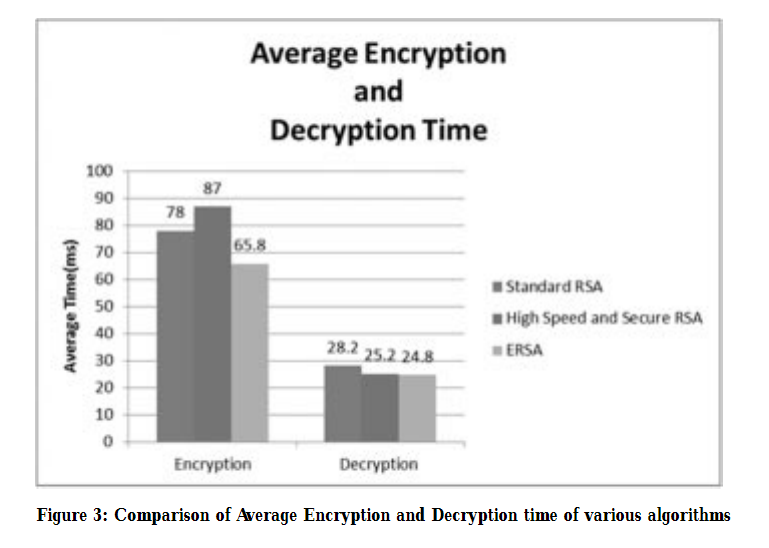
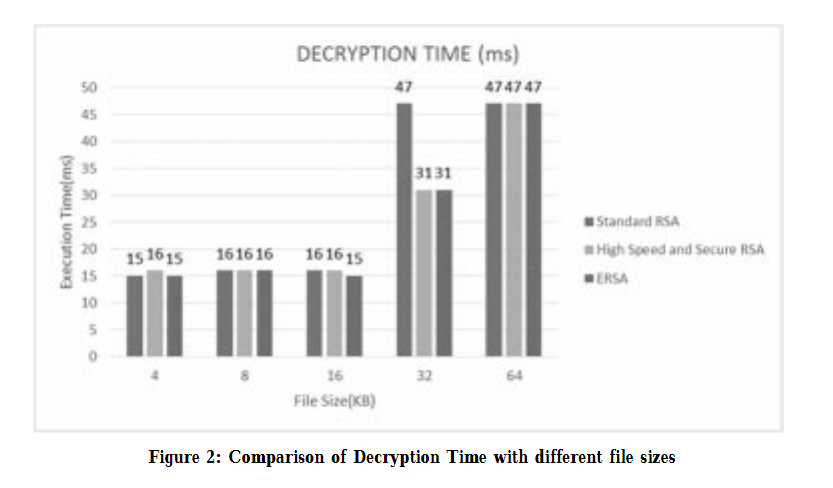
1. Verification –

H (m) Y mod L

***Applications of Enhanced RSA***:

* This approach increases the security by using k-nearest neighbour algorithm.
* It increases the arbitrary of the calculated value in the cipher text since this eradicates the problem of redundancy in cypher text.
* It is nearly impossible for anyone to decrypt the message as this approach uses two public keys and four randomly generated prime numbers.
* This improves approach has verification at both the sender end and also receiver end.
* It reduces the encrypted and decrypted time for encrypting and decrypting the messages.





***Current Research on Topic:***

Tal Rabin was recognized for her innovation and ongoing contributions to the field of cryptography. Tal Rabin is a computer scientist, head of the cryptography research group at IBM’s Thompson J. Watson Research Center.

Her research primarily focuses on cryptography and network security, specifically the design of efficient and secure encryption algorithms. She is currently working on generating high bit RSA algorithm, this enhances the security by taking large prime numbers. She also looks at secure distributed protocols and the theoretical foundations of cryptography, as well as number theory, and the theory of algorithms and distributed systems. Her works also concentrates on making communications over the internet more secure. Rabin’s most cited works in this field explore the design of digital signature schemes, among other applications, in protocols for secure web communications.

William Kuszmaul is currently pursuing Ph.D in Massachusetts Institute of Technology (MIT).William’s research focuses on algorithms and high-performance engineering which is funded by Hertz Fellowship, the NSF GRFP Fellowship, and the MIT Akamai Fellowship. Currently, he is carrying the research work along with his team on the security of web applications in cloud through the help of encryption and decryption algorithms. This research work is being conducted in Massachusetts Institute of Technology Computer Science and Artificial Intelligence Lab. William stated that “if the RSA encryption algorithm used carelessly it provides less than 99.8 % security”.

References

1. <https://security.stackexchange.com/questions/57205/why-is-rsa-decryption-slow>
2. [https://sci-hub.tw/https://ieeexplore.ieee.org/document/7977330](https://sci-hub.tw/https:/ieeexplore.ieee.org/document/7977330)
3. Possible attacks on RSA – Ronan Killen <http://www.members.tripod.com/irish_ronan/rsa/attacks.html>
4. <https://blog.trailofbits.com/2019/07/08/fuck-rsa/>
5. *Cryptography and Data Security*, Dorothy Elizabeth Robling Denning Reading (Mass.). - London : Addison-Wesley, 1982