# IMPLEMENTATION OF SECURE IMAGE TRANSFERRING BY USING RSA AND SHA-1

***Khet Khet Khaing Oo[1], Yan Naung Soe[2]***

*Faculty of Computer System and Technologies, University of Computer Studies, Myitkyina and 1011, Myanmar*

**Abstract**

***Nowadays, security is an important thing that needs to transfer data from admin to user safely. Communication security requires the integration of people, process, and technology. There are so many different techniques should be used to safe confidential image data from unauthorized access. Information security is becoming more in data storage and transmission from data exchange in electronic way. Industrial process is used to images, this is useful to protect the secure image data files from unauthorized access. To implement this system RSA ( Rivest Shamir Adelman) public key cryptography with OAEP (Optimal Asymmetric Encryption Padding ) and SHA-1 hash algorithms are used. This system security for image transferring using OAEP technology. The implementation of RSA-OAEP algorithms for Secure Image Transferring presented by using C#.Net programming language.***

***Keyword: Cryptography, OAEP with RSA public-key algorithm, SHA-1***

## 1.INTRODUCTION

Cryptographic techniques convert original data (message, image and so on)to another data. Cryptographic is not easily to understand; to keep the data secret between users, another way, it is necessaries that any another don't get to know the content without a key for decryption. This system aims to implement asymmetric (public key) encryption that is provably secure. Encryption is the used to converting plaintext data into cipher text in order to conceal its meaning and so barring any unauthorized recipient from retrieving the original data. Optimal Asymmetric Encryption Padding is a algorithm for secure messages created by Mihir Bellare and Phil Rogaway. [8] OAEP is a padding plan often used to content with RSA encryption. A public key which encrypted data only be decrypted key but it requires a private key which is associated together. It must be careful that the encrypted public key and decrypted public key must not be the same.

## 2.PUBLIC KEY CRYPTOSYSTEM

RSA cryptosystem was invented by Ron Rivest, Adi Shamir, and Leonard Adleman at MIT in 1978. [4,6] RSA (Rivest–Shamir–Adleman) algorithm is a public-key cryptography system. These algorithm is profoundly used in electronic commerce protocols, and is confided to be secure given enough long keys and the use of up-to-date implementations. They went on to patent the system and to form a company of the same name—RSA Security [2].

The RSA algorithm include of the following perform: Key Generation, Encryption and Decryption. [4]

• Key Generation

RSA consist a public key and a private key. The public key use to everyone and encrypting messages. Messages encrypted with the public key can lone be decrypted using the private key. The keys for the RSA algorithm are produced the following steps: [1, 6]

Select p, q; where p and q are both prime, $p \neq q$

Calculate $n=p*q$
Calculate $\Phi(n) = (p-1)(q-1)$
Select integer e; $gcd(\Phi(n), e) = 1$; $1<e<\Phi(n)$
Calculate d; $d=e-1 \bmod \Phi(n)$
Public Key; KU = {e, n}
Private Key; KR = {d, n}

- Encryption

Plaintext: M < n
Cipher text: C = $M^e$ mod n

- Decryption

Cipher text: C
Plaintext: M = $C^d$ mod n

## 3.OPTIMAL ASYMMETRIC ENCRYPTION PADDING

Optimal Asymmetric Encryption Padding (OAEP) with RSA encryption is a padding plan often used jointly with RSA encryption in cryptography. Bellare and Rogaway was introduced by OAEP and atter standardized in PKCS #1v2 and RFC 2437.[8]
*This algorithm is a design of the Feistel network that uses a pair of random oracles G and H to operate the plaintext prior to asymmetric encryption. When integrating with any safe trapdoor one-way permutation f, this continuation is verify in the random oracle model to result in a combined scheme which is semantically safe under select plaintext attack (IND-CPA). When performed with safe trapdoor permutations (e.g., RSA), it is also verify safe opposite select ciphertext attack. OAEP can be used to structure an all-or-nothing evolve. OAEP satisfies the following two goals:*

- Add an element of random which can be used to change a deterministic encryption algorithm (e.g., traditional RSA) into a possible plan.

- Prohibit partly decryption of cipher texts (or other information leakage) by ensuring that the enemy cannot recover any portion of the plaintext without being able to reverse the trapdoor one-way permutation f.

RSAES-OAEP is a public-key encryption algorithm which incorporate the encoding method Optimal Asymmetric Encryption Padding (OAEP) with the RSA encryption originally RSAEP. RSAES-OAEP takes a plaintext as input, changes it into an encoded massage via OAEP and relates RSAEP to the result which is interpreted as an integer using an RSA public key. The RSA was created by Ronald L. Rivest, Adi Shamir, and Leonard Adleman, while the OAEP was created by Mihir Bellare and Phillip Rogaway which was later advancement by Don B. Johnson and Stephen M. Matyas. .[3]

### *3.1. OAEP Encryption and Decryption*

The encryption operation make a cipher text from a message with a recipient's public key, and the decryption operation recuperates the message from the cipher text with the recipient's related private key. [3] The following Depicts OAEP encryption operation and decryption operation seen to *Figure 1*. In the encryption operation:

- Pad the plaintext to produce m-bit message M, if M is less than m-bit
- Select a random number r of k-bits. (used only once)
- Operat one-way function G that inputs r-bit integer and outputs m-bit integer. This is the disguise.
- P1 = M Å G(r)
- P2 = H(P1) ⊕ r, task H inputs m-bit and outputs k-bit
- C = E(P1 || P2). Operate RSA encryption here.

In the decryption operation:

- P = D (P1 || P2)
- Receiver first calculate the usefulness of r:
- H(P1) Å P2 = H(P1) Å H(P1) Å r = r
- Receiver recovers the original message M:
- G(r)ÅP1=G(r)ÅG(r)ÅM=M

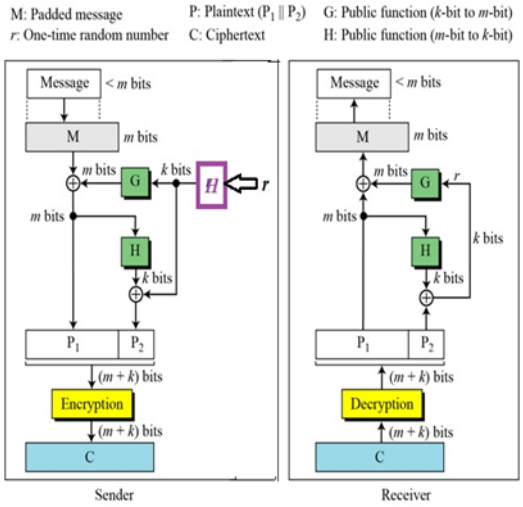


*Figure 1. Optimal asymmetric encryption padding (OAEP)*

## 4.HASH AND MESSAGE DIGEST

The reliable hash algorithms, SHA-1, SHA-256, SHA-384, and SHA-512 are between, one-way hash functions that can process a message to manufacture a dense depiction called a message digest. One-way hash functions( SHA-1, SHA-256, SHA-384, and SHA-512 )are iterative that can process a message to produce a condensed representation called a message digest. These algorithms enable the right to enact of a message's integrity; any change to the message wills, with a very high possibly, result in a different message digests. This valuable is effective in the causing and verification of digital signatures and message authentication codes, and in the generation of random numbers (bits).

Each algorithm can be following in two parts:

- Preprocessing and
- Hash computation.

Preprocessing includes padding a data, analysing the padded data into m-bit blocks, and setting first values to be used in the hash computation. Hash calculation generates a message from the padded message and exert use along with functions, constants, and word operations to between generate a series of hash values . Message digest is determine to used from the final hash value generated by the hash computation.
The four algorithms differ most significantly a number of bits security that are provided to data being hashed ; thus directly related to the message digest length. When a secure hash algorithm is used in combination with other algorithm, there may be requirements define elsewhere that needs the use of a secure hash algorithm with a certain number of bits of security.
Secure hash algorithm (SHA-1) was developed by NIST as a message digests function, which is necessary to ensure the security of the Elliptic Curve Digital Signature Algorithm (ECDSA). SHA-1 takes a message of length at most 264 bits and manufacture a 160-bit output called message digest. The message digest is then input to the ECDSA, which computes the signature for the message. The same message digest should be received by the support of the signature when the received version of the message is used as input to SHA-1. So both the sender and receiver of message computing and verifying a digital signature use the SHA-1 .[1] The SHA-1 is called safe because it is computationally impossible to find a message related to a given message digest, or to find two different messages which make the same message digest. Any change to a message in transfer will, with a very high Possibly, result in a disparate the designing of the MD4 message digest algorithm and is closely modeled after that algorithm.

## 5.DIGITAL IMAGE

Fundamentally, there are three types of image files: bitmap, vector, and metafiles. When an image is saved as a bitmap file, its information is saved as a collection of pixels, manifest as colored or black-and-white dots. Image photos is stored as a vector file, its information is stored as mathematical data. The metafile design can collection image information as pixels (bitmap), mathematical data (vector), or both There is no single pattern that is suitable for all types of images. According to [5], larger files take longer to load, require more disk space and can take longer to print, whereas small file sizes means greater performance. The most common file formats are JPEG, BMP, GIF,PICT and so on.

## 6. IMPLEMENTATION OF THE PROPOSED SYSTEM

### *6.1. Home Page*

This page consists of five buttons such as Home, Key Generation, Encrypt image, Decrypt image and Close.


*Figure 2.Home Page*

### *6.2. Key generation form*

When user click the Key generation button, the key generation form is appear. When the user clicks the Generate Key Pair button consists of the private key and public key for that user is generated randomly as shown in *Figure 3*.

*Figure 3. Key generation form*

### *6.3. Encryption image form*

The encryption form can be shown by clicking Encryption image button as illustrated in *Figure 4*. In this form, the user firstly types the value of R to produce random number. After that value of R is hashed by using SHA-1 hash algorithm. Then the input image is loaded by clicking Load Image button. The image is converted into binary value and padded to get equal 1024 bit blocks. The random number is generated by clicking Generate Random button based on hashed R value. After random number has been generated, the hash and mash values are produced to produce P1 and P2 by clicking Mask and Hash button. Finally the P1 and P2 are concatenated and encrypted using RSA public key cryptography with public key to produce cipher text.

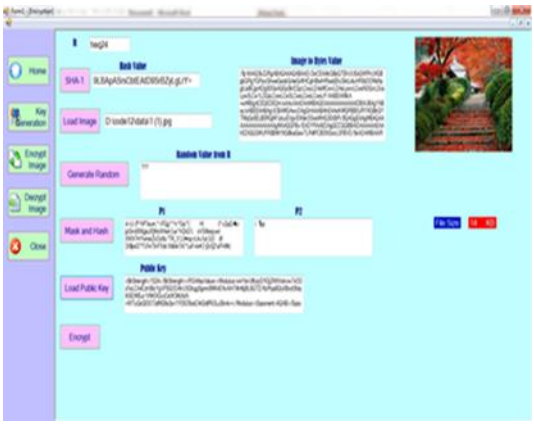

*Figure 4. Encryption image form*

### *6.4. Decryption image form*

In *Figure 5* shows the decryption image form. In this form, the decrypted image is firstly loaded and then it is decrypted by RSA with private key to produce P1||P2. The P1||P2 is split to get P1 and P2. The P1 is hashed and then performing xored with P2 to recover hashed value (r). The hashed value (r) is input to PRNG to produce random. The random number and P1 are xored to recover the padded message. Finally the original image is recovered by clicking Recover Image button. The user can be existed from this program by clicking Close button.

*Figure 5. Decryption image form*

## 7.CONCLUSION

This paper is mainly intended to encrypt/decrypt the images for secure image transferring system using with public key RSA algorithm and SHA-1 hash algorithm . In the image encryption/decryption algorithms, the

encryption/decryption time and secrecy are major goals. Security is very important issue in many applications. The proposed system supports the commercial level of key length 1024 bit for asymmetric key cryptography (RSA). This system is implemented by using .Net frame work (Window XP, Window Vista, Window7 and later). This system can only encrypt image files. This system can be encrypted any image file types such as JPEG, BMP and so on. This system cannot support other file types such as audio files (eg: mp3 and wav), video files (eg: mp4, flv and avi), document files (eg: pdf, ppt and doc) and executable file (eg: exe). And also this system cannot support for image encryption with private key.

## REFERENCES

[1] Bruce Schneier, "Applied cryptography, Second Edition: Protocols, Algorithms and Source Code in C", ISBN: 0471128457.
[2] Narasimham Challa and Jayaram Pradhan, "Performance analysis of public key CRYPTOGRAPHIC systems RSA and NTRU".
[3] Neha Garg, " Comparison of Asymmetric Algorithms in Cryptography", 2014.
[4] Nick Galbreath, "Cryptography for Internet and Database Application Developing Secret and Public Key Technique with Java TM".
[5] Mohammad Alimoh'd bani Younes, "An Approach to Enhance Image Encryption Using Block-based Transformation Algorithm", 2009.
[6] William Stalling, "Cryptography and Network Security".
[7] White.B, Gregory Cisco Security Certification Exam Guide, McGrew-Hill, 2003.
[8] http://en.wikipedia.org/wiki/Optimal_asymmetric_encryption_padding.