Effective Information Security Management in Enterprise Software Application with the Revest- Shamir- Adleman (RSA) Cryptographic Algorithm

Abilimi A. Christopher, Lecturer, Department of CSE, Christian Service University, Kumasi, Ghana.
Dr. Hillar Addo, Lecturer & Coordinator, OUM Programme, Accra Institute of Technology, Ghana.
Dr. Ing. Edward Opoku-Mensah, Lecturer, Department of ITE, UEW, Kumasi, Ghana.

ABSTRACT

This research aims to examine the security strength of RSA encryption algorithms that can be used in enterprise software systems in organizations so as to improve security in enterprise software systems. Instruments such as key generation, encryption and decryption strength of the algorithms are examined. The Inverse Congruential Generator (ICG) for random numbers method was adopted. This method was used instead of Padding Scheme, to solve the problem of semantic insecurity in the RSA Cryptosystem. It was found out that RSA Cryptosystem using Inverse Congruential Generator to generate random numbers reveal user confirmation and privacy can be both obtained like the Padding Scheme with public-key cryptography. It therefore concluded with the design of an improved Rivest-Shamir-Adleman (RSA) algorithm for public-key cryptography.

Keywords *Cryptography, Rivest-Shamir-Adleman, Inverse Congruential Generator, Algorithm.*

1. Introduction

The public-key based cryptography (RSA) is based on the assumed complexity of big integer factors, and the problem of factorizing them. RSA represent the last names of three researchers, namely Ron Rivest, Adi Shamir and Leonard Adleman, in public key cryptographic algorithm which they first described in the year 1977 **[14]**. An English mathematician (Clifford Cocks), came out with a comparable study in early 1973, however it was classified awaiting 1997 **[14]**. Anyone who wants to use RSA generates and after that publishes the product of two large prime numbers (first & second), alongside among a supplementary value, as public key for them. The user must store the prime factors as a secret. A message can be encrypted by anyone with the public key; however through presently available methods, for large enough public key, the message can feasibly be decoded by only somebody by means of acquaintance of the prime factors **[13]**.

RSA cryptosystem basically has two public quantities that can be termed to as *n* (modulus) and *e* (which is the public key), including private quantities *d* (private key) and λ **(*n*)**. λ **(*n*)** is distinct as all the prime factors' Least Common Multiple (LCM) of *n*. You choose the secret exponent *d* as an integer lesser than λ **(*n*)** and relatively prime to λ **(*n*)**. You determine the public key *e* as the "multiplicative inverse" of *d* and might be computed using the relation $d = e^{-1} \bmod \lambda(n)$ **[1].**

Encryption or decryption and the signing or signature verification process **[1]**, are the two processes in RSA cryptosystems. They stated in their Research that before the message is encrypted or signed, it must be divided into a number of blocks (the message to be sent): $m_1, m_2, \cdots m_j$ ($m_k < n$ for $k \in [1, j]$) with the same word length in the case it has larger word length than the modulus *n*. During the encryption/decryption process, the secret key d is used to recover the message *m* from the encrypted information *c* as $m = c^d \bmod n$, and the public key *e* is used to encrypt the message *m* as $c = m^e \bmod n$. During the signing or signature verification procedure, the public key *e* is used to verify the signature *s* by checking whether $s^e \bmod n$ equals to the message *m* and the secret key *d* is used to attain the signature *s* from the message *m* by using $s = m^d (\bmod n)$.

The variable *n*, public quantity of the two prime numbers of RSA cryptosystem is divided into factors of two large prime numbers termed as *p* and *q* in that order and it is associated with the relations $n = p \cdot q$. It also has an *e*, public quantity and *d*, secret quantities and $\lambda(n)$. The *p* and *q* are two positive integers' that are normally selected to have similar word length. Public quantities {*n*, *e*} are prepared public and {*p*, *q*, $\lambda(n)$, *d*} are made private in the two prime numbers of the RSA cryptosystems.

In a case of more than two prime numbers RSA cryptosystem, the public modulus *n* has the least number three prime factors **[2]**. Typically the initial three prime numbers are denoted as *p*, *q* and *r*, so that $n = \sum_{k=1}^{j} i_k = p \cdot q \cdot r \cdots i_j$. Likewise, {*n*, *e*} are public and $\{p, q, r, \cdots i_j, \lambda(n), d\}$ are private **[14]** in multi-prime cryptosystem. A distinctive case of multi prime factors RSA cryptosystem is the three-prime factors RSA, for which the modulus has three prime factors *p*, *q* and *r*. The prime numbers *p*, *q* and *r* are prime numbers and need to be factorised using Chinese Remainder Theorem (CRT).

2. Chinese Remainder Theorem (CRT) Based RSA

The descriptions of the Chinese Remainder Theorem (CRT) according to **[18]**, assumes in the initial place as the number $n = \sum_{k=1}^{j} n_k$ and $x_1, \ x_2, ..., x_j$ must all be positive integers, in which $n_1, \ n_2, \ ..., \ n_j$ must all also be positive integers and comparatively prime to each other, i.e. $\gcd(n_i, n_k) = 1$ for any $i, k \in [1, j]$ where i is not the same as *k*. Afterwards, the scheme of congruencies

$$x \equiv x_1 \bmod n_1$$

$$x \equiv x_2 \bmod n_2$$

…

$$x \equiv x_k \bmod n_k \qquad (k = 3, \cdots, j)$$

With a concurrent solution as x. x might be evaluated from:

$$x = (\sum_{k=1}^{j} x_k \cdot r_k \cdot s_k) \bmod n$$

Where $r_k = \frac{n}{n_k}$ and $s_k = r_k^{-1} \bmod n_k$ for all $k = 1, \cdots, j$.

The Chinese Remainder Theorem is an important step in cryptography that may be adopted to speed up the process of decryption and signing procedure in two-prime or multi-prime RSA**[10][13]**. The RSA applications may be adopted to increase the speed of the computation process by CRT and this is called CRT-based RSA. It was acknowledged in this research paper that RSA is the majority deployed public key cryptosystem. It is applied in some wireless devices, securing web traffic and electronic mail. Because RSA depends on arithmetic modulo of large numbers, it might be sluggish in unnatural environments. In an illustration, 1024-bit RSA decryption on a handheld device takes 40 seconds. Likewise, on a burdened web server, RSA decryption radically decreases the amount of SSL requests per second that the web server can hold at a time. In general, RSA's performance can be improved using special-purpose hardware. Modern RSA coprocessors can complete a lot of 10,000 RSA decryptions per second (1024-bit modulus) and even quicker processors are expected in the near future.

3. CRT attacks-Based RSA

The discipline of infringement on the encoded data in RSA cryptosystem is its attack. The attacks are basically known to be an attack on the elegant Integrated Circuit (IC) card tool of the RSA cryptosystem. This attack can be divided into two simple classes namely; the implementation attacks and the usual mathematical attacks **[4]**. The usual mathematical attacks are algorithms modeled as model mathematical objects. The attacks on the mathematical components are characteristically universal and generally theoretical relatively than operational. The physical implementation attacks strategies are always specific instead of generalized **[4]**. The implementation vulnerabilities historically are known to break down the RSA cryptosystems because the attacks are comparatively very complex to manage **[9]**. Thus, the study of this thesis is concentrated on the implementation attacks.

4. Fault Attack and the Existing Countermeasures

There is high risk of the incidence of hardware faults on every tamperproof tools of cryptosystems that employ public key cryptography for user validation with no particular solutions **[1]**. For instance, cards with the aim

of personalized cellular phones, smart cards that are meant for storage of data, cards that produce digital signatures or validate users for distant login to group networks are target susceptible to the attack.

The fault in the hardware attack is when the opponent introduces a certain type of error into the hardware tools in such a way that the application would have invalid response and then produce an erroneous output. After that the opponent will be able to have access to the secret data of the application with the aid of the flawed output from the scheme. There are two main steps for which the faulty hardware attack of every cryptosystem is made up of. The initial method is the injection of a number of faults into the application at the right moment. The exploitation of the faulty outcomes is used to have access to the secret data of the cryptosystem. The process involved in the fault-based attack is shown in Figure 1.

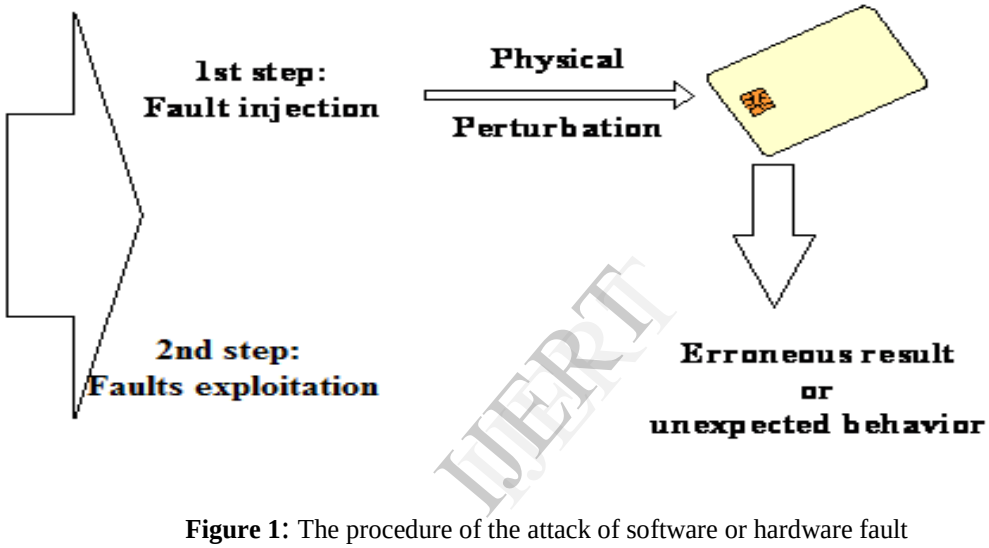

Figure 1: The procedure of the attack of software or hardware fault

The accomplishment of the attacks of the hardware of a cryptosystem is depended on the conditions of the following three statements **[3]**:

(i). The attacker knows the message to be signed.

(ii). Through System computation, random fault occurs.

(iii). The erroneous responses or invalid results are given out of the cryptosystem.

One of the ways to prevent RSA cryptographic tools from attacks is ensuring that anyone none of the three conditions above is successful. With reference to the initial condition, various remedies have been proposed to allow the adversary to have no access to the information to be signed. The Probabilistic Signatures Schemes (PSS) protocols **[12]** and the Full Domain Hash (FDH) (IEEE standard 1363-2000, 2000) are two of the above remedies which have been homogeneous. Mutually in PSS and FDH systems, a unique data *m* is transformed into a hash value *mHash* via the application of a one way hash function to the data *m*. Afterward there is a transformation of

the hash data *mHash* into a message that is encoded called *EM*. In conclusion the *EM,* by the use of a private key, is used to generate *s* (signature). The encoded data or message *EM* cannot therefore fall into the hands of the attackers in order for them to factor the application systems.

There are also some remedies that have been developed and presented to counteract the conditions for the step two as well as the third step, and these enable the systems to stop sending signatures and or responses that contain error components out from the application tools or the systems. Staying away from sending out or receiving from incorrect replies or defective output is the initial plan used as a scrutiny technique **[5]**. The repetition of the calculation and the checking of whether the signatures received in the two cases are the same signature, make the system signing slow down because of factors of two in the signing operations, and this method is the most obvious way to compare the *m* and *s* to see whether the message can be retrieved from the signature and this aids us to determine the signature's correctness. The shortcoming however is that checking whether the message can be received from the signature or the repetition of the calculation, or how fast the computation is, approximately reduces the speed by the two factors (*m* and *s*). A method of inspection using simple arithmetic was instituted by Shamir in order to find out whether the output from the intermediate computation (called the intermediate results) are assessed presented a checking method with simpler calculations, in which the intermediary output are checked ahead of the signature's computation. If this (intermediary output) has fault-free claims, next is the computation of signature that is to be sent from the system, or else, the re-computation of the intermediate output as well the checking to ensure that the result that is error-free is repeated **[5]**.

Furthermore **[15]** proposed another remedy apart from the countermeasures of the above three conditions. Yen et al initiative is the revision of the computation method of the CRT-based RSA signature, and this enable an erroneous signature to still keep information stored in the Chinese Remainder Theorem based RSA cryptosystem secret. Yen et al. anticipated two types of protocols that can be used **[17]**. This proposal ensured that an error in one of the module will automatically have an influence on the others modules and or the entire calculations, and hence ensure that even though faulty signature but it will not reveal the secret information in the cryptosystem.

5. Timing Attack

The measure of the time taken by an attacker to perform operations in cryptography in a form that user's key (private key) information is deciphered is what is meant by a timing attack **[6]**. The attacker can do this by determining the time needed to carry out operations of a private key in a smartcard for which a private RSA key is stored, then the opponent can determine the decryption exponent *d* of the private key if the card is tamper opposing **[16]**. By calculation, this kind of attack needs the opponent to know the cipher text only and hence it is a cheap type of an attack in terms of computation. The system may be used when opponents can perform precise time measurements. This may include tokens used in the cryptography and network based cryptosystems. He **[16]** also proposed some techniques to avoid the attacks on the time of an RSA cryptosystems. In the first proposal, every

systems operation should acquire precisely equal quantity of time used in their computation. Secondly the addition of random interruption to the time used in process by the opponent becomes unachievable and this ensures the time measurement becomes inconsistent. If the data input to the modular exponentiation function is not known to the attacker then attackers cannot get access to the secret code. This method is called a blind signature method.

6. Power Attack

The interpretation of power consumption measurements of cryptographic operations gathered so as to expose the secret key exponent *d* of a smartcard to attackers is known as a power attack **[17]**. A number of remedies to the power attack problem of a cryptosystem were initiated. He **[17]** stated that the information leakage to attackers can be avoided by the reduction of the signal sizes and the choice of smaller leak information in terms of power usage. Nevertheless, building the attack of the system intricately in order to shield the tool may most importantly raise the price and size too. Secondly one could introduce into power consumption dimensions some noise in order to cause opponent's dimensions to be inconsistent.

7. The Improved RSA-Cryptographic Algorithm.

This algorithm introduces a random number generator that generates integers of any sizes. Two prime numbers are then selected from the generated numbers and this adds some kind of randomness into the RSA-Cryptographic algorithms. This ensures that RSA-Cryptography is semantically secured without Padding Schemes. For example let's consider the case of a system trying to generate numbers from zero to 8000, a simple fix to this is the algorithm shown below. It's a modified version of what's called an Inverse Congruential Generator:

$$r_n = p_1(r_{n-1})^{-1} + (Y_{n-1} mod p_2) \text{ .....(*),}$$

$$Y_n = r_n mod S \forall S = \{0 \dots 1000\} \text{ .....(**)}$$

Where r is a running static integer that is allowed to overflow, p_1 and p_2 are prime numbers, and Y is the value you're interested in using for your random choice. This algorithm is summarized below:

i. Choose two distinct prime numbers p_1 and p_2, for each $p_j \geq 5$.
ii. Generate random numbers in a specified range using the Modified Inverse Congruential Generator formula named (*) above to get r set of random numbers.
iii. Using the formula (**), generate Y set of random numbers
iv. From the set of random numbers Y, generate two different primes p and q
v. Calculate the modulus n = p × q
vi. Calculate the totient $\phi(n) = (p - 1) \times (q - 1)$
vii. Select for public exponent an integer e such that $1 < e < \phi(n)$ and $\gcd(\phi(n), e) = 1$
viii. Calculate for the private exponent a value for d such that
    i. $d = e^{-1} \bmod \phi(n)$
ix. Public Key = [e, n]
x. Private Key = [d, n]

7.1 The Inverse Congruential Generator.

The Inverse Congruential Generator Algorithm is as follows:

- $X_n=(a * \sim X_{n-1} + b) \bmod m$
- m should be prime
- ~y is the multiplicative inverse of y in the field over {0,1,...,m-1}.

To decode
Listing 1: Inverse congruential PRNG

```
Unsigned int rprime;
Static unsigned int r;
/*begin critical section*/
if (r==0 || r==1 || r==-1)
 r=rprime;  /* keep from getting stuck */
r = (9973 * ~r) + ((Y) % 701);  /* the actual algorithm */
Y = (r>>24) % 9;  /* choose upper bits and reduce */ /*end critical section*/
```

7.2 The Complexity of Inverse Congruential Generator

- The elementary steps are r and Y.
- Any algorithm with two input steps runs at a quadratic time(r*Y).
- Therefore Inverse Congruential Algorithm runs in quadratic time O(n*m).

7.3 Padding Scheme Algorithm

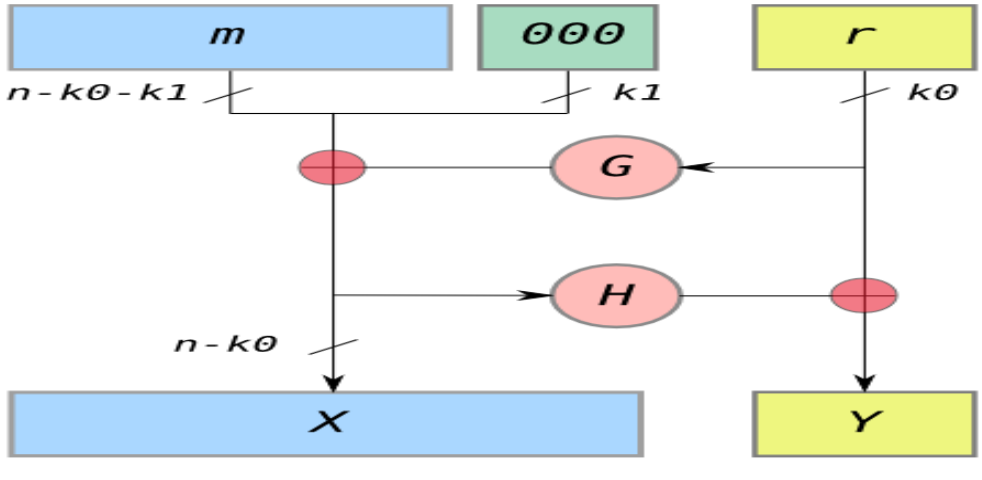

Figure 2: Oaep for Padding Scheme in RSA Algorithm **[8]**

From Figure 2,

- RSA modulus bits number, n .
- The static integer's protocol, k_0 and k_1.
- The bit string is $(n - k_0 - k_1)$ with plaintext message m.
- Some cryptographic hash functions fixed by the protocol are typically G and H.

In coding the above algorithms,

1. All messages or data uses k_1 to pad with zeros to be $n - k_0$ bits in length.
2. r is a unsystematic k_0-bit series
3. G increases the k_0 bits of r to $n - k_0$ bits.
4. $X = m00...0 \oplus G(r)$
5. H decreases the $n - k_0$ bits of X to k_0 bits.
6. $Y = r \oplus H(X)$
7. The result is shown in the Figure above as $X \parallel Y$.

In decoding the above algorithms,

1. convalesce r from $r = Y \oplus H(X)$
2. convalesce m from $m00..0 = X \oplus G(r)$

The "every or none" security is from the fact that to recover m, you must recover the entire X and the entire Y; X is required to recover r from Y, and r is required to recover m from X. Since any changed bit of a cryptographic hash completely changes the result, the entire X, and the entire Y must both be completely recovered. This repetition makes the algorithm run in polynomial time **[7][2]**.

8. Conclusions

The study's identification of factors contributing to the RSA encryption algorithms weaknesses and subsequent loss of assets (data and finance) are worthwhile. One of those factors identified was the semantic security problem. The semantic security problem can be solved by using padding scheme **[2]**. Rivest, Shamir, and Adleman (1978) **[11]** in their paper showed that padding scheme used in the RSA Algorithm has a complexity time of polynomial run time. The Research presented an alternative method called Inverse Congruential Generator **.** The running time (complexity) of this generator was determined to be quadratic time (r*Y) by computing and counting the number of steps in the algorithm. Inverse Congruential Generator was then used in the RSA Cryptographic to present the proof of the Improved-RSA Algorithms mathematically. It can also be concluded from the findings that RSA with Inverse Congruential Generator is more efficient (quadratic run-time) than RSA with Padding scheme (polynomial time)**[2]**.

9. References

[1]. Bell Communications research (1996). *New threat model breaks crypto codes.* Bellcore press release, Morristown.

[2]. Billy, B. (2012). *Covert timing channels, caching, and cryptography*. Billy Bob Brumley. Doctoral dissertation for the degree of Doctor of Science.

[3]. Boneh, D., DeMillo, R., & Lipton, R. (2001). On the importance of checking cryptographic protocols for faults. *Journal of Cryptology*, 14 (2), pp. 101-119.

[4]. Dan, B. (2000). Twenty years of attacks on the RSA cryptosystem. Retrieved on 10th May, 2012, from http://crypto.stanford.edu/~dabo/papers/RSA-survey.pdf.

[5]. Dinur, I, & Shamir, A. (2013). Applying cube attacks to stream ciphers in realistic scenarios. *Cryptography and Communications* 4(3-4) pp 217-232.

[6]. English, E. & Hamilton, S. (1996). Network security under siege: The timing attack. *IEEE Computer*, 29, pp. 95-97.

[7]. Henk, C.A. ,Van T., & Sushil J. (2011). Encyclopedia of Cryptography and Security, 1, p 385

[8]. Paillier, P., & Villar, J. (2006). Oaep for Padding Scheme in RSA Algorithm.

[9]. Peter, G. N., & Ulf, L. (2012). The IEEE Symposium on Security and Privacy Is Moving to San Francisco. *IEEE Security & Privacy* 10(2), 65-66.

[10]. Quisquater, J. J., & Couvreur, *C.* (1982). Fast decipherment for RSA public-key crypto-system. *Electronics Letters*, 18, (14), pp. 905 - 907.

[11]. Rivest, R., Shamir, A. & Adleman, L. (1978). A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM 21 (2): 120–126.* doi:10.1145/359340.359342.

[12]. Robert, G. (2009). *Cryptographic key management systems workshop*. NIST Key Management Workshop

[13]. RSA Security Inc. (2000). *Crypto FAQ.* (4th ed. pp 8-45). Cambridge Center,USA.

[14]. RSA Security Inc. (2000). *Crypto FAQ: Chapter 6: Laws concerning Cryptography, 6.3. Patents on cryptography*. (4th ed. pp 174-197). Cambridge Center, USA.

[15]. Thomas, S. (2000). *Messages: Power analysis attack countermeasures and their weaknesses*. Security Technology Research Laboratory.

[16]. Wolfram, S. (2002). *A New Kind of Science. Wolfram Media*, pp. 975–976. ISBN 1-57955-008-8

[17]. Yen, S., Kim, S., Lim, S., & Moon, S.(2003). RSA speedup with Chinese Remainder Theorem immune against hardware fault attack. *IEEE Transactions on computer*s, 52, pp. 461-472.

[18]. YU, L. R. (2002). The generalization of the Chinese Remainder Theorem. *Acta Mathematica Sinica, English Series*, 18, pp. 532-538.