A RESEARCH ON RSA CRYPTOGRAPHY, POSSIBLE ATTACKS, ITS SECURITY, AND FUTURE

Submitted in partial fulfillment of the requirements for

the Mathematics of Internet Course

Prepared by

Elif Cemre Durgut

Group 8

March 12, 2022

A RESEARCH ON RSA CRYPTOGRAPHY, POSSIBLE ATTACKS, ITS SECURITY, AND FUTURE

(ABSTRACT)

This paper introduces the RSA algorithm by Rivest, Shamir, Adleman which is a well-known public cryptosystem. After a brief introduction, how the keys should be chosen is explained. Following the set-up phase, the proofs of encryption and decryption are given along with a small example. Later, the paper discusses the possible attack ways in terms of cryptanalytic and side-channel attacks. RSA-OAEP to increase the security of RSA by padding follows the attacking methods. Finally, the future of RSA is presented in the case of the invention of quantum computers.

Keywords: RSA, RSA-OAEP, SPA, DPA, Shor's algorithm

CONTENT **PAGE**

INTRODUCTION

Security is one of the greatest concerns of today's world, consisting of zettabytes of data that need to be protected. Many cryptographic algorithms have been introduced from the past to the present, and most of them have failed due to different kinds of attacks. The examples of most popular and successful ones are RSA, DES, AES, Diffie-Helman, and ECC.

Cryptographic encryption algorithms are divided into two groups as symmetric keys and asymmetric keys. Cryptosystems with symmetric keys such as AES and DES use the same private key for encryption and decryption. However, two parties should share the same private key before starting the communication. This sharing is possible with public-key cryptosystems since they do not require private key sharing beforehand. One example of such a system is RSA which is introduced first by Rivest, Shamir, and Adleman in 1977. RSA uses different keys for encryption and decryption. The sender requests the receiver's public key and then s/he encrypts the message with the receiver's public key. The receiver decrypts the message with his/her private key. The algorithm is based on the hardness of the factorization problem.

RSA SETUP

As explained by Rivest et al. (1977), for the RSA setup,

Step 1) Two large and distinct primes are chosen randomly as p and q which are usually 1024 bit to 2048 bit long.

Step 2) n is calculated by multiplying p and q:

$$n = p \cdot q$$

Step 3) Euler totient function of n is computed as

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

Step 4) An integer e is chosen such that

$$0 < e < \Phi(n) \text{ and}$$

$$\gcd(e, \Phi(n)) = 1.$$

Step 5) Modular multiplicative inverse of e is computed as

$$d = e^{-1} \bmod \Phi(n).$$

This is also equal to $e \cdot d \equiv 1 \bmod \Phi(n)$. (1)

Whereas e and n are public keys, p, q, and d are kept privately. After setting up the parameters for the RSA. The sender requests public keys of the receiver and the encryption is done by using the public key e of the receiver:

$$y \equiv x^e \bmod n \qquad (2)$$

where x is the message ($x < n$), and y is the encrypted message.

Then, the receiver decrypts the ciphertext by using their private key d:

$$x \equiv y^d \bmod n$$

UNDERLYING MATHEMATICS

The mathematical reasoning behind this decryption operation can be explained using either Fermat's Little Theorem or Euler's Theorem. The following is proof of the decryption using Euler's Theorem:

$y^d \bmod n \equiv x$

$(x^e \bmod n)^d \bmod n \equiv x^{ed} \bmod n$ (from 2)

$e \cdot d \equiv 1 \bmod \Phi(n) \rightarrow e \cdot d = 1+k \cdot \Phi(n)$ for a non-negative integer k (from 1)

$x^{ed} \bmod n \equiv x^{1 + k\Phi(n)} \bmod n \equiv x \cdot x^{k\Phi(n)} \bmod n$

From this point on, there are two cases:

1) gcd(x, n) = 1

> **Euler's Theorem:**
> If gcd(a, b) = 1 then $a^{\Phi(b)} \equiv 1 \bmod b$

Using Euler's Theorem, given that the gcd(x, n) = 1, $x^{\Phi(n)} \equiv 1 \bmod n$.

$x \cdot x^{k\Phi(n)} \bmod n \equiv x \cdot 1^k \bmod n \equiv x \rightarrow y^d = x \bmod p$

2) gcd(x, n) ≠ 1 (This case has a low probability however, the algorithm still works.)

Euler's Theorem is not useful anymore.

> **Chinese Remainder Theorem:**
> Gıven that gcd(p, q) = 1,
> x = y mod p and x = y mod q, the equality x = y mod pq holds.

Fermat's Little Theorem:
If a is not divisible by p,
$a^{p-1} \equiv 1 \bmod p$

By Chinese Remainder Theorem, it is enough to prove that $y = x^d \bmod p$ and $y = x^d \bmod q$. Because these two equalities along with gcd(p, q) = 1 makes the decryption equality $x = y^d \bmod pq = y^d \bmod n$ hold.

$y = x^e \bmod n \rightarrow y = x^e \bmod p$

$\rightarrow y^d = x^{ed} \bmod p = x^{k(p-1)(q-1)+1} \bmod p$

$\rightarrow yd = x \cdot 1^{k(q-1)} \bmod p \rightarrow y^d = x \bmod p$ (by Fermat's Little Theorem)

Small Example

Since Alice wants to send a message to Bob, Bob first shares his public keys with Alice.

ALICE	*BOB*
	Chooses p = 3 and q = 11
	$N = p \cdot q = 33$
	$\Phi(n) = (p-1)(q-1) = 20$
	Chooses e = 3
	$d = e^{-1} \bmod \Phi(n) = 7$
	Sends (e, n) to Alice.
Message x = 4	
Encrypts $y \equiv x^e \bmod n = 31$	
Sends y to Bob.	
	Decrypts $x \equiv y^d \bmod n = 4$

SECURITY OF RSA CRYPTOGRAPHY

As stated by Rivest, Shamir, and Adleman (1977), it is not possible to prove the security of the algorithm, the only way is to investigate the attacking methods to break the algorithm (p. 11). Attacking methods can be divided into two parts as mathematical and non-mathematical approaches.

1) Mathematical attacks

Factoring n is the underlying mathematical problem of RSA cryptography. It enables the attacker to resolve d, thereby decrypting the ciphertext. Integer factorization for large numbers is considered as in class-NP. There are no efficient polynomial-time factoring algorithms for very large numbers found for the time being. The most efficient algorithm is the general number field sieve which is still slow due to exponential time complexity. The largest factored number so far is 829-bit, 250 decimal digits (RSA-250). This does not pose a threat to RSA today's RSA keys are at least 1024-bit. Shor's algorithm for the factorization problem will be discussed later in the future of the RSA cryptography part.

a) Brute Force

Brute force attack by trying all possible keys for d is not possible. Because $0 < d < \Phi(n)$, the key size is $|K| = \Phi(n) \approx n = p \cdot q$, p and q are usually at least 1024 bits as mentioned above.

b) Computing Φ(n)

Another way of attacking is to find $\Phi(n)$ so that d can be found by computing $d = e^{-1} \bmod \Phi(n)$. This attack method is considered as difficult as factoring n since n is a composite of two large prime numbers.

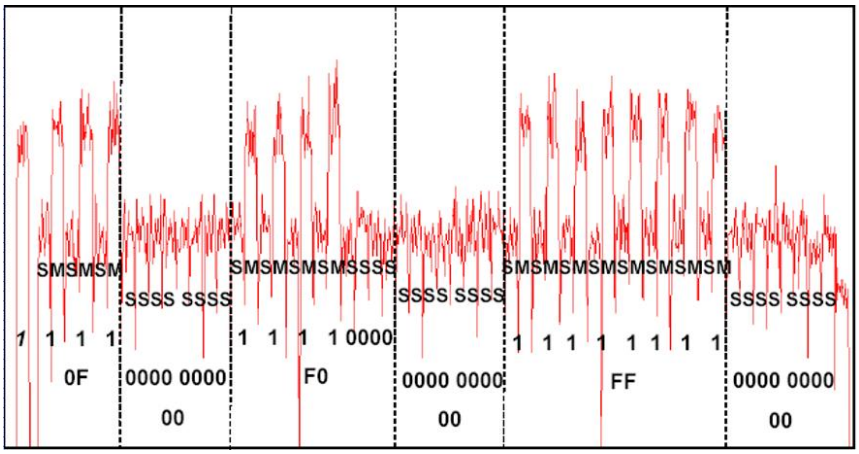
2) Side-Channel Attacks

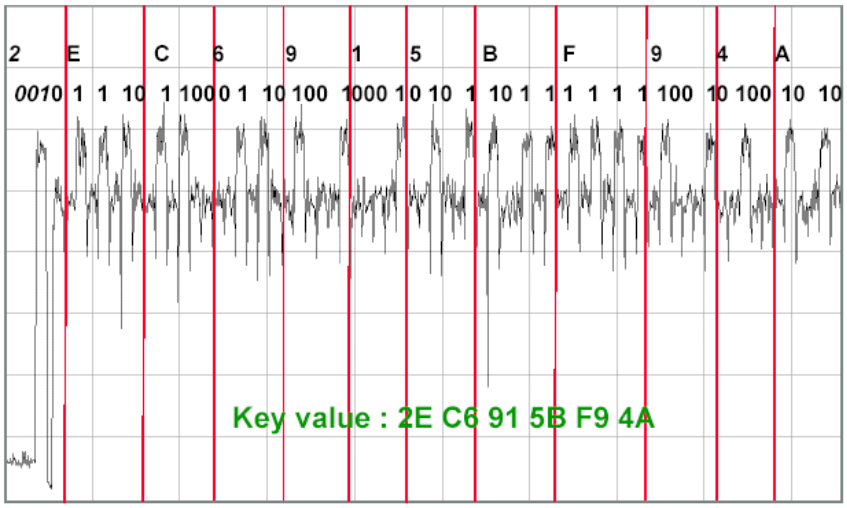
In addition to mathematical attacking methods, there are some side-channel attacks. As discussed by Kocher et. al., cryptosystem designers assume that the operations using private keys will be done in isolated secure environments (1999, p.1). However, this is not always the case. One of these side-channel attacks is power analysis which consists of two types as Simple PA and Differential PA introduced by Kocher in 1999 et al.. Power analysis attacks are applicable only for hardware devices such as smart cards and systems that store the private key embedded (Zhou & Feng, 2005, p.15).

a) Simple Power Analysis (SPA)

As explained by Kocher, nowadays modern electronic labs can measure voltage differences very accurately. Simple power analysis works by interpreting these voltage differences which are collected during cryptographic operations (1994, p. 2).

In the square-and-multiply exponentiation algorithm, which is used in RSA, the operation is based on the next bit. First, the exponent that is given as input is converted to binary expansion. Then the bits in the expansion are checked one by one. If the bit is 0, the result is squared. Otherwise, it is first squared and then multiplied. If the attacker can observe and record the power consumption during the decryption, s/he can find the private key by analyzing the power spectrum since the extra multiplication operation consumes more power.





As shown in the figures above, if the current bit is 1, power consumption is more than the power consumption of bit 0 case as it is possible to see the amplitude difference in the figure. Therefore, it is possible to extract the private key.

As defense mechanisms, noise can be added to the signal, or some dummy instructions can be added to the main operation.

Another approach to prevent SPA is side-channel atomicity proposed by Chevallier-Mames et al. in 2004. Figure 1 below represents the classical way of calculating the modular exponentiation which is not balanced. On the other hand, figure 2 demonstrates a balanced solution for the same problem which is called the multiply-always exponentiation algorithm.

Input: x, d = $(d_{m-1}, \ldots, d_o)_2$

Output: $y = x^d$

$R_0 = 1$

$R_1 = x$

i = m-1

while i >= 0

    $R_0 = R_0 \cdot R_0$

    if $d_i == 1$

        $R_0 = R_0 \cdot R_1$

    i = i - 1

return R_0

Figure1

Input: x, d = $(d_{m-1}, \ldots, d_o)_2$

Output: $y = x^d$

$R_0 = 1$

$R_1 = x$

i = m-1

k = 0

while i >= 0

    $R_0 = R_0 \cdot R_k$

    $k = k \oplus d_i$ ($\oplus$: XOR operation)

    i = i – ¬k

return R_0

Figure 2

Example using the algorithms above, x = 2, d = $(10)_{10} = (1010)_2$ (m = 4)

(S: Square, M: Multiply)

Square-and-multiply algorithm

i	d_i	R_0	*Operation*
3	1	2	S+M
2	0	4	S
1	1	32	S+M
0	0	1024	S

Multiply-always algorithm

i	d_i	R_0	*k*	*Operation*
3	1	1	$0 \oplus 1 = 1$	M
3	1	2	$1 \oplus 1 = 0$	M
2	0	4	$0 \oplus 0 = 0$	M
1	1	16	$0 \oplus 1 = 1$	M
1	1	32	$1 \oplus 1 = 0$	M
0	0	1024	$0 \oplus 0 = 0$	M

As can be seen in the tables, the operations in the first algorithm are not balanced which makes the algorithm vulnerable to attacks. However, the multiply-always algorithm is balanced.

b) Differential Power Analysis (DPA)

As introduced by Kocher in 1999, DPA is another power analysis attack method like Simple Power Analysis. As the name suggests, SPA is a *simple* power analysis meaning that it can be used only in single-purpose systems like smart cards. SPA cannot be used in multi-purpose systems such as modern computers today because they run lots of signals in parallel. In contrast, DPA aims to attack such systems as well.

DPA is based on a statistical analysis of the comparison between the power consumption of non-cryptographic operations and cryptographic operations. The goal is here to subtract the noise from the power spectrum to reach the plain cryptographic signal.

In their paper “Power Analysis of FPGAs: How practical is the attack?”, Standaert et al. (2003, p. 8) compared the power signal of exponentiation using a known exponent to an unknown exponent in the Square-and-Multiply algorithm. They start the technique by exponentiating L random values using known and unknown exponents, $S_i[j]$ and $P_i[j]$ respectively. Then, they calculate D[j] using the following equation:

$$D[j] = 1/L \cdot \sum Si[j] – 1/L \cdot \sum Pi[j]$$

They follow the simple “Multiple-Exponent, Single-Data” mode, guess the secret exponent starting from the most significant bit, and check its correctness by making D[j] = 0. They were successful at inferring the secret exponent of RSA on the field-programmable gate array (FPGA).

Kocher suggests three countermeasures against DPA (1999, p. 8). The first one is to reduce the signal size by using algorithms that leak less information in their power consumption, balancing the weights and transitions, physically protecting the device by shielding. The second approach is adding noise to the signal or randomizing the order of the executions. The last one is applying nonlinear key update procedures such as hashing the key.

c) Timing attacks

Timing attacks are based on the fact that operations in cryptographic algorithms take different times given different inputs. One can obtain private information by measuring these time variations during the operations carefully and then deducing the secret key using statistical methods.

Kocher introduced timing attacks in 1996. In his paper, he explains that the secret can be found using the property of the simple modular exponentiation algorithm as shown below. According to the algorithm, the operations to be done are based on the bit of the secret. If the bit is 1, it takes more time compared to bit 0 since the modular multiplication operation is done when the bit is 1. This time difference measurement allows the attacker to guess the next bit b if s/he knows the previous b-1 bits.

Let $s_0 = 1$.
For k = 0 up to w-1:
    If (bit k of x) is 1 then
        Let $R_k = (s_k \cdot y) \bmod n$.
    Else
        Let $R_k = s_k$.

    Let $s_{k+1} = R^2_k \bmod n$.
End For.
Return (R_{w-1}).

As explained by Kocher (1999), the attack method has also an error detection property. The probabilities of each likely exponent are kept, and the attack is continued based on the most likely exponent. If it is incorrect, its probability is decreased.

Kocher suggests three countermeasures against timing attacks. The first one is to set the amount of time to a fixed value for all the operations. However, this is not good against power consumption attacks since the power usage will be decreased significantly while waiting.

Another approach is to add random delays to processing time so that timing measurements are inaccurate. This approach increases the required number of samples however it is still possible to collect more samples to some extent.

The last solution expressed by both Kocher and Zhou & Feng is to do a blinding transformation on the parameters before the operation and unblinding after the operation by random values to make the operations independent of the input. A random v_f coprime to n is chosen and vi is calculated as $v_i = v_f^{-e} \bmod n$. Before the operation, input is multiplied by v_i (modn) and the output is multiplied by v_f (modn) after the operation.

Further details about the aforementioned attacks and other attack methods such as fault attacks, electromagnetic, error message attacks can be found in Zhou&Feng's paper in 2005.

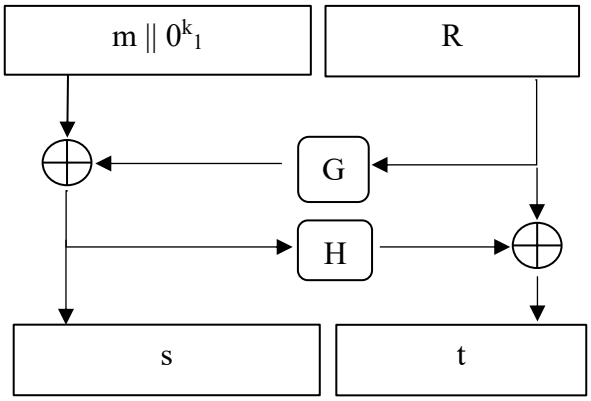
RSA-OAEP (Optimal Asymmetric Encryption Padding)

As it can be seen in the encryption process of RSA, the algorithm is deterministic since it does not include any randomness. This means that the same plaintexts will result in the same ciphertext which makes RSA vulnerable to chosen-plaintext attacks.

Bellare & Rogaway introduced RSA-OAEP in 1995 to include randomness in the algorithm.

The schema below explains the general idea of how OAEP works. Two rounds of Feistel networks are applied. In the end, s||t is encrypted using classic RSA.

m: the message to be encrypted, length is n-k0-k1
R: random k0 bit string $\{0,1\}^{k0}$
G: hash function $\{0,1\}^{k0} \rightarrow \{0,1\}^{k-k0}$
H: hash function $\{0,1\}^{k-k0} \rightarrow \{0,1\}^{k0}$



FUTURE OF RSA

RSA is being widely used in many areas such as banking, e-commerce, TLS although it is not proven to be secure. Today's technology is not enough to break the RSA with large enough keys, how about the future?

In 1994, American mathematician Peter Shor introduced an algorithm called Shor's algorithm which increased the interest in quantum computers rapidly. With this algorithm, it is possible to break the RSA by factoring N in a reasonable time.

How does Shor's Algorithm manage to factor large integers?

Shor's algorithm has two main parts, the classical part and the advanced part which requires quantum computers.

Steps for finding the factors of N in RSA:

1) Choose a random number k between 1 and N.
2) If gcd(N, k) ≠ 1, this means that k is a factor of N, the algorithm ends here.
3) Find the period r of the modular exponentiation $r = k \pmod N$.
4) If r is odd, repeat the same steps with a different k value.
5) Using the identity $a^2 - b^2 = (a-b)(a+b)$:
   $k^r - 1 = (k^{r/2} - 1)(k^{r/2} + 1)$
6) If $k^{r/2} \pmod N = -1$, this means the factors are trivial meaning that they are N and 1. Repeat the steps with a different k value.
   Else, one factor of N is $\gcd(k^{r/2} - 1, N)$ and the other is $\gcd(k^{r/2} + 1, N)$ using Euclidean algorithm.

Steps 1, 2, 4, 5, and 6 can be done in classical computers quickly, however classical computers are quite slow for Step 3. That's where quantum computers are needed, and Shor's algorithm is used. "Shor's algorithm factors an RSA public key n almost as quickly as the legitimate RSA user can decrypt" (Bernstein et al., 2017, p.2).

Shor's algorithm works based on the Central Fourier Transform. The largest integer that has been factored with Shor's algorithm using qubits is 21. It may not be seen as a threat for the time being however many researchers and scientists supported by big technology companies are working on quantum computers.

Bernstein et al. try to find an answer to the question "Is it actually true that quantum computers will kill RSA?" in their paper.

CONCLUSION

To conclude, many attacks have been developed to infer the secret key, however, RSA is still secure and being used in many areas thanks to its 2048-bit long key which prevents integer factorization algorithms and developed algorithms against side-channel attacks. Also, RSA-OAEP provides randomization which changes RSA from being deterministic to randomized.

The future of quantum computers is a matter of curiosity. The invention of such computers with enough qubits may change not only the future of RSA but all public-key cryptosystems since it can crack these algorithms in a very short amount of time. Therefore, further research might be done about post-quantum RSA and different types of side-channel attacks.

REFERENCES

Bellare, M., Rogaway, P. (1995). Optimal Asymmetric Encryption How to Encrypt with RSA. https://cseweb.ucsd.edu/~mihir/papers/oaep.pdf

Bernstein, D. J., Heninger, N., Lou, P., Valenta, L. (2017). Post-quantum RSA. https://cr.yp.to/papers/pqrsa-20170419.pdf

Chevallier-Mames, B., Ciet, M., Joye, M. (2004). Low-Cost Solutions for Preventing Simple Side-Channel Analysis: Side-Channel Atomicity. *IEEE Transactions on Computers, Vol. 53, No. 6.* https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.457.2254&rep=rep1&type=pdf

Courrege, J., Feix, B., Rousselet, M. (2010). Simple Power Analysis on Exponentiation Revisited. *IFIP International Federation for Information Processing.* https://link.springer.com/content/pdf/10.1007%2F978-3-642-12510-2_6.pdf

Kocher, P. (1996). Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems. https://paulkocher.com/doc/TimingAttacks.pdf

Kocher, P., Jaffe, J., Jun, B. (1994). Differential Power Analysis. https://www.paulkocher.com/doc/DifferentialPowerAnalysis.pdf

Rivest, R. L., Shamir, A., Adleman, L. (1977). A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. https://people.csail.mit.edu/rivest/Rsapaper.pdf

Standaert, F., Oldenzeer, L., Quisquater, J. (2003). Power Analysis of FPGAs: How possible is the Attack? https://www.researchgate.net/publication/220760742_Power_Analysis_of_FPGAs_How_Practical_is_the_Attack

Zhou, Y., Feng, D. (2005). Side-Channel Attacks: Ten Years After Its Publication and the Impacts on Cryptographic Module Security Testing. *National Natural Science Foundation of P.R. China.* https://csrc.nist.gov/csrc/media/events/physical-security-testing-workshop/documents/papers/physecpaper19.pdf