A Report on

Cryptoware: Rabin PKC (512 bits)

(Mini-Project of CSE1007- Introduction to Cryptography)

Submitted by

Aman Rangapur

(Registration. No 18BCN7075)

on

12 November 2020

School of Computer Science and Engineering

VIT-AP University, Andhra Pradesh

Abstract

The modular square root problem has a special property of the having computational equivalent to a well-known hard mathematical problem namely integer factorization problem. The Rabin scheme used in public-key cryptosystem is here revisited with a focus limited to a few specific open issues. In particular, message decryption requires one out of four roots of a quadratic equation in a residue ring to be chosen, and a longstanding problem is to identify unambiguously and deterministically the encrypted message at the decryption side by adding the minimum number of extra bits to the cipher-text. The proposed Rabin-p Key Encapsulation Mechanism is built upon the said problem as its source of security, aiming for efficient and practical modular square root-based cryptosystem of which accompanied with the following properties:

1. Improves the performance without plaintext padding mechanisms or sending extra bits during encryption and decryption processes.
2. The plaintext is uniquely decrypted without decryption failure.
3. Improve decryption efficiency by using only one modular exponentiation.
4. A decryption key using a single prime number.
5. It has sufficient large plaintext space.

1 Introduction

Rabin Cryptosystem is an public-key cryptosystem invented by Michael Rabin. It uses asymmetric key encryption for communicating between two parties and encrypting the message.

The Rabin encryption scheme is one of an existing workable asymmetric cryptosystem that comes with nice cryptographic properties. For instance, it has low-cost encryption of which the Rabin encryption is relatively fast to encrypt compared to the widely commercialized RSA cryptosystem, and it has been proven to be as difficult as the integer factorization problem. On the other hand, the decryption of Rabin's scheme produces four possible answers, which only one is correct. This four-to-one decryption setting of the Rabin decryption could lead to a decryption failure scenario since no indicator for selecting the correct plaintext is given.

We revisit the Rabin cryptosystem and then aspire to furnish a new design aiming for efficient, secure and practical Rabin-like cryptosystem.

1.1 About the project

The security of Rabin cryptosystem is related to the difficulty of factorization. It has the advantage over the others that the problem on which it banks has proved to be hard as integer factorization. In our design, we use the modulus $N = p^2q$ and we restrict the plaintext to be less than p^2. Hence, to decrypt correctly, it suffices to apply an efficient algorithm that solves the square root of quadratic congruence modulo p instead of modulo $N = p^2q$.

We will also show the real world implementation of Rabin 512 bits cyptosystem.

1.2 Implementation Environment

My primary programming language is Python, so I will make use of the Rabin Algorithm for making an encryption and decryption program in Python. We are also going to use few python libraries to make code easy as we have large mathematical operations. There are three inputs for Rabin 512 encryption and decryption 1 output for each of them respectively.

2 Procedure

2.1 Rabin Key Generation Algorithm

The key generation algorithm then produces an integer N as a product $N = p^2q$, which is denoted as the public key. The private key is the prime p.

Algorithm 1 Rabin Key Generation Algorithm

Input: The size k of the security parameter.

Output: The public key $N = p^2q$ and the private key. p

1: Choose two random and distinct primes p and q such that $2k < p, q < 2^{k+1}$ satisfy $p, q \equiv 3 \pmod 4$.

2: Compute $N = p^2q$.

3: Return the public key N and the private key p.

2.2 Rabin Encryption Algorithm

To encrypt a plaintext, the Rabin-p encryption algorithm with the public key N does the following.

Algorithm 2 Rabin Encryption Algorithm

Input: The public key N

Output: A ciphertext c

1: Choose plaintext $0 < m < 2^{2k-1}$ such that $\gcd(m, N) = 1$.

2: Compute $c \equiv m^2 \pmod N$).

3: Return the ciphertext c.

Remark 1. *The encryption algorithm (Algorithm 2) takes the plaintext* $m < 2^{2k-1}$ *and compute* $c \equiv m^2 \pmod{N}$*. We observe that the plaintext* m *is restricted to the range of* $m < 2^{2k-1} = (2^{2k})/2 < p^2/2 < p^2$*.The output is the ciphertext* c*.*

2.3 Rabin Decryption Algorithm

To decrypt a ciphertext, the Rabin-p decryption algorithm with the private key p does the following.

Algorithm 3 Rabin Decryption Algorithm

Input: A ciphertext c and the private key p

Output: The plaintext m

1: Compute $w \equiv c \pmod{p}$.

2: Compute $m(p) \equiv w^{(p+1)/4} \pmod{p}$.

3: Compute $i = (c - m(p)^2)/p$.

4: Compute $j \equiv i/(2 * m(p)) \pmod{p}$.

5: Compute $m(1) = m(p) + jp$.

6: If $m(1) < 2^{2k-1}$, then return $m = m(1)$. Else, return $m = p^2 - m(1)$.

Remark 2. *We observe that the decryption algorithm needs only a single prime number as its key. Hence, only one modular exponentiation is taking place during the decryption process. Such computational advantage would positively affect the overall operations.*

Remark 3. *We see that since our proposed scheme does not need to carry out any CRT computation, thus the Novak's attack is not applicable on the Rabin-p cryptosystem (i.e. resilient against Novak's attack).*

2.4 Proof for Rabin Decryption

Lemma 1. Let p be a prime number such that $p \equiv 3 \pmod 4$ and c be an integer such that $\gcd(c, p) = 1$. The congruence $c \equiv m^2 \pmod p$ has either no solutions or exactly two solutions. If $m(1)$ is a solution, then $-m(1) \pmod p$ is the other solution.

Lemma 2. Let p be a prime number such that $p \equiv 3 \pmod 4$ and c be an integer such that $\gcd(c, p) = 1$. The congruence $c \equiv m^2 \pmod{p^2}$ has exactly two solutions if $c \equiv m^2 \pmod p$ has exactly two solutions.

Proof. Suppose that $c \equiv m^2 \pmod{p^2}$ has a solution $m(1)$. Then any other solution of m is such that $p^2 \mid m^2 - m^2(1)$, that is $p^2 \mid (m + m(1)) * (m - m(1))$. Hence, we consider $p \mid (m + m(1))(m - m(1))$ as well. If $p \mid (m + m(1))$ and $p \mid (m - m(1))$, then p would divide $(m + m(1)) + (mm(1)) = 2m$ and $(m + m(1)) - (m - m(1)) = 2m(1)$. Since $p \equiv 3 \pmod 4$ is an odd prime, then $p2$ so, p would divide both m and $m(1)$. Consider $c \equiv m^2 \pmod{p^2}$. If $p \mid mthenp \mid m2$ therefore $p \mid c$, however $\gcd(c, p) = 1$. Therefore pc, hence pm. The same goes for $p \mid m(1)$. Now, consider in the case if $p \mid (m + m(1))$ or $p \mid (m - m(1))$ but not both. Since $p^2 \mid (m + m(1)) * (m - m(1))$, therefore either $p^2 \mid (m + m(1))$ or $p^2 \mid (m - m(1))$ implies that $m \equiv \pm m(1) \pmod{p^2}$. Since $-m(1) \pmod{p^2}$ is always a solution when $m(1) \pmod{p^2}$ is a solution, therefore the congruence has exactly two solutions.

Lemma 3. If $m(1)$ and $m(2)$ are the two distinct integers solution for $c \equiv m(2) \pmod{p^2}$, then $m(1) + m(2) = p^2$.

Proof. Suppose $m(1) \neq m(2) \pmod{p^2}$ such that $m^2(1) \equiv m^2(2) \equiv c \pmod{p^2}$. Observe that, if $m(1)$ is a solution for $c \equiv m(2) \pmod{p^2}$, then $m(2) \equiv -m(1) \pmod{p^2}$ is also a solution. Thus, $m(2) \equiv -m(1) \pmod{p^2}$ can be reinterpreted as $m(2) = p^2 - m(1)$. Hence $m(1) + m(2) = p^2$.

2.5 General Toy Example of Rabin PKC (took p and q smaller primes for better understanding)

Implementation of Rabin 512 are shown in **Results section**

Suppose we have two communicating parties, namely Bob as the sender of a message and Alice as its corresponding receiver. Let the security parameter $k = 15$.

Key generation:

Alice generate two distinct primes $p = 32779, q = 40829$.

1. Compute $N = p^2q = 43869243335189$.
2. Alice keeps her private key p.
3. Alice publish her public key N.

Encryption:

Bob receives Alice's public key. He would like to send a message $m = 479571937$.

1. Compute $c \equiv m^2 \pmod{N} = 26669194871231$.
2. Bob send c to Alice as his ciphertext.

Decryption:

Alice receives a ciphertext $c = 26669194871231$ from Bob. To decrypt c, Alice then executes:

1. Compute $w \equiv c \pmod{p} = 27646$.
2. Compute $m(p) \equiv c^{(p+1)/4}(modp) = 15167$.
3. Compute $i = (c - m^2(p))/p = 24318$.
4. Compute $j \equiv [i/(2m(p))] \pmod{p} = 14630$.
5. Compute $m(1) = m(p) + jp = 479571937$.
6. Since the integer $m(1) < 2^{2k-1}$, then return the plaintext $m = m(1)$.

2.6 Reduction to Factoring $N = p^2q$

In this section, we show that if there exists an algorithm that can decrypt message m from any random Rabin-p ciphertext, then such algorithm also be able to factor $N = p^2q$. We observe the following.

Theorem. Let $N = p^2q, m < 2^{2k-1}$ and $2^{2k-1} < m < p2$ such that $m+m = p$. Then $\gcd(m + m, N) = p^2$.

Proof. Suppose $2^k < p < 2^{k+1}$, then $2^{2k} < p^2 < 2^{2k+2}, and 2^{2k-1} < p^2/2 < 2^{2k+1}$. Suppose $m < 2^{2k-1}$, then there exists another integer $m > 2^{2k-1}$ such that $m + m = p^2$. Thus this implies $p^2 - m = m < 2^{2k-1}$. Now, we determine the range of the m such that $p^2m < 2^{2k-1}$. Then we obtain the lower bound for m, of which

$$m > p^2 - 2^{2k-1}$$
$$m > 2^{2k} - 2^{2k-1}$$
$$m > 2^{2k-1}$$

and upper bounded by $m < p^2$. Take the $\gcd(m + m, N)$, then we obtain p^2. Hence $q = N/p^2$.

2.7 Resistant to Novak's Attack

In general, the decryption algorithm of a Rabin-like cryptosystem consists of two parts. The first part is for the modular exponentiation operation of which in order to obtain the message in the form of m modulo p and m modulo q from its corresponding ciphertext c. The second part then would be the recombination process using the Chinese Remainder Theorem (CRT) algorithm to recover the proper message m. Most side channel attacks deal with the first part. For instance, the work by earlier which uses the timing attack approach or it enables side channel attack using the power analysis approach.

Alternatively, we proposed a very efficient side channel attack upon the CRT computation (i.e. the second part of the Rabin-p decryption). We observe that all variants of the Rabin-p cryptosystem (except Rabin-Williams scheme) involves a process that hardly

depends on the CRT or Garner's algorithm (i.e. the process to recover all the modulo square roots). Therefore, Novak's attack is indeed applicable for such computation, of which can result in the insecurity of the cryptosystems.

Remark. *We reason that since our proposed scheme does not need to carry out any CRT computation, thus the Novak's attack is not applicable on the Rabin-p cryptosystem (i.e. resilient against Novak's attack).*

2.8 Resistant to Chosen Cipher text Attack

Notice that the factoring mentioned by the Algorithm 4 could provide a way to launch a chosen ciphertext attack upon the proposed scheme in polynomial time, hence resulting in the system totally insecure in this sense. Therefore, to provide security against this kind of attack, we need to implement a Key Encapsulation Mechanism.

3 Code:

Here is the minor snippet of code implemented in Python for Encryption and Decryption.

```
    def encryption(plaintext, n):

    plaintext = padding(plaintext)
return plaintext ** 2 % n
def padding(plaintext):

    binary_str = bin(plaintext)
output = binary_str + binary_str[-16:]
return int(output, 2)

    def decryption(a, p, q):
n = p * q
r, s = 0, 0
```

```
if p % 4 == 3:

    r = prime.sqrt_p_3_mod_4(a, p)
elif p % 8 == 5:

    r = prime.sqrt_p_5_mod_8(a, p)
if q % 4 == 3:

    s = prime.sqrt__3_mod_4(a, q)
elif q % 8 == 5:

    s = prime.sqrt_p_5_mod_8(a, q)
gcd, c, d = prime.egcd(p, q)
x = (r * d * q + s * c * p) % n
y = (r * d * q - s * c * p) % n
lst = [x, n - x, y, n - y]
lst = [x, (n - x), y, (n - y)]
plaintext = choose(lst)
string = bin(plaintext)
string = string[:-16]
plaintext = int(string, 2)

    return plaintext

    def delete_space(string):
lst = string.split(' ')
output = ''
for i in lst:
output += i
return output

    def add_space(string):
```

```
string = string[::-1]
string = ' '.join(string[i:i + 8] for i in range(0, len(string), 8))
return string[::-1]

    #for encryption
p = int(delete_space(input('p = ')), 32)
q = int(delete_space(input('q = ')), 32)
n = p*q
#print('n = pq =', add_space(format(n, 'x')))
plaintext = int(delete_space(input('Plaintext = ')), 32)
ciphertext = Rabin.encryption(plaintext, n)
print('Ciphertext =', add_space(format(ciphertext, 'x')))

    #for decryption
ciphertext = int(delete_space(input('Ciphertext = ')), 32)
p = int(delete_space(input('p = ')), 32)
q = int(delete_space(input('q = ')), 32)
plaintext = Rabin.decryption(ciphertext, p, q)
print('Plaintext =', add_space(format(plaintext, 'x').zfill(512 // 4)))
```

4 Results

4.1 Output of Encryption:

Here is the output:

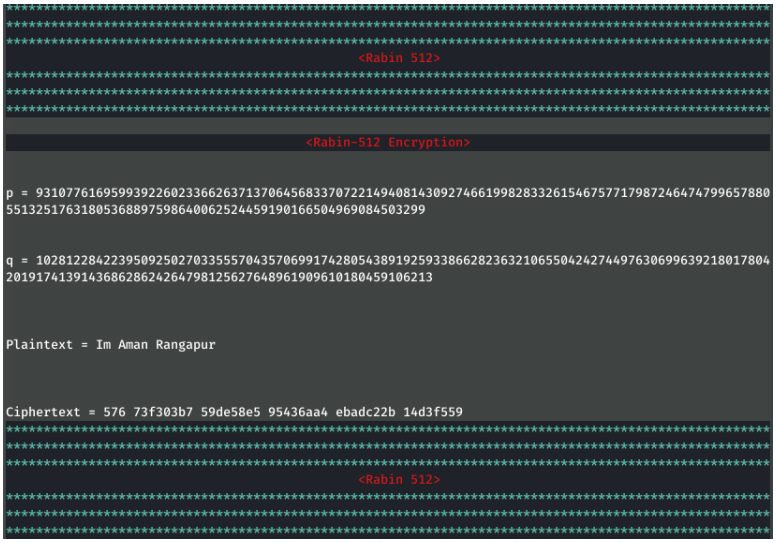

Fig 1

4.2 Output of Decryption:

Here is the output:

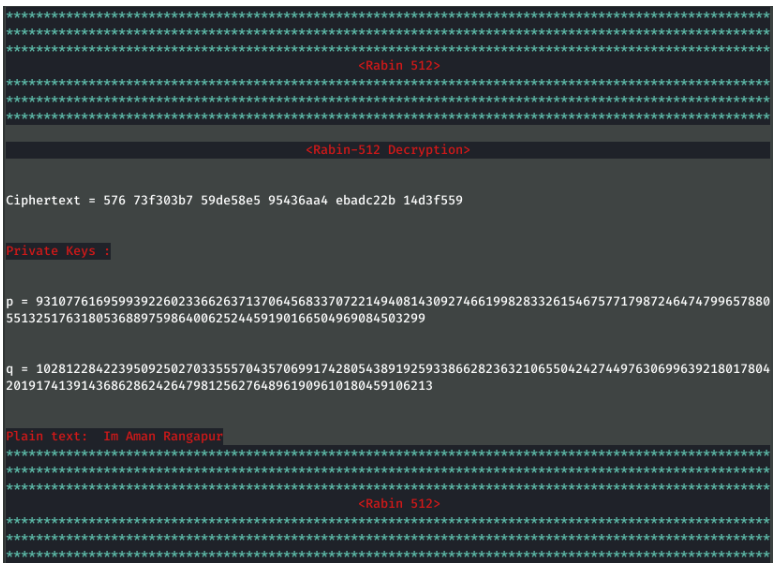

Fig 2

Outputs of sender and receiver:

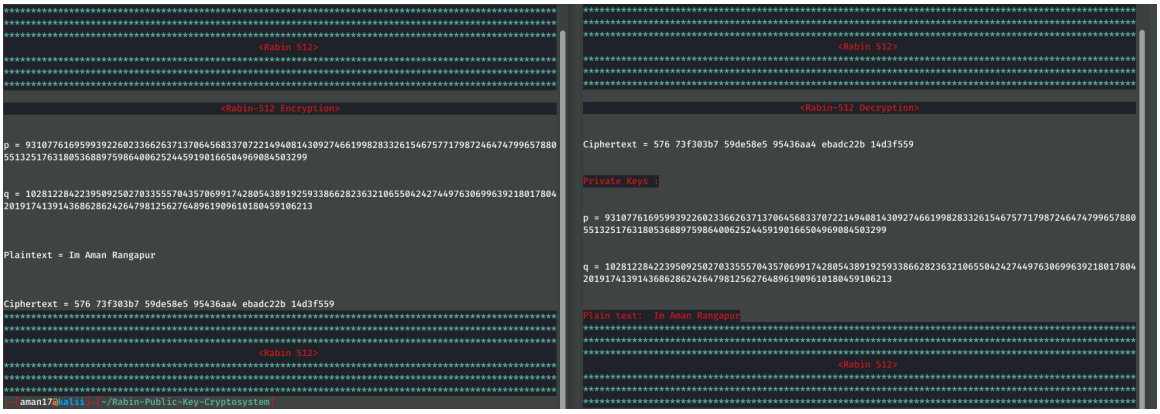

Fig 3

In figure 1, the sender enters the private keys p and q and he enters the Plain text he wants to send. He gets the output as cipher text and passes the cipher text to receiver.

In figure2, the receiver enters the cipher text along with the private keys p and q and gets the plain text as output.

5 References

[1] Behrouz A Forouzan, Debdeep Mukhopadhyay, "Cryptography and Network Security", Mc Graw Hill, Third Edition, 2015.

[2] Rabin Cryptosystem with Implementation GeeksforGeeks. https://www.geeksforgeeks.org/rabin-cryptosystem-with-implementation.

[3] William Stallings, "Cryptography and Network Security: Principles and Practice", Pearson Education, Seventh Edition, 2017.

[4] Overleaf: a collaborative cloud-based LaTeX editor used for writing, editing and publishing scientific documents. http://www.overleaf.com.

[5] The Laws of Cryptography: Rabin's Version by Neal R. Wagner. http://www.cs.utsa.edu/ wagner/laws/Rabin.html.

[6] Rabin's Cryptosystem. https://programmingpraxis.com/2011/11/22/rabins-cryptosystem/