2022

RSA encryption: Creating and breaking keys

HRITHIK ADHIKARY

ANALYSIS PAPER

MELBOURNE, VICTORIA | Information Technology

Contents

Introduction

Cryptography refers to securing and protecting information and communication using formulas and concepts derived from mathematics and set of rule-based procedures called algorithms. The cryptographic algorithms are used in key generation, key exchange, and digital signatures to protect sensitive data in network and communication systems.

Two types of encryption algorithms are as follows:

Symmetric key algorithm

Symmetric key is a encryption methodology where a single key is used to encrypt and decrypt a piece of information. The encryption key 'e' is random bit string which generates a cipher text C performing mathematical calculation with message M. To decrypt the cypher text C, same key e is used to perform calculations with cypher text C which generates a decrypted original message M.

Examples of symmetric algorithms are as follows:

- RC6
- AES
- DES

Asymmetric key algorithm:

Asymmetric key encryption (public key cryptography) uses key pairs that are mathematically interrelated. The key pair are known as public key and private key. The public key is used to encrypt the message to receive a cypher text, which is sent to receiver via communication media. Afterwards, receiver uses private key which is mathematically related to public key to decrypt the cypher text and to retrieve original message.

Examples of asymmetric encryption method are as follows:

- RSA encryption
- Elliptic curve digital signature algorithm
- Digital signature algorithm

Rivest Shamir Adleman encryption (RSA)

RSA is a asymmetric encryption algorithm which is widely used for secure data transfer. The encryption and decryption methodology are based on two large random prime numbers which are kept secret. Message once encrypted using public key 'e' can only be decrypted using relative prime number 'd.'

The report is focused on explaining RSA encryption, its working mechanisms and performing algorithmic attack to break a simple RSA encryption .

Aim

The aim of this assignment is to provide detailed information about how message is encrypted and decrypted using RSA algorithm. A student ID is encrypted using a public key which generates cypher text. The resulting cypher text is decrypted with recovered private key to retrieve original student ID thus aiding our understanding of how RSA algorithm works.

The loopholes in RSA algorithm which can be used to break RSA algorithm is described providing various techniques to break RSA algorithm. Effectiveness and success rate of each technique is described pointing out the vulnerability of RSA algorithm.

Methods

Part 1: make the keys

Generating primes

The first step of RSA algorithm is to generate two large prime numbers P and Q. The prime number chosen were Pythagorean prime numbers of form 4n+1. Pythagorean primes are odd prime numbers that are exactly sum of two perfect squares.

Using student ID 1486987, two primes were generated using next prime() function in wolfram alpha which resulted to 1487009 as first prime number(P) and 1487053 as second prime number (Q)

Generating Large Composite Number N

N is a composite number generated by multiplication of two prime numbers.

$N = P * Q$

As for our solution,

N = 1487009*1487053

N= 2211261194477

Totient ϕ(n)

After generating N, totient is an essential element in RSA encryption. Totient is generated by multiplication of two numbers retrieved by subtracting both prime numbers P and Q by 1.

Mathematically,

$\phi(n) = (P-1) * (Q-1)$

Totient of 2211258220416 was generated plugging in the two primes.

Encryption exponent

Encryption exponent is denoted by ‘e’ is a prime number which is used to encrypt the message in RSA algorithm. Anyone can have access to public key in RSA encryption.

There are certain requirements that needs to be met while choosing e. They are as follows:

- Encryption exponent should be a prime number
- Encryption exponent should be coprime with composite number (N).

Mathematically expressing,

{ 1<e< ϕ(n)} AND { GCD(N,e)= 1} (Overmars, 2019)

There is no requirement for e to be a large number, so prime number chosen for e are usually 13, 17 or 65537 if they pass the above requirements.

17 is chosen as Encryption exponent.

Decryption exponent

Decryption exponent is a prime number denoted by 'd' which is used as a part of private key to decrypt a cypher text generated by using public key. Decryption exponent is mathematically related to encryption exponent and is retrieved by performing extended Euclidean algorithm to e and ϕ(n)

Multiplication of e and d e*d modulus ϕ(n) should be equivalent to 1.

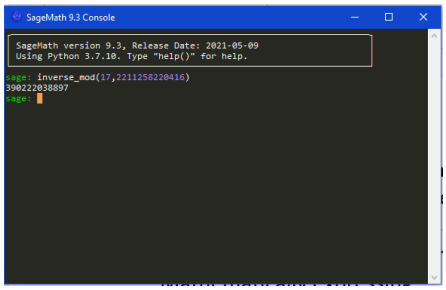
Mathematically expressing,

{mod(e*d, ϕ(n)=1} (Overmars, 2019)

For our problem, Mod Inverse function is used between e and ϕ(n) to generate d.

While e = 17 , ϕ(n)=2211258220416

d = 390222038897



Public key

Public key (N,e) is used to encrypt the message M which results in Cypher text C. Public key can be kept in public directory and does not need to be hidden.

(17, 2211261194477)

Private Key

Private key (N,d) remains hidden and is only accessed by owner. Private key is used to decrypt he cypher text and to retrieve a original message.

Pr(390222038897, 534326705170)

Encryption

Message is encrypted using public key (N,e) using following formula:

C = M^e MOD N

If alice needs to send message to Bob, Alice will have access to Bob's public key (N,e) from public directory which is not hidden. Cypher text is generated as per message and Bob's public key and is sent to Bob.

```
sage: mod(1486987^17,2211261194477)
534326705170
sage:
```

Decryption

Cypher text retrieved using public key is decrypted using private key(N,d) using following formula:

M = C^d MOD N

Once Bob receives a cypher text C, it is decrypted using Bob's private key (N,d) which retrieves original message M . Only Bob has access to his private key and is hidden and kept secret.

Breaking the keys

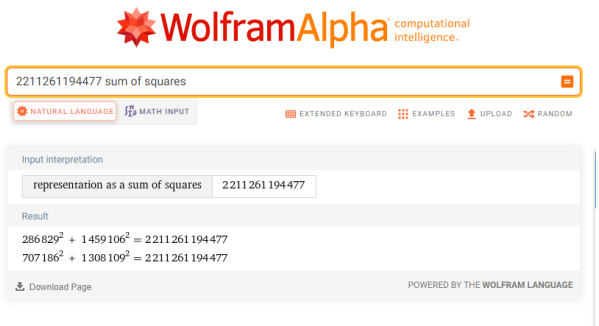
Euler's factorization method

Pythagorean primes in form P= 4x+1 can be expressed as P = a^2+b^2

Original Pythagorean primes can be generated by using Euler's factorization method if composite number N is known. (Overmars, 2019)

Consider, Composite number N= 2211261194477 which is generated by using multiplications of two Pythagorean primes P and Q.

Using sum of squares in wolfram alpha,



286829^2 + 1459106^2= 707186^2 + 1308109^2 are in a form a^2+b^2 = c^2+d^2

Combining even and odds results in d^2-a^2 = b^2- c^2

1308109^2-286829^2 = 1459106^2-707186^2

Factorizing form x^2-y^2 to (x-y)(x +y)

(1594938) (1021280) = (2166292) (751920)

Performing GCD

GCD = (D+A)(B+C)/2 = 1103

= GCD((D-A),(B+C))/2= 982

= GCD((D+A),(B-C)/2 = 723

= GCD((D-A),(B-C)/2 =520

Now to generate P we must use formula GCD((D-A),(B-C)/2) + (D+A)(B+C)/2

1103^2 + 520^2

= 1487009

P was successfully retrieved.

Now to generate Q we must use formula GCD((D+A),(B-C)/2) + GCD((D-A),(B+C))/2

= 723^2 +982^2

= 1487053

Q was successfully retrieved.

Breaking keys using Fermat algorithm if N is known

Gaussian primes in form 4x+3 can be generated if composite number N is known by using Fermat algorithm. (Overmars, 2019)

N = 2211421797029

As per Fermat algorithm a composite number resulting through multiples of two prime numbers a*b is difference of squares of those prime numbers a^2-b^2.

Mathematically,

N = a^2-b^2 = (a-b)(a+b)

b^2 = a^2-N

Initially, prime a need to be picked which can be square root of N (rounded up to integer number if the square root is decimal).

a = [square root(N)]

a = 1487085

Now to find prime b,

To find b, the resulting number from a^2-N should have a integer square value. If the resulting square value does not have a integer square value, then the number is discarded. Prime number a is incremented by 1 till the number is found.

b^2 = {x∈Z+:b =x*x }

If the result does not have square value, then, {a=a+1}

b^2 = a^2-N^2

if (Squareroot(b^2) is not a integer value) Then {Increment a by 1 (a=a+1)}

a^2 – N is a square number.

b^2 = (a^2 – N2)

b^2 = ((Δa1)^2 – N2) = 1487085^2-2211421797029

b^2= 2211421797225 – 2211421797029

b^2= 196 {196 has a integer square root}

b= Sqrt(196)

b= 14

Finding prime number P3 and P4 from generated a and b

P3 = a-b

= 1487085 – 14

= 1487071

P3 = a+b

= 1487085 + 14

= 1487099

P3=1487071 and P3= 1487099 matches two primes generated earlier. Hence original prime numbers can be generated once N2 is known using Fermat factorization method.

Breaking keys using quadratic equation

Original primes P and Q can be recovered if totient and N are known by using general form of quadratic equation.

N = 2211261194477

Totient ϕ(n) = 2211258220416

ϕ(n) = (P-1) * (Q-1) {expressing in simpler form}

= PQ – P- Q+1 {N = P*Q}

= N – P – Q + 1

Express Primes in terms of N and expressing in quadratic equation results to:

N = 2211261194477 ϕ(n)= 2211258220416

P1,P2 = (2974062+- SQRT((-2974062)^2- 8845044777908))/2

=1487031 +- 22

Solving with positive,

P1 = 1487009.

Solving with negative

P2= 1487053

Retrieved P = 1487009 Q= 1487053 matches 2 prime numbers generated earlier. Hence, P and Q is successfully recovered once N and ϕ(n) are known.

Generating private key using public key using wiener's attack

Decryption exponent can be recovered if public key Pu(N,e) is known with $e\ d \equiv 1\ mod\ \phi$ n. The method implements continuous fraction to private key d. (Overmars, 2019)

PU(2211261194477,17)

Finding continuous fraction of 17/2211261194477/. The results are as follows:

```
sage: a = (17/2211261194477).continued_fraction()
sage: a
[0; 130074187910, 2, 2, 3]
sage: a.convergents()
[0, 1/130074187910, 2/260148375821, 5/650370939552, 17/2211261194477]
sage:
```

As a requirement, d needs to be an odd number so all the even numbers on denominator are discarded. : 2/26014837582 is only possible fraction which is expressed in ϕn = ed-1/k form.

While calculating the totient, resulting number should not be an integer value. However, in the given scenario, the number is a decimal number. So, the number is discarded.

```
sage: ((17*26014837582)-1)/2
442252238893/2
sage:
```

Hence, testing all the generated continuous fractions and expressing them in form of ϕn = ed-1/k none of the fractions generated a integer totient failing to generate a required solution. Decryption exponents could not be retrieved using wieners attack with provided public key PU(2211261194477,17).

An essential criterion for wiener attack to generate a correct solution d needs to be a small number. Contrarily, encryption exponent is a small number whereas decryption exponent is a large number in the scenario we are provided. It can be concluded that correct solution was not generated because the scenario did not the requirement of wiener attack to have a small decryption exponent (Overmars, 2019).

As an alternative, the encryption exponent and decryption exponents are interchanged such that e is a large number and d is a small number so that criteria for wiener's attack is met.

Using PU(2211261194477,390222038897) as public keys

Generating continuous fraction of 390222038897/2211261194477

```
sage: CF = continued_fraction(390222038897/2211261194477)
sage: CF
[0; 5, 1, 2, 14578, 2, 1, 1, 2, 1, 967, 2, 1, 1, 10, 1, 3, 2]
sage: CF.convergents()
[0,
 1/5,
 1/6,
 3/17,
 43735/247832,
 87473/495681,
 131208/743513,
 218681/1239194,
 568570/3221901,
 787251/4461095,
 761840287/4317100766,
 1524467825/8638662627,
 2286308112/12955763393,
 3810775937/21594426020,
 40394067482/228900023593,
 44204843419/250494449613,
 173008597739/980383372432,
 390222038897/2211261194477]
```

Discarding fractions with even numbers as denominator and expressing in ϕn = ed-1/k.

Expressing 1/5 in ϕn = ed-1/k generates decimals as prime numbers; hence the number is discarded.

Expressing 3/17 in ϕn = ed-1/k generates totient of 2211258220416.

```
sage: ((390222038897*17)-1)/3
2211258220416
sage:
```

The equation X^2-(N- Φn+1)*X+N =0 generated value of x as two primes 1487053 and 1487009 which should retrieve composite N provided above. Performing multiplication of 1487053 and 1487009 generated composite number 2211261194477 which matched public key above.

Hence decryption exponent of 17 retrieved from calculation is an actual private key of PU(2211261194477,390222038897) which was retrieved successfully using wieners attack.

Results

Breaking keys using Euler's factorization

Composite number generated from two Pythagorean prime is successfully factorized applying Euler's factorization method to retrieve two original primes. The process involved finding sum of 2 squares of N and combining odd and even numbers. Afterwards, using common divisor and factorizing, original two primes is retrieved. (Overmars, 2020) (Overmars, 2019)

Breaking keys using Fermat algorithm if N is known

Reason for original primes to be retrieved with ease is because the primes chosen for encryption were close to each other. If the primes were linearly apart from each other the calculation would have been more complex. It would be hard to generate b because a had to go through number of increments and iterations, till the right number is found. It is more memory and CPU incentive.

Breaking keys using quadratic equation

Original primes are recovered expressing primes in terms of N, $\phi_n P_1 = N - \phi_n - P_2 + 1$, $P_2 = N - \phi_n - P_1 + 1$ and using general equation quadratic formula. Two results retrieved from quadratic equation. The first result is first prime P whereas second result is second prime Q. (Overmars, 2019)

Generating private key using public key using wiener's attack

Wiener's attack generates retrieves correct result only if decryption exponent is a small number. Performing wieners attack on public key with small encryption generates totient that does not match the original totient hence proving that the attack was failed. Wiener attack retrieves correct result if the public key matches all its criteria and if e happens to be a large number whereas d a small number. Expressing resulting continuous fraction in form of ϕn = ed-1/k knowing value of e, d and k generates value of totient in the key. The attack was successful because the equation e*d mod ϕn =1 is always true in RSA encryption. As e is already known, generating number that matches the criteria for d and plugging it into equation ϕn = ed-1/k generates a correct result if result is a positive integer. (Overmars, 2019)

Discussion

RSA encryption if smaller primes are used can be cracked easily using various algorithms. However, the number used for RSA are large complex prime numbers which computers with present day specification lack resource to break. As quantum computing is being developed and computers are getting exponentially faster, it may be correct to assume that RSA encryption may not be optimal enough to secure future data. Special considerations should be taken while choosing two primes so that it cannot be retrieved using above methods. Some of the considerations are choosing primes further from each other so that two numbers cannot be traced, and CPU falls out of resource, choosing decryption exponent as a large number so wiener attack cannot be performed and not choosing both primes having similar characteristics such as Pythagorean primes and gaussian primes.

Conclusion

The report discussed two types of encryption method and performed thorough analysis on RSA algorithm. The analysis creating RSA encryption using set of prime numbers and encrypting and decrypting the message. In addition, algorithms to break the RSA encryption has been provided and effectiveness and requirement of each cracking method was explained in detail. At last, discussion about why RSA encryption may not be suitable for future is provided.

References

Boneh, D., n.d. Twenty Years of Attacks on the RSA Cryptosystem. In: *RSA Survey.* s.l.:dabo@cs.stanford.edu, pp. 1-16.

Overmars, A., 2019. In: *Overmars 2019.* s.l.:s.n., pp. 1-22.

Overmars, A., 2020. Overmars Euler Factorization Method. *Research in applied science,* 3(12), pp. 25-30.