JOURNAL OF INFORMATION, KNOWLEDGE AND RESEARCH IN COMPUTER ENGINEERING (ISSN 0975 – 6760, GIF: 01.1135, SIF: 02.502)

Key Challenges on Integer Factorization in RSA Public Key Cryptosystem

Ramesh M Badiger[1] Murthy D.H.R[2] and Ningappa Pujar[1]

[1]Assistant Professor, Tontadarya College of Engineering, Gadag, Karnataka

[2] Senior Project Fellow, CSIR Fourth Paradigm Institute, Bangalore

Corresponding Author: rameshmbadiger@gmail.com

Abstract:

RSA is the well known Public Key Cryptographic Algorithm which is more powerful till today because of its hardness in factoring the public modulus N=P*Q. The algorithm begins by considering two large prime numbers P and Q and all the proceeding steps are based on these numbers. Once we are able to find either of the factors successfully from the modulus N then RSA will no more exists. This paper deals with the study of different algorithms used for factorization and their key features.

1. INTRODUCTION

RSA was designed by R. Rivest, A. Shamir and L. Adleman in 1977. It is used widely in most of the e-commerce applications. The implementation steps for RSA algorithm is as follows.

1 Generate randomly two large primes p, q;
2 Compute $N = pq, \phi(N) = (p-1)(q-1)$;
3 Generate e randomly such that $\gcd(e, \phi(N)) = 1$;
4 Compute d such that $ed \equiv 1 \pmod{\phi(N)}$;
5 Encryption: $m \mapsto m^e \bmod N$;
6 Decryption: $c \mapsto c^d \bmod N$;

2. FACTORIZATION ATTACKS

- Trail Division
- Pollard (P-1)
- Pollard Rho
- Fermat's Factoring method
- Quadratic Sieve Factoring

Trial division is the oldest technique for factorization in which we start dividing the given any composite number by the prime number starting from 2,3,5,6 and so on. Obviously it will take huge time for a big composite numbers. Therefore it cannot be used for real world scenario.

Next well known factoring algorithm is Fermat's factoring algorithm. It is an extremely simple method essentially based on the relation

$x^2 - y^2 = (x - y)(x + y)$.

If we can find y such that $n + y^2 = x^2$ then $(x - y)|n$ and also $(x + y)|n$. Fermat's method works well when the number's factors into two terms of approximately equal size that is factors which are nearer to the square root of the number to be factorized. It works poorly when the factors are of very different sizes.

The Pseudo code for Fermat's Theorem is given as

```
FermatFactor(N): // N should be odd
  a ← ceil(sqrt(N))
  b² ← a*a - N
  while b² is not a square:
  a ← a + 1 // equivalently:
  b² ← b² + 2*a + 1
  b2 ← a*a - N // a ← a + 1
  end while
  return a - sqrt(b²) // or a + sqrt(b²)
```

3. FACTORIZATION

In number theory, integer factorization is the decomposition of a composite number into a product of smaller integers. If these integers are further restricted to prime numbers, the process is called prime factorization. When the numbers are very large, no efficient, non-quantum integer factorization algorithm is known; an effort by several researchers concluded in 2009,

factoring a 232-digit number (RSA-768), utilizing hundreds of machines over a span of two years. However, it has not been proven that no efficient algorithm exists. The presumed difficulty of this problem is at the heart of widely used algorithms in cryptography such as RSA. Many areas of mathematics and computer science have been brought to bear on the problem, including elliptic curves, algebraic number theory, and quantum computing. Not all numbers of a given length are equally hard to factor.

The hardest instances of these problems (for currently known techniques) are semi primes, the product of two prime numbers. When they are both large, for instance more than two thousand bits long, randomly chosen, and about the same size (but not too close, e.g., to avoid efficient factorization by Fermat's factorization method), even the fastest prime factorization algorithms on the fastest computers can take enough time to make the search impractical; that is, as the number of digits of the primes being factored increases, the number of operations required to perform the factorization on any computer increases drastically.

Many cryptographic protocols are based on the difficulty of factoring large composite integers or a related problem—for example, the RSA problem. An algorithm that efficiently factors an arbitrary integer would render RSA-based public-key cryptography insecure.

Special-purpose

A special-purpose factoring algorithm's running time depends on the properties of the number to be factored or on one of its unknown factors (for eg: the size of the factors) etc.

Exactly what the running time depends on varies between algorithms.

Some of the examples are as follows:

- Trial division
- Wheel factorization
- PollardHYPERLINK "https://en.wikipedia.org/wiki/Pollard's_rho_algorithm"'HYPERLINK "https://en.wikipedia.org/wiki/Pollard's_rho_algorithm"s rho algorithm
- FermatHYPERLINK "https://en.wikipedia.org/wiki/Fermat's_factorization_method"'HYPERLINK "https://en.wikipedia.org/wiki/Fermat's_factorization_method"s factorization method
- EulerHYPERLINK "https://en.wikipedia.org/wiki/Euler's_factorization_method"'HYPERLINK "https://en.wikipedia.org/wiki/Euler's_factorization_method"s factorization method
- Special number field sieve

4. TIME COMPLEXITY

In computer science, the time complexity of an algorithm quantifies the amount of time taken by an algorithm to run as a function of the length of the string representing the input . The time complexity of an algorithm is commonly expressed using big O notation, which excludes coefficients and lower order terms. When expressed this way, the time complexity is said to be described asymptotically, i.e., as the input size goes to infinity. For example, if the time required by an algorithm on all inputs of size n is at most 5n3 + 3n for any n (bigger than some n0), the asymptotic time complexity is O(n3).

5. BREAKING RSA:CRYPTANALYSIS

Cryptanalysis refers to the study of ciphers, ciphertext, or cryptosystems (that is, to secret code

systems) with a view to finding weaknesses in them that will permit retrieval of the plaintext from the ciphertext, without necessarily knowing the key or the algorithm.

Cryptanalysis refers to the study of ciphers, ciphertext, or cryptosystems (that is, to secret code systems) with a view to finding weaknesses in them that will permit retrieval of the plaintext from the ciphertext, without necessarily knowing the key or the algorithm. This is known as breaking the cipher, cipher text, or cryptosystem. Breaking is sometimes used interchangeably with weakening.

This refers to finding a property (fault) in the design or implementation of the cipher that reduces the number of keys required in a brute force attack (that is, simply trying every possible key until the correct one is found). For example, assume that a symmetric cipher implementation uses a key length of2^128 bits (2 to the power of 128): this means that a brute force attack would need to try up to all 2^128 possible combinations (rounds) to be certain of finding the correct key (or, on average, 2^127 possible combinations) to convert the ciphertext into plaintext, which is not possible given present and near future computing abilities.

However, a cryptanalysis of the cipher reveals a technique that would allow the plaintext to be found in 2^40 rounds. While not completely broken, the cipher is now much weaker and the plaintext can be found with moderate computing resources. There are numerous techniques for performing cryptanalysis, depending on what access the cryptanalyst has to the plaintext, ciphertext, or other aspects of the cryptosystem. Below are some of the most common types of attacks:

Attacks on RSA

1) **Known-plaintext analysis**: With this procedure, the cryptanalyst has knowledge of a portion of the plaintext from the ciphertext. Using this information, the cryptanalyst attempts to deduce the key used to produce the ciphertext.

2) **Chosen-plaintext analysis (also known as differential cryptanalysis):** The cryptanalyst is able to have any plaintext encrypted with a key and obtain the resulting ciphertext, but the key itself cannot be analyzed. The cryptanalyst attempts to deduce the key by comparing the entire ciphertext with the original plaintext. The Rivets-Shamir-Adleman encryption technique has been shown to be somewhat vulnerable to this type of analysis.

3) **Cipher text-only analysis:** The cryptanalyst has no knowledge of the plaintext and must work only from the ciphertext. This requires accurate guesswork as to how a message could be worded. It helps to have some knowledge of the literary style of the ciphertext writer and/or the general subject matter.

4) **Man-in-the-middle attack:** This differs from the above in that it involves tricking individuals into surrendering their keys. The cryptanalyst/attacker places him or herself in the communication channel between two parties who wish to exchange their keys for secure communication (via asymmetric or public key infrastructure cryptography). The cryptanalyst/attacker then performs a key exchange with each party, with the original parties believing they are exchanging keys with each other. The two parties then end up using keys that are known to the cryptanalyst/attacker. This type of attack can be defeated by the use of a hash function.

5) **Timing/differential power analysis:** This is a new technique made public in June 1998, particularly useful against the smart card, that measures differences in electrical consumption over a period of time when a microchip performs a

function to secure information. This technique can be used to gain information about key computations used in the encryption algorithm and other functions pertaining to security.

The technique can be rendered less effective by introducing random noise into the computations, or altering the sequence of the executables to make it harder to monitor the power fluctuations. This type of analysis was first developed by Paul Kocher of Cryptography Research, though Bull Systems claims it knew about this type of attack over four years before.

6. FACTORING ALGORITHMS

6.1 Brute force attack

A brute force attack against a cipher consists of breaking a cipher by trying all possible keys. Statistically, if the keys were originally chosen randomly, the plaintext will become available after about half of the possible keys are tried. The underlying assumption is, of course, that the cipher is known. A brute-force attack, or exhaustive key search, is a cryptanalytic attack that can, in theory, be used against any encrypted data (except for data encrypted in an information-theoretically secure manner). Such an attack might be used when it is not possible to take advantage of other weaknesses in an encryption system (if any exist) that would make the task easier. It consists of systematically checking all possible keys or passwords until the correct one is found. In the worst case, this would involve traversing the entire search space.

6.2 Fermat's factorization tests

It is an extremely simple method essentially based on the relation

x2 − y2 = (x − y)(x + y). If we can find y such that n + y2 = x2 then (x − y)|n and also (x + y)|n .Fermat's method works well when the number's factors into two terms of approximately equal size that is factors which are nearer to the square root of the number to be factorised. It works poorly when the factors are of very different sizes.

The Pseudocode for Fermats Theorem is given as

FermatFactor(N): // N should be odd

a ← ceil(sqrt(N))

b2 ← a*a - N

whileb2 isn't a square:

a ← a + 1 // equivalently: b2 ← b2 + 2*a + 1

b2 ← a*a - N // a ← a + 1

endwhile

returna - sqrt(b2) // or a + sqrt(b2)

For example, to factor N=5959, the first try for a is the square root of 5959 rounded up to the next integer, which is 78. Then b2=782-5959=125. Since 125 is not a square, a second try is made by increasing the value of a by 1. The second attempt also fails, because 282 is again not a square.

Try:	1	2	3
a	78	79	80
b^2	125	282	441
b	11.18	16.79	21

6.3 TRIAL DIVISION

Given an integer n, trial division consists of systematically testing whether n is divisible by any smaller number. Clearly, it is only worthwhile to test candidate factors less than n, and in order from two upwards because an arbitrary n is more likely to be divisible by two than by three, and so on.

With this ordering, there is no point in testing for divisibility by four if the number has already been determined not divisible by two, and so on for three

and any multiple of three, etc. Therefore, effort can be reduced by selecting only prime numbers as candidate factors. Furthermore, the trial factors need go no further than because, if n is divisible by some number p, then n = p × q and if q were smaller than p, n would have earlier been detected as being divisible by qor a prime factor of q.

6.4 QUADRATIC SIEVE

The quadratic sieve algorithm (QS) is an integer factorization algorithm and, in practice, the second fastest method known (after the general number field sieve). It is still the fastest for integers under 100 decimal digits or so, and is considerably simpler than the number field sieve. It is a general-purpose factorization algorithm, meaning that its running time depends solely on the size of the integer to be factored, and not on special structure or properties. It was invented by Carl Pomerance in 1981 as an improvement to Schroeppel's linear sieve.

Partial breaks

The results of cryptanalysis can also vary in usefulness. For example, cryptographer Lars Knudsen (1998) classified various types of attack on block ciphers according to the amount and quality of secret information that was discovered:

- Total break — the attacker deduces the secret key.
- Global deduction — the attacker discovers a functionally equivalent algorithm for encryption and decryption, but without learning the key.
- Instance (local) deduction — the attacker discovers additional plaintexts (or cipher texts) not previously known.
- Information deduction — the attacker gains some Shannon information about plaintexts (or cipher texts) not previously known.
- Distinguishing algorithm — the attacker can distinguish the cipher from a random permutation.

References

[1]. William Stallings, ***"Cryptography and Network Security - Principles and Practice",*** *Fifth Edition*, Prentice Hall, ISBN: 978-0-13-609704-4.

[2]. Hans Riesel, ***"Prime Numbers and Computer methods for factorization",*** Progress in Mathematics, Vol.57, ISBN: 0-8176-3291-3.

[3]. Prime Numbers – A Computational Perspective

[4]. Santanu Sarkar, ***"Some Results on Cryptanalysis of RSA and Factorization"***, PhD thesis, ISI Kolkata, 2011