The Application of Fermat's Little Theorem in Cryptography

Hengli Wang

Abstract

This article aims to demonstrate the application of Fermat's theorem in cryptography. With the development of technology, we have access to more and more information, and the necessity of encryption has also increased. As a mainstream public key cryptography algorithm, RSA encryption will demonstrate the application of Fermat's theorem in RSA encryption in this article. To achieve this, I will introduce cryptography and Fermat's theorem separately. Then, I will explain how the RSA encryption algorithm and Fermat's theorem are applied to the RSA encryption algorithm. Finally, I will demonstrate it using Python. The theoretical proof and encryption process were elaborated on through empirical Python programming examples. Then, I will discuss whether RSA encryption is safe. Finally, I will give my conclusion.

Keywords: public key, private key, sharing prime, Fermat's little theory.

Introduction of Cryptography (Mohamed Barakat, Christian Eder, Timo Hanke An Introduction to Cryptography)

Cryptography is a technique that ensures information and communication security using code. Therefore, only the intended recipient of the information can process it, preventing others from accessing the private information. The technologies used in cryptography mainly include:

Symmetric key cryptography: In this encryption system, the sender and receiver of a message use a single shared key to encrypt and decrypt the message. Although this method is efficient, sometimes receivers and senders cannot exchange keys safely. DES and AES are two members of a large family of symmetric encryption systems.

Hash function: Hash function is an algorithm that generates fixed-length hash values based on input data. These hash values are unidirectional transformations, making reverse engineering the original data difficult. Hash functions are used to encrypt passwords and ensure data integrity.

Asymmetric key cryptography, also known as public key cryptography, uses one encryption public key and one decryption private key. The sender can encrypt the information using the public key from the recipient, while the recipient decrypts the message using their private key. Asymmetric key cryptography solves the problem of key exchange in symmetric cryptography. Examples include RSA and Diffie Hellman.

Cryptography plays a crucial role in protecting digital communication, sensitive data, and privacy. It involves encryption and decryption processes, using a key to convert plaintext into ciphertext and vice versa. Cryptography protects data during transmission and storage, allowing secure communication in the presence of adversaries.

Fermat's little theory

Fermat's little theory is that if p is a prime number, a is an integer that cannot be divided by p, then . (A GENERALIZATION OF FERMAT'S LITTLE THEOREM Frank S. Gillespie)

Now, I will use different methods to prove this theory.

1. Using polynomial theorem

This summation is the sequence of all non-negative integer indices k_1 obtained by k_a. Therefore, the sum of all k_i is n; if we represent an as the sum of 1 to the path, such as power $(1 + 1+ 1+\ldots 1_a)^p$, we can get:

$$a^p = \sum_{k_1,k_2\cdots,k_a} \binom{p}{k_1,k_2,\cdots,k_a}$$

If p is prime, k_j isn't equal to p for any j, then we have:

$$\sum_{k_1,k_2,\cdots,k_a} \binom{p}{k_1,k_2,\cdots,k_a} \equiv 0 (\mathrm{mod}\, p)$$

If k_j=p, we have:

$$\sum_{k_1,k_1,\ldots,k_a} \binom{p}{k_1,k_2,\ldots,k_{ca}} \equiv 1 (\mathrm{mod}\, p)$$

one element existing, such that k_j = p, made the theorem hold.

(Veisdal, Jørgen. "Fermat's Little Theorem." Medium, Cantor's Paradise, 25 July 2019)

2. Modular Algorithm Proof

Proof of Fermat's Little Theorem

If we aim to prove this theorem, we begin by establishing two sets:

One set, S, contains integers from 1 to P - 1.
The other set, Sa, comprises a, 2a, 3a, and so on up to (P - 1) * a.
Our objective is to demonstrate that the elements in set Sa when divided by P, yield remainders that exactly match another permutation of set S. In simpler terms, when divided by P, all elements in Sa correspond to the elements in set S, albeit in a different order.
We employ proof by contradiction:
Let's assume the existence of i_a and j_a, where $i_a \equiv a \pmod P$ and $j_a \equiv a \pmod P$. On subtracting, if we find $i > j$ such that $(i - j) * a \equiv 0 \pmod P$, this leads to a contradiction, as $(i - j) * a$ cannot be congruent to zero modulo P.
This step demonstrates that ai and aj cannot exist with the same remainder.
Therefore, when the set S is divided by P, the resulting set remains as 1, 2, 3, 4... up to P - 1, albeit in a different arrangement.
Hence, $a \equiv k_1 \pmod P$, $2a \equiv k_2 \pmod P$, $3a \equiv k_3 \pmod P$, ..., $(P - 1)a \equiv k_{(P-1)} \pmod P$, where k_1, k_2, up to $k_{(P-1)}$ are mutually distinct.
With $1 \le k_i \le P - 1$, the product $k_1 * k_2 * k_3 * \ldots * k_{(P-1)}$ equals P - 1.
Considering $a \equiv k_1 \pmod P$, $2a \equiv k_2 \pmod P$, $3a \equiv k_3 \pmod P$, ..., $(P - 1)a \equiv k_{(P-1)} \pmod P$ and multiplying these congruences yields:
(P - 1)! * a^(P-1) ≡ (P - 1)! (mod P)
a^(P-1) ≡ 1 (mod P)
Thus, the proof is complete.
(Veisdal, Jørgen. "Fermat's Little Theorem." Medium, Cantor's Paradise, 25 July 2019)
The above two methods are difficult to understand; I will use an easy-to-understand method to prove: the bead threading proof method.
Let p be a prime number and any integer prime to p. Then we must show that.

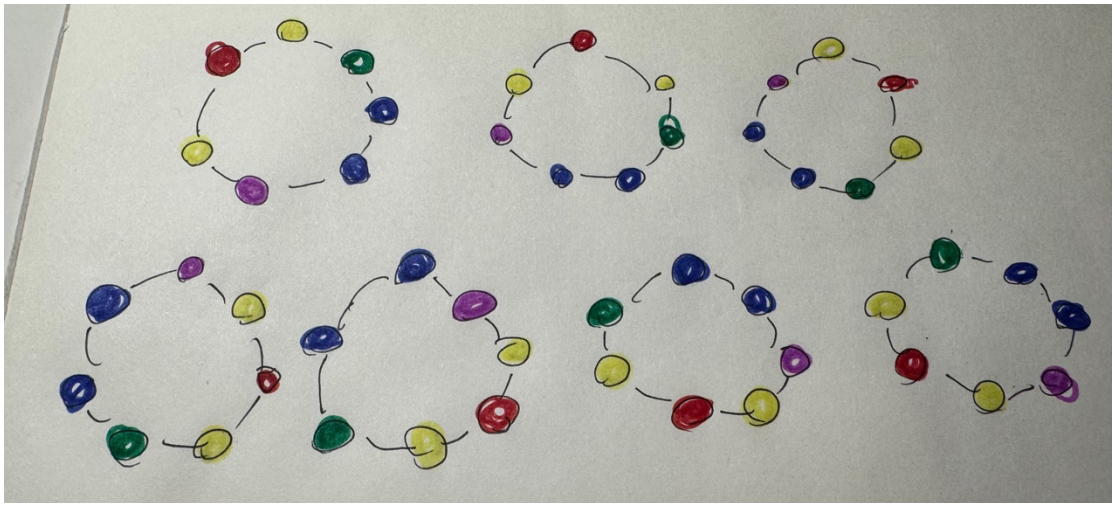
$$a^{p-1} \equiv 1 (mod\ p)$$

$$a^p \equiv a (mod\ p)$$

We have p discs and colors, so there are a^p coloring methods.

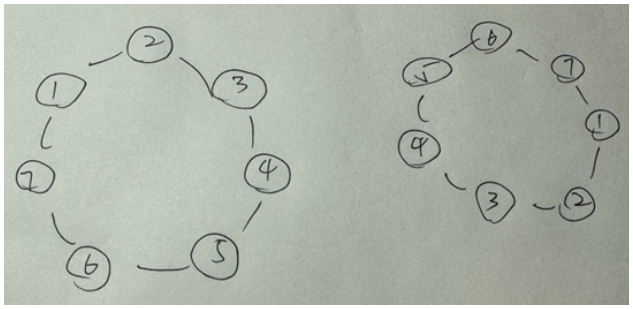
$$\begin{cases} using\ one\ color: a \\ at\ least\ using\ two\ colors: a^p - a \end{cases}$$

If we divide the a^p-a's into groups of p's, then we can show that the a^p-a's are divisible by p's, which will be proved.
We can imagine this as a string of beads on a necklace, where each disc represents a bead on the necklace. No matter how you rotate it around the center in a plane, it remains the same necklace. This gives us an idea: we can group identical necklaces, where 'identical' means they have the same coloring after rotation, considering it as an approved coloring method within a group. Let's take an example: assuming we have a total of seven beads, and we color them with at least two different colors in any order. Afterward, we rotate it. Each rotation results in a new coloring method, but the order of these colors remains the same. Therefore, we obtain six distinct coloring methods after six rotations, and the seventh rotation brings us back to the initial coloring. Hence, precisely seven distinct coloring methods belong to the same necklace. These methods can be grouped. It seems that we have resolved this proof.



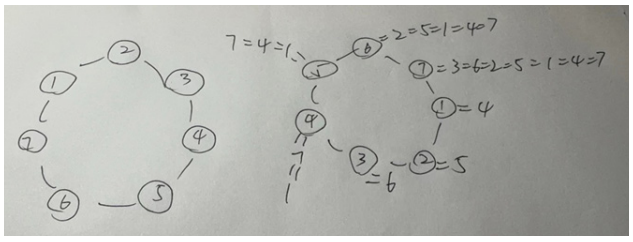
For any given coloring method, it will be grouped with others; within that group, there are a total of P distinct coloring methods. In our previous example, seven distinct coloring methods indicated that a^P - a can be divided into multiple groups, each containing P elements. However, some might wonder if two identical coloring methods could exist within a single group. Or if during the rotation process, a particular coloring method encounters the same

color arrangement before returning to its initial position. If this were the case, then a group should not have P distinct coloring methods; we would need to deduct at least one from the total P possibilities. Now, let's explain why this scenario is impossible. For simplicity, let's label these discs. Subsequently, we'll use at least two colors to color them easily. After coloring, let's assume these discs are rotated K times and return to the exact initial color arrangement, using K equals three to illustrate this scenario.

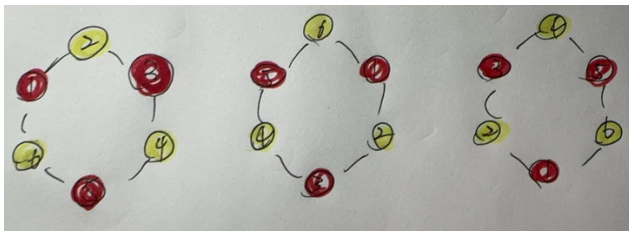


After three rotations, if the colors match, a specific sequence of discs must align with another sequence. For instance, disc 1 aligns with disc 4, indicating their colors must be identical. Likewise, disc 2 aligns with disc 5, and disc three must match disc 6. Further, disc 4 aligns with disc 7, but as disc 4 is already identical to disc 1, it follows that disc 7 is also identical to disc 1. Then, disc 5 aligns with disc 1, making disc 5 identical to disc 1 disc 4, and 7 by extension. Next, disc 6 aligns with disc 2, making disc 6 identical to disc 2 and, therefore, to disc 5, disc 1, disc 4, and disc 7.

Consequently, disc 7 aligns with disc 3, implying that disc 7 is identical to disc 3 and identical to disc 6, disc 2, disc 5, disc 1, disc 4, and disc 7. Ultimately, we realize that the color arrangement of each disc is identical. However, we initially stipulated that at least two colors should be used for coloring. This discrepancy arises because if, after K rotations, the colors are unchanged, it implies that the sequence of discs repeats in cycles of length K. This is where the condition of P being a prime number becomes crucial. Since P is a prime number, no integer greater than one can evenly divide it; therefore, as observed here, cyclic arrangements cannot occur.



So, if P is not a prime number, for example, let's say P equals six, then there is a possibility that a specific coloring method might unexpectedly repeat after rotation. For instance, let's consider a scenario where, after every two rotations, the arrangement returns to its original state. In this case, for the group associated with this particular coloring method, there are only two distinct arrangements instead of forming a total of six. This explains why we require P to be a prime number.



This shows that for a prime number P, there is always a way to distribute a^P - a distinct coloring method into multiple groups based on approved coloring methods. This completes the proof of Fermat's Little Theorem.

the RSA encryption algorithm

I mentioned earlier that RSA encryption is asymmetric, requiring both public and private keys. The public key is public, meaning that both the encryptor and the decryptor have these numbers (e, n). The private key also has two numbers (d, n). One of the two numbers in the public key, n, is a number that both the encryptor and the decryptor have. The only confidential number is d in the private key. The rules for generating public key and private key are as follows: (Gurpreet Singh, Supriya A Study of Encryption Algorithms (RSA, DES, 3DES and AES) for Information Security)

1. Choose two prime numbers that are big enough.
2. Calculate the product of p and q as n, the number available for both the encryptor and the decryptor.
3. Using Euler function, Represents the number of coprime with n in positive integers less than or equal to n; we can figure out .
4. Choose an integer e, let), and e is coprime with $\phi(n)$, and e is a number in the public key.
5. Calculate d such that . D is a number in a private key that the encryptor and the decryptor cannot know.

Then, the private key consists of (d, n), and the public key consists of (e, n).

The process of encode and decode:

Assuming that C is the ciphertext, and M is the plaintext.

Encryption:

1. Convert the plaintext into an integer m, ensuring that m is much smaller than n.
2. Compute the ciphertext c: , where ^ denotes exponentiation and mod denotes the modulo operation.
3. The ciphertext c is the result of encryption.

Decryption:

1. Receive the ciphertext c.
2. Compute the plaintext m: .
3. The plaintext m is the result of decryption.

For the security and reliability of encryption, p and q must be two extremely large prime numbers, which means that n is also a very large number.

Relation between encode and decode and Fermat's little theory.

Now, I will prove the decryption process, . Because and , . According to the exponentiation law of mod, . Because , . Because n is a very large number if we can prove that , then .

Therefore, now our goal changes to how to prove . Proving this requires using Fermat's little theory: if p is a prime number, while a is an integer that cannot be divided by p, then . In this case, , and a=m. So, can be transfer to . When n is much larger than , only when can the equation be established. Therefore, . Therefore, . Now, the equation changes to .

The context above proves the process of decryption.

Demonstrate the process of RSA encryption using Python and show its relationship with Fermat's Little Theorem.

1. We need to import the necessary modules. We need sympy to generate a random prime number and random for generating random numbers.

```
from sympy import randprime, mod_inverse
import random
```

2. Generate the key

We need to randomly generate two prime numbers: p and q. Then calculate n=p*q, and (phi), which is

```
p = randprime(1, 100)
q = randprime(1, 100)
n = p * q
phi = (p - 1) * (q - 1)
```

3. Generate a public key e which satisfies , and e is coprime with .

```
def generate_e(phi_n):
    e = 2
    while True:
        if math.gcd(e, phi_n) == 1:
            break
        e += 1
    return e

e = generate_e(phi_n)
```

Because e is coprime with , e has a multiplicative inverse under mod φ(n).

4. Generate the private key.

```
d = mod_inverse(e, phi_n)
```

We can directly calculate the private key d, using the mod_inverse function of sympy library, which has already been introduced from step 1. This step uses Fermat's theory, which ensures the multiplicative inverse exists, as e is coprime with .

I will explain the two steps above:

1. generate_e(phi_n): Because the public key must satisfies two conditions, which is , and it must coprime with , so we use math.gcd(e, phi_n) == 1 to check whether e coprime with . If we find that e satisfies the condition above, we can stop the loop and use that e as a public key

2. mod_inverse(e, phi_n): this step uses Fermat's little theory to find the private key.

In RSA encryption, e and phi_ n must satisfy coprime, which means that there exists a d such that (d * e) mod φ (n) =1; we call this d as the inverse of multiplication under e mod φ (n). Using the library function mod provided by Python_ Inverse can directly calculate d.

5. Encryption

```
M = 123
C = pow(M, e, n)
```

Using the public key(n, e), we can calculate . In Python, the pow function can directly perform the modulo power operation.

6. Decryption

```
decrypted_M = pow(C, d, n)
```

The reversion of the encryption process is the decryption process, and then can be calculated by using the private key (n, d). After this step, we can obtain the original plaintext message.

RSA encryption is not 100% secure.

According to Patrick nohe's essay, is it still safe to use encryption? RSA encryption has two major problems. First, they are not really that random. Second, almost everyone uses the same ones. In other words, there are a lot of common CSPRNGs which can be easily cracked. These two problems will result in many public keys sharing a prime.

(Is it still safe to use RSA Encryption? Patrick Nohe)

According to the 2012 research paper titled "Ron was wrong, Whit is right", many public keys sharing a prime will reduce security for the following reasons.

If two public keys have the same prime number, then that prime number is the public divisor. Since determining a greatest common divisor is much simpler than factoring a number prime, if someone knows this public divisor, they can decrypt any message sent with this public key. This makes RSA encryption insecure.

(Ron was wrong; Whit is right)
The researchers tested 6.2 million public keys that people were using, and they cracked 12,934 of them. This means that RSA encryption is less than 99.8% secure (Ron was wrong, Whit is right)
RSA encryption hasn't been cracking, but it is vulnerable. One method in a 2013 essay proposed by researchers helped to factor and crack nearly 13000 public keys.

Conclusion

The Fermat's Little Theorem plays a significant role in RSA encryption. It can be used to check if a number is prime, which is crucial for key generation. I first introduced cryptography and then explained three different methods of proving the Fermat's Little Theorem. Next, I described the RSA encryption algorithm and demonstrated the decryption process using the Fermat's Little Theorem.
Since most RSA encryption is employed in programs, I demonstrated it in Python to be more realistic and included explanations for each step. Finally, I questioned whether RSA encryption is secure by looking up citation examples and data.

Acknowledgment

At first, I thought that the knowledge of number theory was useless because we don't need to use it in real life. It was Prof. H. L. Bray who showed me that such a simple and seemingly impractical number theory can be so skillfully integrated with cryptography, which made me deeply feel the charm of mathematics and inspired me to write an article about the use of Fermat's Little Theorem in RSA encryption. I will fulfill his expectations, do my best to apply math to life, and try to bed in convenience for mankind. Heartfelt thanks to Prof. H. L. Bray for guiding me. I wish you all the best.

Reference

1. "Fermat's Little Theorem." From Wolfram MathWorld, mathworld.wolfram.com/FermatsLittleTheorem.html?utm_content=buffer436bc&utm_source=buffer&utm_medium=twitter&utm_campaign=Buffer. Accessed 29 Aug. 2023.
2. Lenstra, Arjen K. Ron Was Wrong, Whit Is Right - IACR, eprint.iacr.org/2012/064.pdf. Accessed 28 Aug. 2023.
3. Mohamed Barakat, Christian Eder, Timo Hanke. Rptu, agag-ederc.math.rptu.de/~ederc/download/Cryptography.pdf. Accessed 29 Aug. 2023.
4. nohe, Patrick. "Google Chrome Adds Support for a Hybrid Post-Quantum Cryptographic Algorithm." Hashed Out by The SSL StoreTM, 18 Aug. 2023, www.thesslstore.com/blog/is-it-still-safe-to-use-rsa-encryptio.
5. Pournaki, M. R. Generalizations of Fermat's Little Theorem via Group Theory, www.researchgate.net/profile/M-R-Pournaki/publication/228636515_Generalizations_of_Fermat%27s_Little_Theorem_via_Group_Theory/links/09e4150d54990bda04000000/Generalizations-of-Fermats-Little-Theorem-via-Group-Theory.pdf?origin=publication_detail. Accessed 29 Aug. 2023.
6. Veisdal, Jørgen. "Fermat's Little Theorem." Medium, Cantor's Paradise, 25 July 2019, www.cantorsparadise.com/fermats-little-theorem-fbc88498d54e.