The Mathematics Behind RSA Encryption

A.Xu ma OS

1 Introduction

Every time you log in to online banking or buy something online, your data is protected by mathematics. One of the most famous systems for doing this is RSA, named after Ron Rivest, Adi Shamir and Leonard Adleman, who published it in 1978.

RSA is a public-key cryptographic system whose security relies on prime numbers and modular arithmetic. Unlike symmetric encryption, RSA uses two keys: a public key, which can be shared openly, and a private key, which must remain secret.

Its central idea is the construction of a *trapdoor function*: an operation that is easy to perform but computationally infeasible to reverse without additional information, known as the trapdoor.

2 Constructing RSA

RSA starts with two distinct prime numbers, p and q, which are multiplied to produce the modulus $n = pq$. The prime factors are kept secret, whilst their product n becomes public.

Then it uses Euler's totient function, $\varphi(n)$, which counts the positive integers up to n that are co-prime to n. Since p and q are distinct primes,

$$\varphi(n) = (p-1)(q-1).$$

An integer e is then chosen such that $1 < e < \varphi(n)$ and $\gcd(e, \varphi(n)) = 1$. In practice, $e = 65537$ is almost always used. To reverse the process, we find an integer d satisfying

$$\boxed{ed \equiv 1 \pmod{\varphi(n)}}.$$

Therefore, d is the multiplicative inverse of e modulo $\varphi(n)$, and can be calculated efficiently using the Extended Euclidean Algorithm.

The public key is (n, e), and d forms an essential part of the private key.

A message is first converted into an integer m with $0 \leq m < n$. Encryption and decryption are then defined by

$$\boxed{c \equiv m^e \pmod{n}} \qquad \text{and} \qquad \boxed{m' \equiv c^d \pmod{n}},$$

respectively, where m is the original message and c is the ciphertext.

2.1 A Worked Example

Take the small primes $p = 61$ and $q = 53$. Then

$$n = 61 \times 53 = 3233 \qquad \text{and} \qquad \varphi(n) = 60 \times 52 = 3120.$$

Choosing $e = 17$, which is co-prime to 3120, the Extended Euclidean Algorithm gives $d = 2753$, since

$$17 \times 2753 = 46801 = 15 \times 3120 + 1.$$

To encrypt the message $m = 65$, we calculate $c \equiv 65^{17} \equiv 2790 \pmod{3233}$. To decrypt, we calculate $2790^{2753} \equiv 65 \pmod{3233}$, and the original message is recovered. Real RSA works in exactly the same way, but with primes hundreds of digits long.

3 Why Does RSA Work?

The correctness of RSA follows from Euler's theorem, which states that

$$\gcd(a, n) = 1 \quad \Longrightarrow \quad a^{\varphi(n)} \equiv 1 \pmod{n}.$$

Since $ed \equiv 1 \pmod{\varphi(n)}$, we can write $ed = 1 + k\varphi(n)$ for some integer k.

Substituting the encryption equation into the decryption equation gives

$$\begin{aligned} m' &\equiv (m^e)^d \equiv m^{ed} \\ &\equiv m^{1+k\varphi(n)} \\ &\equiv m \left(m^{\varphi(n)}\right)^k \pmod{n}. \end{aligned}$$

By Euler's theorem, $m' \equiv m \pmod{n}$. Since both m and m' lie between 0 and $n-1$, this means $m' = m$, and the original message is recovered.

There is, however, a subtlety: Euler's theorem requires $\gcd(m, n) = 1$. RSA must also work when the message shares a factor with n.

This can be established using the Chinese Remainder Theorem. Since $n = pq$, it is sufficient to prove that

$$m^{ed} \equiv m \pmod{p} \qquad \text{and} \qquad m^{ed} \equiv m \pmod{q}.$$

Consider the congruence modulo p. If $p \mid m$, the result is immediate. Otherwise, Fermat's Little Theorem gives $m^{p-1} \equiv 1 \pmod{p}$. Since $ed - 1 = k\varphi(n) = k(p-1)(q-1)$, $p-1$ divides $ed-1$, and it follows that

$$m^{ed} = m(m^{p-1})^r \equiv m \pmod{p}$$

for some integer r. The same argument applies modulo q. The Chinese Remainder Theorem therefore gives

$$\boxed{m^{ed} \equiv m \pmod{n}}$$

for every message m.

4 Euler's Function and Carmichael's Function

Although Euler's totient function is sufficient for constructing RSA keys, $\varphi(n)$ is not the smallest exponent that guarantees the required result.

A more precise function is Carmichael's function, $\lambda(n)$, defined as the smallest positive integer such that $a^{\lambda(n)} \equiv 1 \pmod{n}$ for every a co-prime to n.

For an RSA modulus consisting of two distinct primes, $\lambda(n) = \operatorname{lcm}(p-1, q-1)$. Using the identity $\operatorname{lcm}(a, b) = \frac{ab}{\gcd(a,b)}$, we obtain

$$\boxed{\lambda(n) = \frac{(p-1)(q-1)}{\gcd(p-1, q-1)}}.$$

Therefore, $\lambda(n) \mid \varphi(n)$.

The distinction is important: $\varphi(n)$ counts the integers relatively prime to n, whereas $\lambda(n)$ describes the smallest universal exponent for which Euler's theorem holds.

Instead of requiring $ed \equiv 1 \pmod{\varphi(n)}$, RSA can therefore use the weaker condition

$$\boxed{ed \equiv 1 \pmod{\lambda(n)}}.$$

This is sufficient for correct decryption because Carmichael's function guarantees that $m^{\lambda(n)} \equiv 1 \pmod{n}$ whenever $\gcd(m, n) = 1$. When m shares a factor with n, the argument from Section 3 still works modulo p and modulo q, since $p-1$ and $q-1$ both divide $\lambda(n)$.

In modern implementations, Carmichael's function is preferred because it gives the smallest valid private exponent d, and modern standards for generating RSA keys specify it. In our example, $\lambda(3233) = \operatorname{lcm}(60, 52) = 780$, which gives $d = 413$ instead of 2753, since $17 \times 413 = 7021 = 9 \times 780 + 1$.

5 The Trapdoor

The security of RSA comes from the difference between performing an operation and reversing it.

Calculating $c \equiv m^e \pmod{n}$ is computationally trivial, even for very large values of n. However, reversing it without the private key is believed to be infeasible. The trapdoor is knowing the primes p and q: with them, anyone can calculate $\varphi(n)$ and then d, but nobody knows an efficient way to find d from n and e alone. In fact, knowing $\varphi(n)$ is enough to factorise n, so finding d this way is as hard as factorising.

RSA therefore exploits the difficulty of factorising sufficiently large integers into primes. In practice, these primes are usually 1024 bits long, a decimal order of magnitude around 10^{308}, so the modulus n is a 2048-bit number of around 10^{617}. The largest RSA challenge number factorised so far has only 896 bits, a record set in September 2026, and factorising a 2048-bit modulus with known methods is far beyond the reach of any existing computer. However, a large enough quantum computer could factorise such numbers efficiently using Shor's algorithm, which is why new "post-quantum" cryptographic systems are being developed.

The version of RSA described here, known as "textbook RSA", always encrypts the same message to the same ciphertext, which makes it insecure on its own. Real implementations therefore always use padding schemes such as RSA-OAEP, which add randomness to the message before it is encrypted. For highly classified data, the bit length can also be increased, for example to 4096 bits.

6 Conclusion

RSA demonstrates a fascinating paradox: mathematical operations can be easy to perform but extraordinarily difficult to reverse. By turning number theory into the foundation of digital privacy, it shows that maths is not just about finding solutions, but also about exploiting the limits of computation.

References

"Carmichael Function." *Wolfram MathWorld*. `https://mathworld.wolfram.com/CarmichaelFunction.html`.

Rivest, Ronald L., Adi Shamir, and Leonard Adleman. "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems." *Communications of the ACM*, vol. 21, no. 2, 1978, pp. 120–126. `https://doi.org/10.1145/359340.359342`.

"RSA." *Stanford University Applied Cryptography Group*. `https://crypto.stanford.edu/pbc/notes/crypto/rsa.html`.

"RSA Factoring Challenge." *Wikipedia*, Wikimedia Foundation. `https://en.wikipedia.org/wiki/RSA_Factoring_Challenge`.

"Totient Function." *Wolfram MathWorld*. `https://mathworld.wolfram.com/TotientFunction.html`.