RSA CRYPTOGRAPHY

Introduction

From Wikipedia: The RSA cryptosystem is a public-key cryptosystem, one of the oldest widely used for secure data transmission. The initialism "RSA" comes from the surnames of Ron Rivest, Adi Shamir, and Leonard Adleman, who publicly described the algorithm in 1977. An equivalent system was developed secretly in 1973 at Government Communications Headquarters (GCHQ), the British signals intelligence agency, by the English mathematician Clifford Cocks. That system was declassified in 1997.

How is it done?

In RSA, we start with two large primes p and q, and compute $n = p \times q$. We also compute Euler's totient function $\varphi(n) = (p-1)(q-1)$. We choose an encryption exponent e such that $\gcd(e, \varphi(n)) = 1$, and compute the decryption exponent d as the modular inverse of e modulo $\varphi(n)$, so that:

$$e \times d = 1 \mod \varphi(n).$$

Assume Alice wants to send a message to Bob:

(1) Bob generates an RSA key pair: public key (n, e) and private key d.
(2) Bob shares his public key (n, e) openly. His private key d is kept secret.
(3) Alice converts her message m to a numerical form.
(4) Alice encrypts m using Bob's public key: $c = m^e \mod n$ (Encryption).
(5) Alice sends the ciphertext c to Bob over an open channel.
(6) Bob receives c and decrypts it using his private key: $m = c^d \mod n$ (Decryption).
(7) Bob recovers and reads the original message m.

This process ensures confidentiality: only Bob can decrypt the message because only he knows d.

Euler's Totient Theorem

Euler's Totient Theorem states that for any integer a coprime to n:

$$a^{\varphi(n)} = 1 \mod n,$$

where $\varphi(n)$ is Euler's totient function (the number of integers less than n that are coprime to n). In RSA, Euler's theorem ensures that encryption and decryption correctly reverse each other when the message is coprime to n.

Why does RSA work?

Theorem. Let $n = p \times q$, where p and q are distinct prime numbers, and let $\varphi(n) = (p-1)(q-1)$. Assume that m is an integer such that $\gcd(m, n) = 1$. Let e and d be integers satisfying $e \times d = 1 \mod \varphi(n)$. Then:

$$(m^e)^d = m \mod n.$$

Proof. Since $e \times d = 1 \mod \varphi(n)$, we can write:

$$e \times d = 1 + k\varphi(n), \quad \text{for some integer } k.$$

We have that

$$m^{ed} = m^{1+k\varphi(n)} = m \times m^{k\varphi(n)}.$$

By Euler's Totient Theorem, since $\gcd(m, n) = 1$, we deduce that

$$m^{\varphi(n)} = 1 \mod n.$$

As a result,

$$m^{1+k\varphi(n)} = m \times 1^k = m \mod n.$$

Thus, after decryption, we recover the original message m. □

Factoring n is hard

The security of RSA relies on the difficulty of factoring large integers $n = pq$, where p and q are large primes. The best classical algorithm known for factoring, the General Number Field Sieve (GNFS), has a sub-exponential running time. Below is a summary of the approximate feasibility of factoring different RSA key sizes.

Key Size (bits)	Approx. Digits	Status	Feasibility	Approximate Effort
512	155	Broken	Easy	Already factored; weeks
1024	308	Weak	Hard, possible in future	100,000 core-years
2048	617	Recommended	Infeasible	Billions of years estimated
3072	925	Strong	Infeasible	Beyond realistic capacity
4096	1234	Very strong	Infeasible	Beyond realistic capacity

Fermat's theorem and Fermat primality test

Fermat's Little Theorem states that if p is prime and a is an integer not divisible by p, then:

$$a^{p-1} = 1 \mod p.$$

Fermat primality test uses this to check if a number n is prime:

1. Pick a random integer a such that $1 < a < n - 1$.
2. Compute $a^{n-1} \mod n$.
3. If result $\neq 1$, n is composite.
4. If result $= 1$, n is *probably* prime! (but might still be a Carmichael number).

This is a probabilistic test and may give false positives.

Order of g and Shor's algorithm (Quantum Attack!)

For any integer g coprime to n, there exists a smallest positive integer p (called the order of g modulo n) such that:

$$g^p = 1 \mod n.$$

The idea is that starting from a guess which is a factor of n, $(g^{p/2} - 1)$ or $(g^{p/2} + 1)$ is a much better guess for being a factor of n. If we find one factor for n, since $n = pq$, then we are done! Finding this order is a hard problem classically and is related to factoring. Shor's algorithm, developed by Peter Shor, is a quantum algorithm that can efficiently find the order of g modulo n. By finding this order, one can factor n and break RSA security. Thus, quantum computers with enough qubits can break RSA by using Shor's algorithm, which is why post-quantum cryptography is being developed.

References

(1) Lenstra, A. K., & Verheul, E. R. (2001). Selecting Cryptographic Key Sizes. Journal of Cryptology. Available at: https://www.iacr.org/archive/asiacrypt2001/22480516.pdf

(2) Shor, P. W. (1997). Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. SIAM Journal on Computing, 26(5), 1484–1509.

(3) RSA Factoring Challenge Results. See: https://en.wikipedia.org/wiki/RSA_Factoring_Challenge