RSA — Simple Walkthrough

Key generation, encryption, decryption with a tiny numeric example

Version	1.0
Date	2025-10-19
Owner	Security Team
Audience	Engineering & Non■specialists

Overview

Part	What happens	Key formula (simple)
Key generation	Build public/private key pair	$n = p \times q$; $\varphi=(p-1)(q-1)$; choose e; find d where $d \cdot e \equiv 1$ (n
Encryption	Lock a number < n with the public key	$c \equiv m^e \pmod n$
Decryption	Unlock with the private key	$m \equiv c^d \pmod n$

1) How the key is generated

- Pick two large primes p and q (kept secret).
- Compute $n = p \times q$. Example: $p=61$, $q=53 \Rightarrow n=3233$.
- Compute Euler's totient $\varphi=(p-1)(q-1) \Rightarrow \varphi=3120$.
- Choose a public exponent e coprime with φ (common: 65537; here: 17).
- Compute the private exponent d as the modular inverse of e mod φ: $d \equiv e^{\blacksquare 1} \pmod{\varphi}$. Example: $d=2753$.
- Public key = (n, e). Private key = (n, d). Keep p and q secret and destroy them after keygen.

2) How a message is encrypted

- Encode the message as a number m with $0 \le m < n$ (in practice, use padding like OAEP; here we use a tiny plain number).
- Compute ciphertext: $c \equiv m^e \pmod n$. Example with $m=65$: $c \equiv 65^{17} \pmod{3233} = 2790$.

3) How a message is decrypted

- Compute: $m \equiv c^d \pmod n$. Example: $m \equiv 2790^{2753} \pmod{3233} = 65$.
- Correctness (idea): raising to the power e then d restores m because $d \cdot e \equiv 1 \pmod{\varphi}$.
- Speed-up in practice: compute mod p and mod q (Chinese Remainder Theorem), then recombine — much faster than working mod n directly.
- Limits: never encrypt raw data; always use padding (OAEP) and size checks to prevent oracle attacks.