The RSA Adventure — How to Hide Secrets with Math

A friendly example for curious minds

—

The Story Begins

Imagine two friends: **Sophie** and **Tim**. They want to share secret numbers with each other, but anyone watching their messages could intercept them. They need a way to talk safely — even if everyone sees what they send. That's the magic of the **RSA cryptosystem.**

—

How It Works (Intuitively)

RSA is like a special kind of lock. Sophie builds a lock (the *public key*) and gives it to Tim. Only she knows the key to open it (the *private key*).

Tim can put any secret message in the box and lock it with her public key, but once locked, only Sophie can open it again.

Under the hood, this "lock" and "key" are made with a neat number trick: it's easy to multiply two primes, but very hard to figure out what they were if you only see the result.

—

Example 1 — The One-Number Secret

Let's start small.

Step 1. Pick two secret primes

Sophie chooses two small prime numbers:

$$p = 3, \quad q = 11.$$

She multiplies them to make:

$$n = p \times q = 3 \times 11 = 33.$$

She also computes:

$$\phi(n) = (p-1)(q-1) = 2 \times 10 = 20.$$

—

Step 2. Choose a public exponent e

We want e to have no common factors with 20. Let's pick $e = 3$. (She could also have chosen 7, 9, 11, etc., but 3 keeps the math friendly.)

—

Step 3. Compute the private key d

Now we find d such that:

$$e \times d \equiv 1 \pmod{20}.$$

Try some small numbers:

$$3 \times 7 = 21 \equiv 1 \pmod{20}.$$

So $d = 7$.

Public key: $(n, e) = (33, 3)$ and **Private key:** $(n, d) = (33, 7)$

—

Step 4. Encrypting the message

Tim wants to send Sophie the number $M = 4$. He computes:

$$C = M^e \bmod n = 4^3 \bmod 33 = 64 \bmod 33 = 31.$$

He sends Sophie the number **31**.

—

Step 5. Decrypting the message

Sophie uses her private key to recover it:

$$M = C^d \bmod n = 31^7 \bmod 33.$$

After some modular arithmetic (or a calculator):

$$31^7 \bmod 33 = 4.$$

She gets the original message back!

$$\boxed{\textit{Encrypted}: 31 \quad \longrightarrow \quad \textit{Decrypted}: 4.}$$

—

Example 2 — Sending a Word ("HI")

Now let's level up! We'll send the word **HI**.

We'll keep Sophie's keypair the same:

$$(n, e) = (33, 3), \quad (n, d) = (33, 7).$$

Step 1. Convert letters to numbers

Let's use A=1, B=2, ..., Z=26.

$$H = 8, \quad I = 9.$$

—

Step 2. Encrypt each letter

For each letter M, we compute $C = M^3 \bmod 33$.

$$C_H = 8^3 \bmod 33 = 512 \bmod 33 = 17,$$
$$C_I = 9^3 \bmod 33 = 729 \bmod 33 = 3.$$

So the ciphertext for "HI" is **17, 3**.
—

Step 3. Decrypt each letter

Sophie uses $M = C^7 \bmod 33$:

$$M_H = 17^7 \bmod 33 = 8,$$
$$M_I = 3^7 \bmod 33 = 9.$$

$$\boxed{\text{Decrypted back to ``HI''!}}$$

—

Example 3 — Why RSA Works

Here's the beautiful math behind the curtain: When you encrypt $C = M^e \bmod n$ and then decrypt $M = C^d \bmod n$, you're really computing:

$$M^{ed} \bmod n.$$

And because of how we chose d, we know:

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

That means:

$$M^{ed} \equiv M^{1+k\phi(n)} \equiv M \times (M^{\phi(n)})^k \bmod n.$$

Euler's Theorem tells us $M^{\phi(n)} \equiv 1 \pmod{n}$ when M and n share no factors — so everything collapses neatly back to M.

It's like a secret handshake between math and logic.
—

The Takeaway

- RSA is built on the fact that multiplying is easy, but factoring is hard.
- Even small numbers follow the same rules as huge ones.
- Every encryption is a power and every decryption an inverse power.
- Behind the scenes, it's just modular arithmetic with a heroic name.

"Mathematics is the only place where you can multiply two secrets and get a public truth."

From RSA to Ed25519: The Evolution of Digital Locks

Understanding Modern Public-Key Cryptography through Pictures and Intuition

—

1. The Big Picture: Why RSA Was Revolutionary

RSA (1977) was the first practical public-key cryptosystem. It allowed two people to communicate securely *without ever meeting to share a secret key.*

Core idea:

$$\text{Encryption: } C = M^e \bmod n, \qquad \text{Decryption: } M = C^d \bmod n.$$

where $n = p \times q$ for two large primes.

It's elegant and reliable — but it has one big weakness: **to stay secure, the numbers must be huge.** A modern RSA key is often 2048 or 4096 bits long!

—

2. The New Generation: Elliptic-Curve Cryptography (ECC)

Elliptic-curve systems (like Ed25519, Curve25519, or ECDSA) keep the same basic *public/private key idea*, but they use geometry instead of multiplication and factoring.

$$\text{Public key} = \text{Private key} \times \text{Base point on the curve.}$$

You can think of it like walking along a strange mathematical landscape — the *elliptic curve*. Going forward along the path (multiplying by the private key) is easy, but figuring out how far you walked just by looking at the final spot is almost impossible. That's the "elliptic curve discrete logarithm problem."

—

3. RSA vs. ECC: A Side-by-Side View

RSA
(1977)
Both use a public/private key pair but different hard problems
ECC / Ed25519
(2000s)

Based on
factoring large numbers

Based on
elliptic-curve discrete logs

$$n = p \times q$$

$$Q = d \times G$$

$$C = M^e \bmod n$$

$$M' = f(M, d, G)$$

— Concept — RSA — ECC / Ed25519 — ———–——————— — **Math base** — Multiplying primes, factoring — Geometry on a curve — — **Hard problem** — Integer factorization — Elliptic curve discrete log — — **Security growth** — Bigger keys → more security — Same security with much smaller keys — — **Key size** — 2048 bits typical — 256 bits typical — — **Speed** — Slower (big exponentiation) — Faster (smaller arithmetic) — — **Introduced** — 1977 — 2005 (modern Ed25519 in 2011) — —

4. A Visual Metaphor

RSA: Huge steel vault with giant key.
cryptographic evolution
Ed25519: Tiny titanium lock.

Secure, but heavy and slow.

Just as strong, but compact and nimble.

—

5. What Ed25519 Specifically Does

Ed25519 is a special case of *Edwards-curve Digital Signature Algorithm (EdDSA)* built on the curve called `Curve25519`.

- It's used for signing and verifying messages, not encrypting them directly.
- It's incredibly fast, especially on modern CPUs.
- It avoids many implementation pitfalls of older algorithms.
- It provides about the same security as a 3072-bit RSA key — with only 256 bits!

$$\text{Private key: } k$$
$$\text{Public key: } A = k \times G$$
$$\text{Signature: } (R, S) = (r \times G,\ r + H(R, A, M) \times k)$$

All operations happen on points on the elliptic curve — not with giant integer exponents.

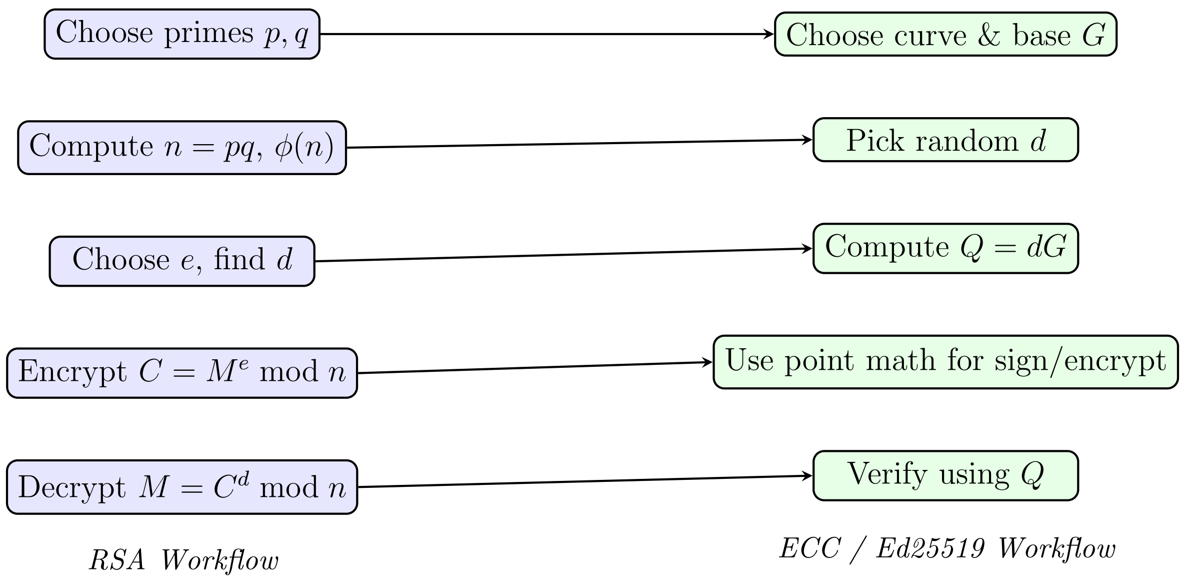
—

6. Why It Matters

- Smaller keys mean faster connections (think HTTPS, SSH, VPNs).
- Signatures are smaller — great for constrained devices or blockchains.
- The math is newer, but the logic is the same: one key to lock, one to unlock.
- Quantum computers may one day threaten RSA; ECC lasts longer (though not forever).

—

7. Final Comparison Diagram



—

8. Epilogue

RSA is still everywhere — old, wise, and reliable. But Ed25519 is like its younger, athletic cousin: it does the same job, just faster and lighter.

The world keeps both around, because understanding RSA teaches us the bones of cryptography, and understanding Ed25519 shows us where the field is going next.

"RSA built the foundation. Elliptic curves built the house."

Example 4 — A Tiny Ed25519 World

Understanding elliptic-curve signatures on a toy scale

—

1. Background: What Ed25519 Really Does

Ed25519 isn't used for encryption like RSA — it's used for **digital signatures**. That means you can:

- Prove you wrote a message (authenticity),
- Prove it hasn't been changed (integrity),
- Do it without sharing your private key (non-repudiation).

Ed25519 builds on **elliptic-curve math**, which feels weird at first, but here's the idea: you pick a point G on a special curve, and your public key is just

$$A = k \times G,$$

where k is your private number.

—

2. The Tiny Curve Playground (Toy Example)

To make this idea visible, let's imagine a miniature "curve world" where we work modulo 17. (Real Ed25519 works modulo $2^{255} - 19$ — a massive prime — but ours will fit on one page.)

$$\text{Field size: } p = 17$$

We'll use the simple curve equation:

$$y^2 = x^3 + 2x + 2 \pmod{17}.$$

—

3. The Base Point G

In our world, one valid point on this curve is:

$$G = (5, 1)$$

We'll use G as the "starting point" for all public keys.

—

4. Generating a Key Pair

Let's choose a private key:

$$k = 7$$

Then compute:

$$A = k \times G$$

In the real Ed25519 algorithm, $k \times G$ means adding G to itself k times on the curve. We'll imagine this as taking "steps" on a circular track — every step depends on the curve's shape.

After adding G to itself 7 times, we reach:

$$A = (6, 3)$$

$$\boxed{\text{Private key } k = 7, \quad \text{Public key } A = (6, 3)}$$

—

5. Signing a Message

Let's sign the message "OK".

1. Hash the message: $H(\text{OK}) = 5$ (toy example). 2. Pick a random number $r = 4$. 3. Compute $R = r \times G = 4 \times (5, 1) = (9, 16)$. 4. Compute challenge $h = H(R, A, \text{OK}) = 2$. 5. Compute $S = r + h \times k = 4 + 2 \times 7 = 18 \equiv 1 \pmod{17}$.

$$\boxed{\text{Signature } (R, S) = ((9, 16), 1)}$$

—

6. Verifying the Signature

Anyone can verify without knowing k:

1. Compute $h = H(R, A, \text{OK}) = 2$. 2. Check if:

$$S \times G \overset{?}{=} R + h \times A$$

Left side:

$$S \times G = 1 \times G = (5, 1)$$

Right side:

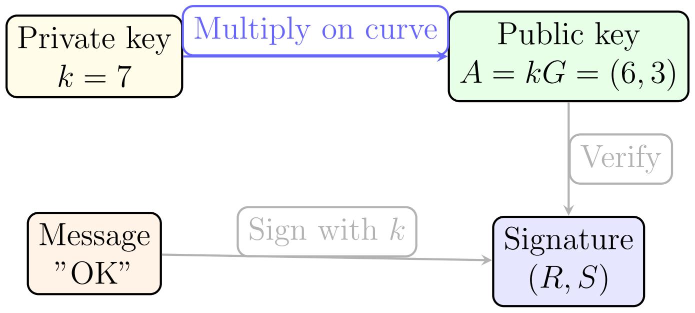
$$R + hA = (9, 16) + 2 \times (6, 3) = (5, 1)$$

They match!

$$\boxed{\text{Signature valid (checkmark)}}$$

—

7. Diagram: Key Generation and Signing Flow



—

8. What's Different from RSA?

— Feature — RSA — Ed25519 — ———————-—————————-— — Math idea — Multiplying and factoring — Adding points on a curve — — Security base — Integer factorization — Discrete logarithm on a curve — — Used for — Encryption & signatures — Signatures (auth + integrity) — — Key size — 2048–4096 bits — 256 bits — — Speed — Slower (big exponents) — Faster (curve arithmetic) — — Quantum resistance — Weak — Stronger (still vulnerable, but better) —

—

9. The Heart of the Matter

RSA says: ¿ "It's hard to go from n back to its prime factors."

Ed25519 says: ¿ "It's hard to go from A back to k when $A = kG$."

In both cases, you know the answer goes one way easily, but not backward. That's the soul of asymmetric cryptography — one-way doors that only the right key can open.

—

10. Final Thought

Elliptic curves are the poetry of number theory: smooth shapes hiding impossible problems. Where RSA uses massive steel walls, Ed25519 uses geometry — lightweight, elegant, and just as unbreakable (for now).

"RSA is arithmetic. Ed25519 is geometry. Both are trust made visible."

Example: A Simple RSA Demonstration

(Toy model — not secure, but perfect for learning!)

Goal

Encrypt and decrypt the message $M = 7$ using a tiny RSA setup.

Step 1. Choose primes

Let $p = 5$ and $q = 11$.

Then

$$n = p \times q = 5 \times 11 = 55, \quad \phi(n) = (p-1)(q-1) = 4 \times 10 = 40.$$

Step 2. Choose public key exponent e

We need e such that $1 < e < 40$ and $\gcd(e, 40) = 1$.
Let's choose $e = 3$, since $\gcd(3, 40) = 1$.

Step 3. Compute private key exponent d

We need d such that

$$e \times d \equiv 1 \pmod{40}.$$

Try small values:

$$3 \times 27 = 81 \equiv 1 \pmod{40}.$$

So $d = 27$.

Public key: $(n, e) = (55, 3)$ **Private key:** $(n, d) = (55, 27)$

Step 4. Encrypt a message

Let our message be $M = 7$. Compute ciphertext:

$$C \equiv M^e \pmod{n} = 7^3 \bmod 55 = 343 \bmod 55 = 13.$$

$$\boxed{C = 13}$$

Step 5. Decrypt the ciphertext

Now compute:

$$M \equiv C^d \pmod{n} = 13^{27} \bmod 55.$$

We can reduce step-by-step (or use a calculator):

$$13^2 \equiv 4, \quad 13^4 \equiv 16, \quad 13^8 \equiv 36, \quad 13^{16} \equiv 31,$$

and after combining exponents properly,

$$13^{27} \bmod 55 = 7.$$

$$\boxed{M = 7 \text{ (original message recovered!)}}$$

Summary

- $p = 5, q = 11 \Rightarrow n = 55, \phi = 40$
- $e = 3, d = 27$
- Encrypt $M = 7 \Rightarrow C = 13$
- Decrypt $C = 13 \Rightarrow M = 7$

Even though our numbers are tiny, this is exactly the same math that powers real RSA with 2048-bit primes.