Public-Key Cryptosystem & RSA

DSC 4110/6111: Modern Cryptography and AI/ML Security

Dr. Laltu Sardar

School of Data Science,
Indian Institute of Science Education and Research Thiruvananthapuram (IISER TVM)

August 10, 2026

Table of Content

IISER THIRUVANANTHAPURAM

Public-Key Cryptosystems

Public-Key Cryptosystem:
A cryptosystem in which each entity has a key pair

$$\boxed{(pk, sk)}$$

where pk is publicly known and sk is kept secret.
Security relies on a computationally hard mathematical problem.

	Public-Key	**Private-Key (Symmetric)**
Keys	pk, sk	Shared secret K
Key relation	Mathematically related	Same secret key
Key distribution	Public key can be published	K must be shared secretly
Main use	Key establishment, signatures	Bulk encryption, MAC/AEAD
Speed	Relatively expensive	Highly efficient
Examples	RSA, ElGamal, DH, ML-KEM	AES, ChaCha20

Fundamental Difference

Public-key: no pre-shared secret is required

Symmetric: a shared secret is required

RSA Cryptosystem

RSA Cryptosystem: An Introduction

RSA is a foundational public-key cryptosystem introduced in 1977 by **Rivest, Shamir, and Adleman**.

- **Encryption Scheme** Provides public-key encryption for establishing or transporting secret information.
- **Digital Signature:** Provides authentication and integrity using the same RSA key framework.

Underlying Assumption

Security is related to the computational difficulty of factoring

$$N = pq,$$

where p and q are large secret primes.

Historical Significance

RSA demonstrated that public-key cryptography could provide encryption and digital signatures without requiring a pre-shared secret key. It became one of the most widely deployed public-key cryptosystems.

Why Is Computing the RSA Private Exponent Difficult?

RSA publishes $N = pq, e,$ while keeping p, q secret.
The private exponent is $d = e^{-1} \pmod{\phi(N)}, \quad \phi(N) = (p-1)(q-1).$

If $\phi(N)$ were known, computing d would be easy using the Extended Euclidean Algorithm:

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

The difficulty is therefore obtaining $\boxed{\phi(N)}$ from the public value N.

For $N = pq$, knowing $\phi(N)$ immediately gives

$$\phi(N) = N - (p+q) + 1,$$

and hence

$$p + q = N - \phi(N) + 1.$$

Therefore p and q are the roots of

$$x^2 - (p+q)x + N = 0.$$

Conclusion

Computing $\phi(N)$ from N allows us to recover p and q. Thus, hiding the factorization hides the information needed to compute the RSA private exponent.

Extended Euclidean Algorithm

For integers a and b, the Extended Euclidean Algorithm computes

$$\gcd(a, b)$$

together with integers x, y satisfying

$$\boxed{ax + by = \gcd(a, b)}.$$

Modular Inverse

If

$$\gcd(a, n) = 1,$$

then

$$ax + ny = 1.$$

Taking modulo n gives

$$\boxed{x = a^{-1} \pmod{n}}.$$

Extended Euclidean Algorithm: Concrete Algorithm

Given integers a and b, compute $\gcd(a, b)$ and integers x, y such that $ax + by = \gcd(a, b)$.

Initialize

$$r_0 = a, \quad r_1 = b, \qquad s_0 = 1,\ s_1 = 0, \qquad t_0 = 0,\ t_1 = 1.$$

While $r_1 \neq 0$, compute

$$q_i = \left\lfloor \frac{r_{i-1}}{r_i} \right\rfloor,$$
$$r_{i+1} = r_{i-1} - q_i r_i,$$
$$s_{i+1} = s_{i-1} - q_i s_i,$$
$$t_{i+1} = t_{i-1} - q_i t_i.$$

When $r_1 = 0$,

$$\boxed{\gcd(a, b) = r_0}$$

and

$$\boxed{as_0 + bt_0 = r_0}.$$

If $\gcd(a, b) = 1$, then

$$\boxed{a^{-1} \equiv s_0 \pmod{b}}.$$

Extended Euclidean Algorithm: Example

Find $3^{-1} \pmod{26}$.

Euclidean algorithm:

$$26 = 8(3) + 2$$
$$3 = 1(2) + 1$$
$$2 = 2(1) + 0.$$

Back-substitute:

$$1 = 3 - 2$$
$$= 3 - (26 - 8(3))$$
$$= 9(3) - 26.$$

Thus,

$$9(3) + (-1)(26) = 1.$$

Taking modulo 26,

$$\boxed{3^{-1} \equiv 9 \pmod{26}}.$$

Verification:

$$3 \times 9 = 27 \equiv 1 \pmod{26}.$$

RSA Connection: For RSA, the same procedure computes $d = e^{-1} \pmod{\phi(N)}$.

RSA Encryption Scheme

It consists of the following algorithms.

1. **Key generation**
2. **Encryption** with public key
3. **Decryption** with secret key

RSA: Key Generation

RSA is based on the difficulty of factoring a large composite integer.

1. Choose two large random primes
$$p,\ q.$$
2. Compute
$$N = pq, \qquad \phi(N) = (p-1)(q-1).$$
3. Choose e such that
$$1 < e < \phi(N), \qquad \gcd(e, \phi(N)) = 1.$$
4. Compute the modular inverse
$$d = e^{-1} \pmod{\phi(N)}.$$

The keys are

$$\boxed{pk = (N, e)} \qquad \boxed{sk = (N, d)}$$

RSA: Encryption and Decryption

Given a message representative $m \in \mathbb{Z}_N$:

Encryption

Anyone can encrypt using $pk = (N, e)$:

$$\boxed{c = m^e \bmod N}$$

Decryption

The owner uses $sk = (N, d)$:

$$\boxed{m = c^d \bmod N}$$

Correctness follows from

$$(m^e)^d = m^{ed} \equiv m \pmod{N}.$$

Practical RSA

Textbook RSA is deterministic and must not be used directly. Practical RSA encryption uses randomized padding such as RSA-OAEP.

RSA Digital Signature Scheme

It consists of the following algorithms.

1. **Key generation**
2. **Sign** with secret key
3. **Verify** with secret key

RSA Digital Signature: Key Generation

Choose two large primes

$$p,\ q.$$

Compute

$$N = pq, \qquad \phi(N) = (p-1)(q-1).$$

Choose

$$e \quad \text{such that} \quad \gcd(e, \phi(N)) = 1.$$

Compute

$$d = e^{-1} \pmod{\phi(N)}.$$

The keys are

$$\boxed{pk = (N, e)} \qquad \boxed{sk = (N, d)}.$$

The public key is distributed, while d remains secret.

RSA Digital Signature: Signing

Given a message m, first compute its cryptographic hash

$$h = H(m).$$

The signer uses the secret exponent d to generate the signature:

$$\boxed{\sigma = h^d \bmod N}$$

Thus,

$$m \longrightarrow H(m) \longrightarrow \mathrm{Sign}_{sk}(H(m)).$$

The signature σ is sent along with the message m.

Important

In practical RSA signatures, textbook RSA is replaced by a secure encoding scheme such as **RSA-PSS**.

RSA Digital Signature: Verification

Given the message m and signature σ, the verifier computes

$$h = H(m).$$

Using the public key (N, e), compute

$$v = \sigma^e \bmod N.$$

Accept the signature iff

$$\boxed{v = H(m)}$$

Equivalently,

$$\text{Verify}_{pk}(m, \sigma) = 1$$

iff

$$\sigma^e \bmod N = H(m).$$

RSA Digital Signature: Correctness

During signing,

$$\sigma = H(m)^d \bmod N.$$

During verification,

$$\sigma^e \equiv (H(m)^d)^e = H(m)^{de} \pmod{N}.$$

Since

$$ed \equiv 1 \pmod{\phi(N)},$$

there exists an integer k such that

$$ed = 1 + k\phi(N).$$

Therefore, for $\gcd(H(m), N) = 1$,

$$H(m)^{ed} = H(m)^{1+k\phi(N)} \equiv H(m) \pmod{N}.$$

Hence,

$$\boxed{\sigma^e \bmod N = H(m)}$$

and the valid signature is accepted.

RSA: Classical Construction to Modern Practice

- **Core:** Modular exponentiation over
$$N = pq,$$
with security related to integer factorization.
- **Practical RSA:** Textbook RSA is insecure; use **RSA-OAEP** for encryption and **RSA-PSS** for signatures.
- **PKI:** RSA public keys can be certified by CAs and used for authentication and digital signatures.
- **Key Transport:** RSA was widely used to transport session keys, but modern protocols prefer ephemeral key agreement for forward secrecy.
- **Quantum Threat:** Shor's algorithm can efficiently factor large integers; RSA is therefore not post-quantum secure.
- **Transition:** RSA-based mechanisms are being replaced by post-quantum KEMs and signature schemes.

Perspective

RSA remains important in classical systems, but is not suitable for a post-quantum world.

How different Moduli works?

Correctness of RSA algorithms

Why Does RSA Use Two Different Moduli?

RSA uses two moduli for two different purposes.

1. Modulo N: RSA operates on the message space

$$\mathbb{Z}_N, \qquad N = pq.$$

Encryption and decryption use

$$c = m^e \bmod N, \qquad m = c^d \bmod N.$$

2. Modulo $\phi(N)$: It controls the relationship between the RSA exponents:

$$ed \equiv 1 \pmod{\phi(N)}, \qquad \phi(N) = (p-1)(q-1).$$

For $m \in \mathbb{Z}_N^*$, powers repeat modulo N according to the exponent period related to $\phi(N)$.

$$\boxed{N \to \text{message space} \qquad \phi(N) \to \text{exponent relationship}}$$

RSA Correctness: Case 1, $m \in \mathbb{Z}_N^*$

Let

$$N = pq, \qquad ed \equiv 1 \pmod{\phi(N)}, \qquad \phi(N) = (p-1)(q-1).$$

For

$$m \in \mathbb{Z}_N^*,$$

we have

$$\gcd(m, N) = 1.$$

By Euler's theorem,

$$m^{\phi(N)} \equiv 1 \pmod{N}.$$

Since

$$ed = 1 + k\phi(N),$$

we obtain

$$m^{ed} = m^{1+k\phi(N)} = m\left(m^{\phi(N)}\right)^k \equiv m \pmod{N}.$$

Thus,

$$\boxed{(m^e)^d \equiv m \pmod{N}}$$

Key Idea For $m \in \mathbb{Z}_N^*$, exponentiation is periodic modulo $\phi(N)$, so the inverse-exponent relation $ed \equiv 1 \pmod{\phi(N)}$ gives RSA correctness directly.

RSA Correctness: Case 2, $m \in \mathbb{Z}_N \setminus \mathbb{Z}_N^*$

Now suppose $m \notin \mathbb{Z}_N^*$. Since $N = pq$, this means

$$p \mid m \quad \text{or} \quad q \mid m.$$

The Euler argument cannot be applied directly because $\gcd(m, N) \neq 1$. Instead, work modulo p and q. Since $ed \equiv 1 \pmod{p-1}$, $\quad ed \equiv 1 \pmod{q-1}$, consider modulo p.

$$\begin{cases} p \nmid m: & m^{ed} \equiv m \pmod{p} \quad \text{(Fermat)} \\ p \mid m: & m^{ed} \equiv 0 \equiv m \pmod{p} \end{cases}$$

Similarly,

$$\begin{cases} q \nmid m: & m^{ed} \equiv m \pmod{q} \\ q \mid m: & m^{ed} \equiv 0 \equiv m \pmod{q}. \end{cases}$$

Therefore,

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

By the Chinese Remainder Theorem,

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

References

Textbook:

1. Douglas R. Stinson and Maura Paterson. *Cryptography: Theory and Practice*. 4th Edition. Chapman & Hall/CRC Press, 2019. ISBN: 9780367148591.

Acknowledgment:

- The figures in this work were generated with the assistance of ChatGPT (OpenAI).

Dr. Laltu Sardar, Assistant Professor, IISER Thiruvananthapuram
laltu.sardar@iisertvm.ac.in, laltu.sardar.crypto@gmail.com
Course webpage: https://laltu-sardar.github.io/courses/26_crypto.html.