

Public Key Cryptography (RSA)

Prepared by Dr. Sumaya Alkubaisi

FNT625
Cybersecurity Concepts

Lecture 8



Lecture

8

Learning Goals

By the end of this lecture, you will be able to:

- Explain how RSA works and why it's important
- Generate RSA public & private keys conceptually
- Perform basic RSA encryption and decryption
- Understand RSA's role in digital signatures
- Identify RSA strengths, limitations, and real-world uses

Introduction to Public Key Cryptography (PKC)

- Public Key Cryptography is an encryption approach that uses two different keys: a public key for encryption and a private key for decryption.
- It solves a major limitation of symmetric encryption: secure key sharing.
- PKC enables secure communication between parties without needing a shared secret key in advance.
- In Public Key Cryptography, the public key is not secret — you can:
 - post it online,
 - send it by email,
 - publish it on a website,
 - share it with any user who wants to send you encrypted data.
- Even if everyone knows your public key, they cannot decrypt messages — only the private key (which you keep secret) can do that.

Lecture

8

Why Public Key Cryptography?

- Public Key Cryptography was developed to address key distribution problems in symmetric encryption.
- Before PKC, both sender and receiver needed the same secret key—creating risks during key exchange.
- PKC eliminates this risk: the public key is shared openly, while the private key stays confidential.
- We need the public key so that others can securely send you encrypted data without needing a secret key from you first.
- With a public key:
 - Anyone can use it to encrypt a message for you.
 - Only you can decrypt it using your private key.

So, the public key enables safe communication with people you've never exchanged a secret key with.

Lecture

8

How PKC Works (High-Level Concept)

PKC uses a key pair:

Public Key: Used to encrypt data or verify a digital signature

Private Key: Used to decrypt data or create a digital signature

Anyone can send encrypted data to the key owner using the public key, but only the owner can decrypt it with the private key.

Lecture

8

Public Key vs Symmetric Key Encryption

Feature	Symmetric (Secret-Key)	Public Key (Asymmetric)
Keys Used	One shared key	Two keys (public + private)
Speed	Very fast	Slower
Best Use	Bulk data encryption	Key exchange, authentication, digital signatures
Key Distribution	Difficult	Easy (public key shared openly)

Lecture
8

Core Applications of Public Key Cryptography

Public key cryptography is widely used for:

- Secure Key Exchange – Send symmetric keys securely (Symmetric encryption is fast and efficient, but both sides need the same secret key. Sharing that key over the internet is risky.)
- Digital Signatures – Verify authenticity & integrity
- Authentication – Prove identity of users/devices
- Secure Communication – HTTPS, email encryption, VPNs

What is RSA?

- RSA (Rivest–Shamir–Adleman) is the most widely used public key cryptosystem. Introduced in 1977, it is based on the mathematical challenge of factoring very large prime numbers.
- RSA supports encryption, decryption, digital signatures, and key exchange.
- RSA became the foundation of secure communication on the internet.
- It enables secure web browsing (HTTPS), online banking, VPNs, and digital certificates.
- Even today, RSA remains widely used, especially in combination with symmetric encryption.

Lecture

8

The Mathematics Behind RSA (Simple Idea)

RSA relies on these core mathematical principles:

- Choose two large prime numbers
- Multiply them → result is very large (hard to factor)
- Use modular arithmetic to create public and private keys

Security depends on the fact that factoring a large number (2048-bit) is practically impossible with current computing power.

Lecture

8

Key Concepts Needed for RSA

- **Prime Numbers** – numbers only divisible by 1 & itself; RSA uses two large primes to build the keys.
- **Greatest Common Divisor (GCD)** – needed to choose RSA exponents; Used to ensure the chosen RSA exponent (e) works correctly with $\phi(n)$.
- **Modular Arithmetic** – used in RSA operations;
- **Euler's Totient (ϕ)** – A function used in RSA key generation to determine how many integers are coprime with n .

Lecture

8

RSA Key Pair – Overview

An RSA Key Pair consists of:

- **Public Key (e, n)** → shared with everyone
- **Private Key (d, n)** → kept secret by the owner

Where:

- n = product of two large primes
- e = public exponent (encryption)
- d = private exponent (decryption)

Encrypted with public key, decrypted with private key.

RSA Key Generation (Overview)

RSA key generation involves 5 main steps:

1. Choose two large prime numbers (p and q)
2. Compute $n = p \times q$
3. Compute Euler's Totient: $\phi(n) = (p-1)(q-1)$
4. Choose a public exponent e such that $\gcd(e, \phi(n)) = 1$
5. Compute private key d such that $(d \times e) \bmod \phi(n) = 1$

This produces the public key (e, n) and private key (d, n) .

Step 1 – Select Two Large Prime Numbers

- Choose two large primes p and q (hundreds of digits long).
- They must be random, unpredictable, and different sizes for security.
- These primes must remain secret.

Example (Tiny Numbers for Learning):

$p = 17, q = 11$

Step 2 – Compute $n = p \times q$

- Multiply p and q to form n , which becomes part of both keys.

Example:

$$p = 17, q = 11$$

$$n = 17 \times 11 = 187$$

- n is public, but knowing n shouldn't expose p and q .
- Security relies on the difficulty of factoring n into its prime components.

Note: these are small primes used only for learning and examples.

In real RSA, primes are extremely large (hundreds of digits), e.g.:

1024-bit RSA: p and q are ~150–160 digits each

2048-bit RSA: p and q are ~300+ digits each

Step 3 – Compute Euler's Totient $\phi(n)$

- Euler's Totient $\phi(n)$ counts numbers from 1 to $n-1$ that are coprime with n .

For RSA:

$$\phi(n) = (p - 1) \times (q - 1)$$

Example:

$$p = 17, q = 11$$

$$\phi(n) = 16 \times 10 = 160$$

- ❖ Two numbers are coprime if their GCD (Greatest Common Divisor) = 1.
Meaning: they share no common factors except 1.

Step 4 – Choose Public Exponent e

- Choose an integer e such that:
 - $1 < e < \phi(n)$
 - $\gcd(e, \phi(n)) = 1$ (meaning e and $\phi(n)$ share no common factors)
- Common secure values: 65537, 17, or 3 (65537 is the standard).

Example:

$\phi(n) = 160 \rightarrow$ choose $e = 7$ (since $\gcd(7, 160) = 1$)

Step 5 – Compute Private Key d

Compute d so that:

$$(d \times e) \bmod \phi(n) = 1$$

Meaning: d is the modular inverse of e.

Example:

$$e = 7, \phi(n) = 160$$

$$\text{Find } d \text{ such that } (d \times 7) \bmod 160 = 1 \rightarrow d = 23$$

Now keys are:

- Public Key: (e=7, n=187)
- Private Key: (d=23, n=187)

Summary of RSA Key Generation (Example)

Value $p = 17, q = 11$ $n = 187$ $\phi(n) = 160$ $e = 7$ $d = 23$ **Meaning**

Secret primes

Public (used for encryption/decryption)

Used for key calculations

Public exponent

Private exponent

Public Key = (7, 187)

Private Key = (23, 187)

RSA Encryption (How It Works)

Sender encrypts message M using recipient's public key (e, n):

Formula:

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

Where:

M = plaintext (as a number)

C = ciphertext

Only the receiver with private key can decrypt it.

Example (using previous keys, Public Key = (7, 187)):

$$M = 5$$

$$C = 5^7 \bmod 187 = 78$$

Encrypted message = 78

Lecture

8

RSA Decryption (How It Works)

- Receiver decrypts ciphertext C using private key (d, n):

Formula:

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

Example:

using previous keys: Private Key = (23, 187)

C = 78

$M = 78^{23} \bmod 187 = 5$

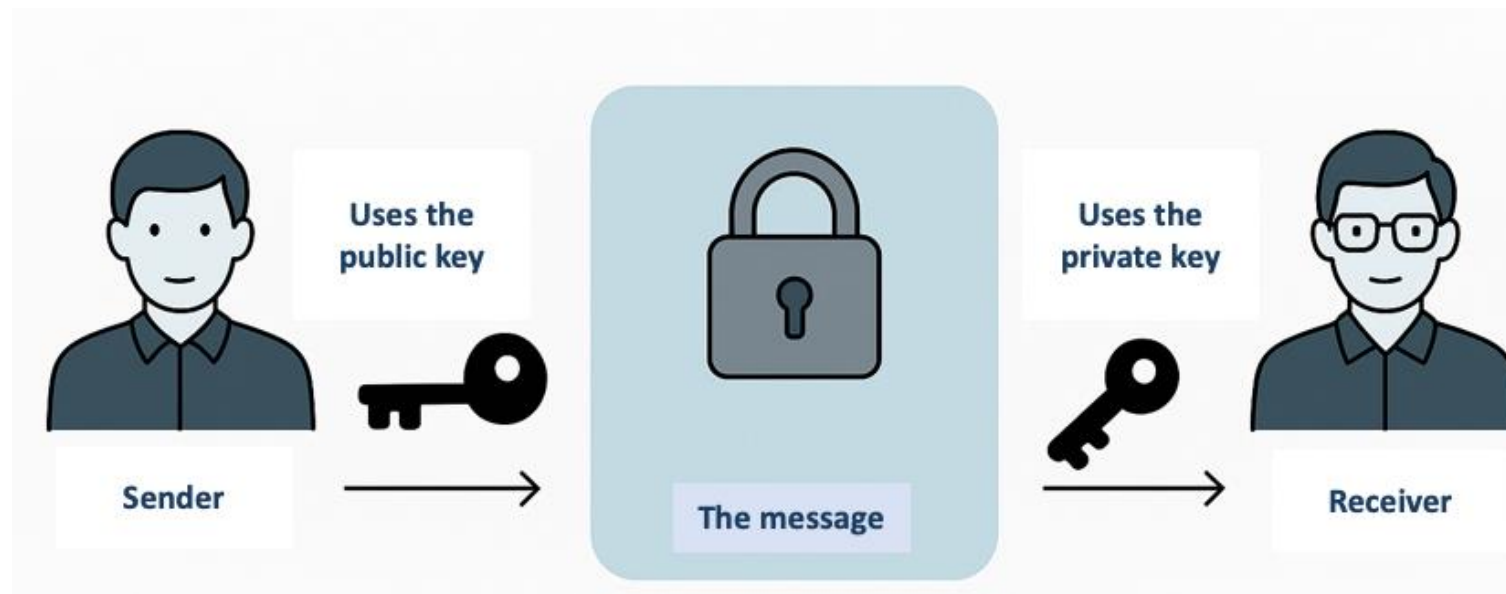
Recovered plaintext = 5

- ❖ RSA never computes the full power directly — it uses modular exponentiation, reducing the value at each step to stay within the modulus.

Lecture

8

RSA Decryption (How It Works)



- The sender uses the public key to encrypt the message.
- The message is transmitted over the channel in encrypted form.
- The receiver uses the private key to decrypt and recover the original text.

Why RSA Works

- RSA works because of a special math relationship between the public key (e) and the private key (d) – properties of modular arithmetic and Euler's Theorem.
- When we generate the keys, we choose e and d so that they undo each other under mod $\phi(n)$.

This means:

- Encryption with e can be reversed only by decryption with d, and vice-versa
- This is based on Euler's Theorem, which ensures:

$$M^{(e \times d)} \bmod n = M$$

So when we:

1. Encrypt a message M using the public key (e): $C = M^e \bmod n$
2. Decrypt using the private key (d): $M = C^d \bmod n$

Then we get back the original message.

Lecture

8

RSA in Real-World Communication

RSA is widely used to establish secure communication channels, especially during key exchange.

Examples where RSA is used behind the scenes:

- Browser connecting to HTTPS website (TLS handshake — Setup process of HTTPS to establish a secure connection.)
- Secure email protocols (PGP, S/MIME — Both use RSA to secure email messages and verify the identity of the sender.)
- VPN authentication
- Digital certificates and signatures

RSA often encrypts only **the symmetric key**, not entire data, because RSA is slower.

RSA for Digital Signatures

RSA for Digital Signatures RSA is used not only for encryption but also for authenticating the sender.

Digital Signature Process:

- Sender signs a message by encrypting a hash (A digital fingerprint of data, used for security) with their private key.
- Receiver verifies by decrypting with the public key.
- Proves authenticity, integrity, and non-repudiation.

Encryption vs Digital Signature (Key Difference)

For RSA Encryption (Confidentiality):

Action	Key Used	Why
Encrypt message	Public Key	Anyone can lock a message for the owner
Decrypt message	Private Key	Only the owner can unlock and read it

So here the analogy is:

Public key = lock

Private key = unlock

Lecture

8

Encryption vs Digital Signature (Key Difference)

For RSA Digital Signature (Authenticity & Integrity):

Action	Key Used	Why
“Sign” (encrypt the hash)	Private Key	Proves the sender is the owner of the private key
Verify signature	Public Key	Anyone can verify authenticity

Digital Signatures Work in Reverse:

- The private key signs (locks the identity proof)
- The public key verifies (unlocks and checks it)

This ensures:

- Authenticity – only the owner could have signed it
- Integrity – if message changes, signature becomes invalid
- Non-repudiation – sender cannot deny sending it

Strengths of RSA

RSA is popular due to several advantages:

- Enables secure communication without prior key sharing
- Provides confidentiality + authentication
- Mature and well-studied algorithm since 1977
- Widely supported in protocols, software, and hardware

RSA revolutionized cryptography by making secure communication possible between strangers — because RSA allows secure communication without needing to share a secret key first, making it safe for strangers to communicate.

Weaknesses & Risks of RSA

RSA has limitations and vulnerabilities:

- Slow performance – inefficient for encrypting large data
- Security depends on key size; small keys can be broken
- Vulnerable to side-channel attacks (timing, power analysis)
- Requires high-quality random prime numbers
- If p and q leak $\rightarrow$ entire system collapses

Modern systems combine RSA + symmetric keys for efficiency.

Lecture

8

RSA Key Sizes & Security Levels

RSA security increases with key size. Today, 2048-bit RSA is considered the minimum secure size. Governments & banking often use 3072-bit or 4096-bit keys.

1024-bit RSA – Considered not secure today; used only in legacy systems.

2048-bit RSA – Provides strong security and is the current standard for most secure applications.

3072–4096-bit RSA – Offers very strong protection, typically used in high-security environments such as banks and government systems.

Lecture

8

Why RSA Must Use Large Prime Numbers?

If p and q are small or predictable, attackers can factor n easily.

Secure RSA primes must be:

- Hundreds of digits long
- Randomly generated
- Not mathematically related

Modern RSA uses primes of ~ 300 digits each to prevent factorization attacks.

Lecture

8

Common Attacks on RSA

Attackers try to break RSA by:

- **Factoring n** — the most direct attack: if an attacker factors $n=p \times q$ they can compute $\phi(n)$ and recover the private key.
Mitigation: use sufficiently large primes (≥ 2048 -bit RSA for today's standards) and strong random prime generation.
- **Poor key generation (weak or reused primes)** — predictable, small, or reused primes make factoring or key recovery trivial.
Mitigation: use cryptographically secure Random Number Generators (RNGs), avoid reused primes/keys, and validate primes.
- **Side-channel attacks** — attackers measure timing, power consumption, electromagnetic leaks, or cache behavior to infer private key bits.
Mitigation: use constant-time algorithms, blinding (e.g., RSA blinding — adding randomness), hardware countermeasures, and side-channel resistant libraries.

Common Attacks on RSA

- **Chosen-ciphertext attack (CCA):** the attacker tricks the system into decrypting specially made messages to learn secret information.
- **Mitigation:** use safe padding (adding extra bits to a message before encryption to make it safer), like Optimal Asymmetric Encryption Padding method (OAEP) and follow secure RSA rules.
- **Mathematical/algorithmic attacks (small exponents, low public exponent misuse)** — using very small public key number (e) or improperly padded messages can enable attacks (e.g., low-exponent broadcast attacks).
- **Mitigation:** choose safe exponent values, proper padding, and avoid encrypting identical messages to multiple recipients without padding.
- **Implementation bugs** — coding mistakes (incorrect padding checks, error messages) leak info that breaks security.
- **Mitigation:** use well-reviewed crypto libraries, follow standards, and perform security audits.

Lecture

8

Common Attacks on RSA

- **Fault injection / hardware tampering** — inducing faults during computation can reveal private key material.
- **Mitigation:** detect and handle computation faults, use redundancy and integrity checks.
- **Quantum attacks (future threat)** — large-scale quantum computers would break RSA via Shor's algorithm.
Mitigation: monitor post-quantum crypto developments and plan migration to quantum-resistant algorithms when required.

RSA remains secure if implemented properly.

Why We Don't Encrypt Large Files with RSA

RSA is computationally heavy and slow for large data.

Instead, real systems use:

- RSA to encrypt a symmetric key
- Symmetric algorithm (AES) to encrypt the actual data

This is called **Hybrid Encryption** and is used in HTTPS, email security, and secure messaging.

Lecture

8

Hybrid Encryption (RSA + AES)

Hybrid encryption combines the strengths of both algorithms:

- RSA: Secure key exchange
- AES: Fast encryption of large data

Process:

- Generate a random AES key
- Encrypt AES key using RSA
- Send encrypted AES key + encrypted data

This ensures speed and security — commonly used in banking TLS transactions.

RSA in Banking & FinTech

RSA plays a key role in financial cybersecurity:

- Secures online banking logins (TLS handshake)
- Ensures secure communication between ATM, Point of Sale (POS), and bank servers
- Protects digital payment transactions
- Used in SWIFT network authentication
- Secures API communication in fintech systems

Banks often use RSA with 2048 or 3072-bit keys for regulatory compliance.

Lecture

8

RSA vs Symmetric Encryption

Feature	RSA (Asymmetric)	AES (Symmetric)
Keys Used	Public & Private	One shared secret key
Speed	Slower	Very fast
Use Case	Key exchange, digital signatures	Data encryption
Security Depends On	Factoring large primes	Key size & algorithm

They complement each other, not replace each other.

When Should RSA Be Used?

RSA is ideal when:

- Two parties have no pre-shared key
- Authentication of identity is required
- Secure key exchange is needed
- Protecting credentials, tokens, or digital signatures

Not recommended for encrypting full files or databases → use AES.

When RSA Should NOT Be Used

Avoid using RSA for:

- Encrypting large data or storage
- Low-power devices (IoT) — RSA is slow & heavy
- Systems with weak random number generation
- Keys under 2048 bits

Low-power devices (IoT) — Small devices like IoT sensors, smart watches, or embedded gadgets that don't have enough processing power to handle slow RSA computations efficiently.

Lecture

8

Why RSA Keys Must Be Rotated

Key rotation means generating new RSA key pairs periodically.

Reasons to rotate keys:

- Reduces long-term exposure if key leaks
- Limits damage of a compromised key
- Meets compliance and banking standards (PCI-DSS, NIST)

Best practice: rotate RSA keys every 1–2 years.

Lecture

8

Hands-On Exercise (Optional)

Try a basic RSA key generation using Python:

```
from Crypto.PublicKey import RSA
key = RSA.generate(1024)
public_key = key.publickey().export_key()
private_key = key.export_key()
print(public_key)
print(private_key)
```

Lecture

8

Recap

- RSA uses two keys: Public (encryption) & Private (decryption)
- Based on large prime factorization & modular arithmetic
- Supports encryption + digital signatures
- RSA is secure when using ≥ 2048 -bit keys
- Used for TLS, banking, secure email, and authentication
- Often combined with AES in hybrid encryption for efficiency

References

- William Stallings, Cryptography and Network Security, 8th Edition (2023)
 - William Stallings, Network Security Essentials, 6th Edition (2022)
 - Cybersecurity Tools and Techniques (2022) – uploaded reference
 - NIST Special Publication 800-57 – Key Management Guidelines
 - RFC 8017 – PKCS #1: RSA Cryptography Standard
 - Research: Rivest, Shamir & Adleman, “A Method for Obtaining Digital Signatures and Public-Key Cryptosystems” (MIT, 1978)