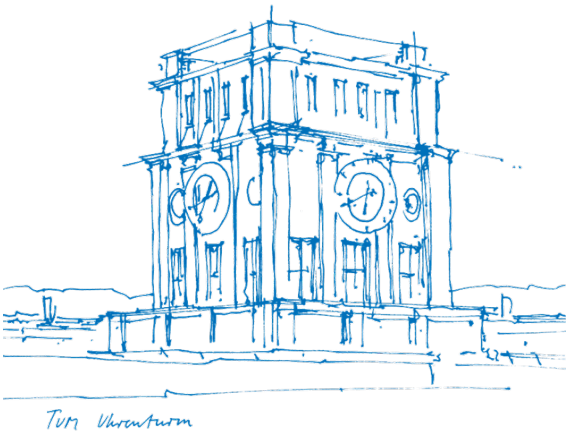
TUM School of Computation, Information and Technology
Technical University of Munich

# Intro to RSA

**Kerui Ren**

TUM School of Computation, Information and Technology
Technical University of Munich

January 14th, 2026



# Outline

# Key generation

1. choose 2 large prime numbers $p, q$
2. calculate $n = pq$
3. calculate $\varphi(n)$ (= $(p-1)(q-1)$), where $\varphi(m) := |\{\, 1 \leq k \leq m \mid \gcd(k, m) = 1 \,\}|$ is the Euler's totient function
4. choose $e \in \{1, 2, ..., \varphi(n) - 1\}$, such that $gcd(\varphi(n), e) = 1$.
5. calculate $d$, such that $ed \equiv 1$ mod $\varphi(n)$.

- public key: n,e
- private key: d

# encryption & decryption

Let $m, c$ denote the message and ciphertext *resp.*

- encryption

$$c = m^e \bmod n.$$

- decryption

$$m = c^d \bmod n.$$

# encryption & decryption

**Theorem 1.1: Euler's Theorem**

Let $n$ be a positive integer, $a$ an integer with $(a, n) = 1$ and let $\varphi$ be Euler's totient function. Then

$$a^{\varphi(n)} \equiv 1 \pmod{n}.$$

So,

$$c^d = m^{ed} = m^{k\varphi(n)} \cdot m \equiv 1 \cdot m \mod n$$

## Choose e

- not too big
- not too small

# Outline

# Chinese Remainder Theorem

Can we speed up the decryption?

Yes!

**Theorem 2.1: Chinese Remainder Theorem**

In case we have system of modular eqations:

$$\begin{cases} x \equiv a \pmod{p} \\ x \equiv b \pmod{q} \end{cases}$$

If $p, q$ are coprime, we have an unique solution in $\mathbb{Z}_n$ ($n := pq$).

# RSA-CRT decryption

Besides $d, p, q$, we precompute three CRT parameters:

$$d_p = d \pmod{p-1}$$
$$d_q = d \pmod{q-1}$$
$$q_{\text{inv}} = q^{-1} \pmod{p}$$

During decryption, first compute

$$m_p = c^{d_p} \pmod{p}$$
$$m_q = c^{d_q} \pmod{q}$$

Then use some CRT Algorithm to recombine the message:

$$m \equiv c^d \bmod n$$

# Outline

# PEM

PEM (Privacy-Enhanced Mail) is a widely used text-based container format for representing certificates and public/private keys.

In practice, a PEM file typically contains the Base64-encoded representation of some underlying binary data (most commonly DER-encoded ASN.1), wrapped with header and footer lines like

```
-----BEGIN PUBLIC KEY-----
-----END PUBLIC KEY-----
```

# Outline

# Factorize n

## Fermat's factorization

If we have

$$n = s^2 - r^2$$

then

$$n = (s - r)(s + r) = pq.$$

So it's equivalent to test if

$$s^2 - n$$

is a square number for arbitrary $s$.

# Factorize n

## Fermat's factorization

**Algorithm**:

1. let $t := \lceil \sqrt{n} \rceil$
2. for $i \in \mathbb{Z}_{\geq 0}$, test if $(t+i)^2 - n$ is a square number
   (a) if yes, then we have factorized $n$.
   (b) if no, then continue.

**But, this algorithm does not always success!**

In the worse case we will come to $t + i = \frac{n+1}{2}$, which means that we have

$$n = (\frac{n+1}{2})^2 - (\frac{n-1}{2})^2$$

leads to the trivial case

$$n = 1 \cdot n$$

# Wiener's Attack

**Theorem 4.1: Wiener's Theorem**

Let $p, q$ be prime numbers with $q < p < 2q$ and $n = pq$. Let $e < \varphi(n)$ be coprime to $\varphi(n)$ and $d \equiv e^{-1}$ mod $\varphi(n)$.
If

$$d < \frac{n^{1/4}}{3},$$

then given $n, e$, one can efficiently recover $p, q$. (in $O((log(n))^2)$ )

# Common modulus attack

Suppose we have 2 ciphertexts:

$$c_1 = m^{e_1} \mod n,$$
$$c_2 = m^{e_2} \mod n$$

with same modulus $n$, but with different $e$.

If moreover $e_1, e_2$ are coprime (i.e. gcd$(e_1, e_2) = 1$ ), we can recover the original message.

# Common modulus attack

**Attack:**

1. find $x, y$ such that
$$x \cdot e_1 + y \cdot e_2 = 1$$
(for example use extended Euclidean Algorithm)
2. compute
$$m = \left( (c_1^x \cdot c_2^y) \mod n \right)$$

**Why?**

$$(c_1^x \cdot c_2^y) \equiv (m^{e_1})^x \cdot (m^{e_2})^y \equiv m^{x \cdot e_1 + y \cdot e_2} \equiv m \mod n$$

# Thanks for listening!

Questions?