

2. Units. Euler function. CRT. RSA.

A. Ushakov

MA503, September 14, 2022

Contents

- Linear congruences.
- Units modulo n .
- Multiplicative inverse modulo n .
- Euler function φ .
- Some properties of units.
- Fermat's little theorem.
- Chinese remainder theorem.
- Modular exponentiation.
- Public key cryptosystems.
- RSA.
- RSA problem.
- RSA: basic security analysis.

Linear congruence: $ax \equiv_n c$

Question. Given $a, c \in \mathbb{Z}$ and $n \geq 1$ determine if there exists $x \in \mathbb{Z}$ such that $ax \equiv_n c$.

We can modify the congruence as follows:

$$\begin{aligned} ax \equiv_n c &\Leftrightarrow n \mid (c - ax) \\ &\Leftrightarrow c - ax = qn \text{ for some } q \in \mathbb{Z} \\ &\Leftrightarrow c = ax + qn \text{ for some } q \in \mathbb{Z}. \end{aligned}$$

Now, can we express c as an integral linear combination of a and n ?

Corollary

$ax \equiv_n c$ has a solution $\Leftrightarrow \gcd(a, n) \mid c$.

- $2x \equiv_5 3$ has a solution because $\gcd(2, 5) = 1 \mid 3$.
- $6x \equiv_{14} 5$ has no solution because $\gcd(6, 14) = 2 \nmid 5$.
- $11x \equiv_{19} 8$ has a solution because $\gcd(11, 19) = 1 \mid 8$.

A particular solution can be found by solving a linear Diophantine equation.

- $2x \equiv_5 3 \Leftrightarrow 2x + 5q = 3$ that has a solution $x = 4, q = -1$.
- $11x \equiv_{19} 8 \Leftrightarrow 11x + 19q = 8$ that has a solution $x = -1, q = 1$.

This is how a computer finds a solution. Below we discuss several tricks/ideas to simplify finding a solution.

Linear congruences: properties

x satisfies $(ad)x \equiv_{nd} cd \Leftrightarrow x$ satisfies $ax \equiv_n c$.

$$(ad)x \equiv_{nd} cd \Leftrightarrow (ad)x + (nd)q = cd \Leftrightarrow ax + nq = c \Leftrightarrow ax \equiv_n c.$$

Suppose that $\gcd(d, n) = 1$. Then x satisfies $(ad)x \equiv_n cd \Leftrightarrow x$ satisfies $ax \equiv_n c$.

$$(ad)x \equiv_n cd \Leftrightarrow d^{-1}(ad)x \equiv_n d^{-1}cd \Leftrightarrow ax \equiv_n c,$$

where d^{-1} is the multiplicative inverse of d modulo n (discussed later).

It is convenient to denote a solution to $ax \equiv_n c$ by $\frac{c}{a}$.

Example. Find a solution for $14x \equiv_{22} 6$.

$$\bullet 14x \equiv_{22} 6 \Leftrightarrow 7x \equiv_{11} 3.$$

$$\bullet x = \frac{3}{7} \equiv_{11} \frac{3+11}{7} = 2.$$

Example. Find a solution for $12x \equiv_{51} 30$.

$$\bullet 12x \equiv_{51} 30 \Leftrightarrow 4x \equiv_{17} 10.$$

$$\bullet x = \frac{10}{4} \equiv_{17} \frac{5}{2} \equiv_{17} \frac{5+17}{2} = 11.$$

Linear congruences: properties

x satisfies $(ad)x \equiv_{nd} cd \Leftrightarrow x$ satisfies $ax \equiv_n c$.

$$(ad)x \equiv_{nd} cd \Leftrightarrow (ad)x + (nd)q = cd \Leftrightarrow ax + nq = c \Leftrightarrow ax \equiv_n c.$$

Suppose that $\gcd(d, n) = 1$. Then x satisfies $(ad)x \equiv_{nd} cd \Leftrightarrow x$ satisfies $ax \equiv_n c$.

$$(ad)x \equiv_n cd \Leftrightarrow d^{-1}(ad)x \equiv_n d^{-1}cd \Leftrightarrow ax \equiv_n c,$$

where d^{-1} is the multiplicative inverse of d modulo n (discussed later).

It is convenient to denote a solution to $ax \equiv_n c$ by $\frac{c}{a}$.

Example. Find a solution for $14x \equiv_{22} 6$.

$$\bullet 14x \equiv_{22} 6 \Leftrightarrow 7x \equiv_{11} 3.$$

$$\bullet x = \frac{3}{7} \equiv_{11} \frac{3+11}{7} = 2.$$

Example. Find a solution for $12x \equiv_{51} 30$.

$$\bullet 12x \equiv_{51} 30 \Leftrightarrow 4x \equiv_{17} 10.$$

$$\bullet x = \frac{10}{4} \equiv_{17} \frac{5}{2} \equiv_{17} \frac{5+17}{2} = 11.$$

Units modulo n

Recall that $\mathbb{Z}_n = \{[0], [1], \dots, [n-1]\}$.

Definition

$[a] \in \mathbb{Z}_n$ is a **unit** if there exists $[b] \in \mathbb{Z}_n$ such that

$$[a] \cdot [b] = [1],$$

in which case we say that $[b]$ is the **multiplicative inverse** of $[a]$.

U_n – the set of all units modulo n .

Lemma

$$[a] \in U_n \Leftrightarrow \gcd(a, n) = 1$$

To simplify notation, we often omit brackets.

$$U_n = \{a \mid 0 \leq a \leq n-1, \gcd(a, n) = 1\}.$$

$$U_1 = \{0\},$$

$$U_5 = \{1, 2, 3, 4\}$$

$$U_9 = \{1, 2, 4, 5, 7, 8\},$$

$$U_2 = \{1\},$$

$$U_6 = \{1, 5\},$$

$$U_{10} = \{1, 3, 7, 9\},$$

$$U_3 = \{1, 2\}$$

$$U_7 = \{1, 2, 3, 4, 5, 6\},$$

$$U_{11} = \{1, \dots, 10\},$$

$$U_4 = \{1, 3\}$$

$$U_8 = \{1, 3, 5, 7\},$$

$$U_{12} = \{1, 5, 7, 11\}.$$

Finding multiplicative inverse modulo n

To find the multiplicative inverse of a modulo n we solve the congruence $ax \equiv_n 1$.

Example. Find 2^{-1} modulo 13.

$$\frac{1}{2} \equiv_{13} \frac{1+13}{2} = 7.$$

Example. Find 6^{-1} modulo 17.

$$\frac{1}{6} \equiv_{17} \frac{1+17}{2} = 3.$$

Example. Find 11^{-1} modulo 27.

$$\frac{1}{11} \equiv_{27} \frac{1+27}{11-27} \equiv_{27} \frac{28}{-16} \equiv_{27} \frac{-7}{4} \equiv_{27} \frac{27-7}{4} = 5.$$

Example. Find 13^{-1} modulo 58.

$$\frac{1}{13} \equiv_{58} \frac{1-58}{13-58} \equiv_{58} \frac{-57}{-45} \equiv_{58} \frac{19}{15} \equiv_{58} \frac{19-58}{15} = \frac{-39}{15} \equiv_{58} \frac{-13}{5} \equiv_{58} \frac{58-13}{5} = \frac{45}{5} = 9.$$

Euler function φ

Definition

The **Euler's function** $\varphi : \mathbb{N} \rightarrow \mathbb{N}$ is defined by $\varphi(n) = |U_n|$.

$\varphi(1) = 1,$	$\varphi(4) = 2,$	$\varphi(7) = 6,$	$\varphi(10) = 4,$
$\varphi(2) = 1,$	$\varphi(5) = 4,$	$\varphi(8) = 4,$	$\varphi(11) = 10,$
$\varphi(3) = 2,$	$\varphi(6) = 2,$	$\varphi(9) = 6,$	$\varphi(12) = 4.$

Lemma

Let p be a prime number and $n \in \mathbb{N}$. Then $\varphi(p^n) = p^{n-1}(p-1)$.

Lemma

If m and n are coprime, then $\varphi(mn) = \varphi(m)\varphi(n)$.

Theorem

If $\text{PPF}(n) = p_1^{s_1} \dots p_k^{s_k}$, then $\varphi(n) = \prod_{i=1}^k (p_i^{s_i} - p_i^{s_i-1})$.

For instance, $\text{PPF}(360) = 2^3 \cdot 3^2 \cdot 5^1$ gives:

$$\varphi(360) = (2^3 - 2^2) \cdot (3^2 - 3^1) \cdot (5^1 - 5^0) = 4 \cdot 6 \cdot 4 = 96.$$

The problem to compute $\varphi(n)$ is computationally hard for large values of n .

Euler function φ

Definition

The **Euler's function** $\varphi : \mathbb{N} \rightarrow \mathbb{N}$ is defined by $\varphi(n) = |U_n|$.

$$\varphi(1) = 1,$$

$$\varphi(4) = 2,$$

$$\varphi(7) = 6,$$

$$\varphi(10) = 4,$$

$$\varphi(2) = 1,$$

$$\varphi(5) = 4,$$

$$\varphi(8) = 4,$$

$$\varphi(11) = 10,$$

$$\varphi(3) = 2,$$

$$\varphi(6) = 2,$$

$$\varphi(9) = 6,$$

$$\varphi(12) = 4.$$

Lemma

Let p be a prime number and $n \in \mathbb{N}$. Then $\varphi(p^n) = p^{n-1}(p-1)$.

Lemma

If m and n are coprime, then $\varphi(mn) = \varphi(m)\varphi(n)$.

Theorem

If $\text{PPF}(n) = p_1^{s_1} \dots p_k^{s_k}$, then $\varphi(n) = \prod_{i=1}^k (p_i^{s_i} - p_i^{s_i-1})$.

For instance, $\text{PPF}(360) = 2^3 \cdot 3^2 \cdot 5^1$ gives:

$$\varphi(360) = (2^3 - 2^2) \cdot (3^2 - 3^1) \cdot (5^1 - 5^0) = 4 \cdot 6 \cdot 4 = 96.$$

The problem to compute $\varphi(n)$ is computationally hard for large values of n .

Some properties of units

Proposition

Product of two units modulo n is a unit.

$$\begin{array}{l} x \in U_n \\ y \in U_n \end{array} \Rightarrow \begin{array}{l} ax \equiv_n 1 \quad (\text{for some } x) \\ by \equiv_n 1 \quad (\text{for some } y) \end{array} \Rightarrow (ab)(xy) \equiv_n 1 \Rightarrow ab \in U_n.$$

Proposition

Multiplicative inverse of a unit modulo n is a unit.

If x is a multiplicative inverse of a , then $ax \equiv_n 1$, then a is a multiplicative inverse of x , then x is a unit modulo n .

Fermat's little theorem

Theorem

If a is a unit modulo n , then $a^{\varphi(n)} \equiv_n 1$.

- Let $U_n = \{[x_1], \dots, [x_k]\}$, where $k = \varphi(n)$.
- Consider the function $\varphi_a : U_n \rightarrow U_n$ defined by $[x_i] \xrightarrow{\varphi_a} [ax_i]$.
- φ_a is injective because

$$\varphi([x_i]) = \varphi([x_j]) \Rightarrow [ax_i] = [ax_j] \Rightarrow [x_i] = [a]^{-1}[ax_i] = [a]^{-1}[ax_j] = [x_j].$$

- Every $[x_i] \in U_n$ has a preimage because $\varphi_a([a]^{-1}[x_i]) = [x_i]$ and, hence, φ_a is surjective.
- Hence, φ_a is bijective and $\{[x_1], \dots, [x_k]\} = \{[ax_1], \dots, [ax_k]\}$.
- Hence, $\prod_{i=1}^k [x_i] = \prod_{i=1}^k [ax_i]$ which implies that

$$\begin{aligned} \prod_{i=1}^k x_i &\equiv_n \prod_{i=1}^k ax_i = a^{\varphi(n)} \prod_{i=1}^k x_i \Rightarrow n \mid (a^{\varphi(n)} - 1) \prod_{i=1}^k x_i \quad (\text{where } \gcd(n, x_i) = 1) \\ &\Rightarrow n \mid a^{\varphi(n)} - 1 \\ &\Rightarrow a^{\varphi(n)} \equiv_n 1. \end{aligned}$$

Chinese remainder theorem (CRT)

Theorem

Assume $n_1, \dots, n_k \in \mathbb{Z}$ are pairwise coprime. Then for any $c_1, \dots, c_k \in \mathbb{Z}$ solutions to the system

$$\begin{cases} x \equiv_{n_1} c_1, \\ \dots \\ x \equiv_{n_k} c_k, \end{cases} \quad (1)$$

form a single congruence class modulo $n = n_1 \dots n_k$.

Proof.

Existence of a solution:

- Define $m_i = n/n_i = n_1 \dots n_{i-1} n_{i+1} \dots n_k$.
- Clearly, $\gcd(m_i, n_i) = 1$.
- Hence, a linear congruence $m_i x \equiv_{n_i} 1$ has a solution d_i .
- Define $x_0 = c_1 m_1 d_1 + c_2 m_2 d_2 + \dots + c_k m_k d_k$.
- Easy to see that $x_0 \equiv_{n_i} c_i m_i d_i \equiv_{n_i} c_i$. Thus, x_0 satisfies (1). □

Chinese remainder theorem (CRT)

Theorem

Assume $n_1, \dots, n_k \in \mathbb{Z}$ are pairwise coprime. Then for any $c_1, \dots, c_k \in \mathbb{Z}$ solutions to the system

$$\begin{cases} x \equiv_{n_1} c_1, \\ \dots \\ x \equiv_{n_k} c_k, \end{cases} \quad (1)$$

form a single congruence class modulo $n = n_1 \dots n_k$.

Proof.

Uniqueness of a solution modulo n :

- Assume $x \in \mathbb{Z}$ is another solution to the system (1).
- Hence, $x - x_0$ is divisible by all n_i 's.
- Since n_i 's are coprime it follows that $x - x_0$ is divisible by their product n .
- Hence, $x \equiv x_0 \pmod{n}$ and $x \in [x_0]$. □

CRT: example

The proof (existence part) of CRT gives a formula for a system of congruences.

Example. Solve
$$\begin{cases} x \equiv 1 \pmod{5} \\ x \equiv 3 \pmod{6} \end{cases}$$

$$n = 5 \cdot 6 = 30$$

$$n_1 = 5, \quad c_1 = 1, \quad m_1 = \frac{n}{n_1} = 6, \quad 6d_1 \equiv_5 1, \quad d_1 = 1$$

$$n_2 = 6, \quad c_2 = 3, \quad m_2 = \frac{n}{n_2} = 5, \quad 5d_2 \equiv_6 1, \quad d_2 = 5$$

Hence, a particular solution can be constructed as $x_0 = 1 \cdot 6 \cdot 1 + 3 \cdot 5 \cdot 5 = 81 \equiv_{30} 21$.

Example. Solve
$$\begin{cases} x \equiv_7 3, \\ x \equiv_8 2, \\ x \equiv_9 1. \end{cases}$$

$$n = 7 \cdot 8 \cdot 9 = 504$$

$$n_1 = 7, \quad c_1 = 3, \quad m_1 = \frac{n}{n_1} = 72, \quad 72d_1 \equiv_7 1, \quad d_1 = 4$$

$$n_2 = 8, \quad c_2 = 2, \quad m_2 = \frac{n}{n_2} = 63, \quad 63d_2 \equiv_8 1, \quad d_2 = -1$$

$$n_3 = 9, \quad c_3 = 1, \quad m_3 = \frac{n}{n_3} = 56, \quad 56d_3 \equiv_9 1, \quad d_3 = 5$$

Hence, $x_0 = 3 \cdot 72 \cdot 4 + 2 \cdot 63 \cdot (-1) + 1 \cdot 56 \cdot 5 = 1018 \equiv_{504} 10$.

CRT: example

We can use a slightly different method to solve a system of congruences:

$$\begin{cases} x \equiv 1 \pmod{2} \\ x \equiv 2 \pmod{3} \\ x \equiv 1 \pmod{5} \\ x \equiv 3 \pmod{7} \end{cases}$$

- Solve a subsystem

$$\begin{cases} x \equiv 1 \pmod{2} \\ x \equiv 2 \pmod{3} \end{cases} \Leftrightarrow x \equiv 5 \pmod{6}.$$

Replace the first two congruences with $x \equiv 5 \pmod{6}$:

$$\begin{cases} x \equiv 5 \pmod{6} \\ x \equiv 1 \pmod{5} \\ x \equiv 3 \pmod{7} \end{cases}$$

- Solve a subsystem

$$\begin{cases} x \equiv 5 \pmod{6} \\ x \equiv 1 \pmod{5} \end{cases} \Leftrightarrow x \equiv 11 \pmod{30}.$$

Replace the first two congruences with $x \equiv 11 \pmod{30}$:

$$\begin{cases} x \equiv 11 \pmod{30} \\ x \equiv 3 \pmod{7} \end{cases}$$

- Finally, we get $x \equiv 101 \pmod{210}$.

Modular exponentiation: computing $a^p \% n$

(Modular exponentiation) Given $a, p, n \in \mathbb{N}$, compute $a^p \% n$.

Straightforward approach: compute a^p , and then divide by n .

Straightforward approach is very inefficient for large numbers. It can be used for small numbers though. Fermat's little theorem can help.

Example. Compute $7^{1000} \% 22$.

By FLT, $7^{10} \equiv_{22} 1$. Hence, $7^{1000} = (7^{10})^{100} \equiv_{22} 1$.

Example. Compute $5^{1000} \% 13$.

By FLT, $5^{12} \equiv 1$. But, it is easy to note that $5^2 \equiv -1$. So,

$$5^{1000} = (5^2)^{500} \equiv_{13} (-1)^{500} = 1.$$

Example. Compute $2^{1000} \% 102$.

FLT not applicable. This problem can be approached by finding a small number x satisfying

$$x \equiv_{102} 2^{1000} \Leftrightarrow \begin{cases} x \equiv_2 2^{1000} \\ x \equiv_3 2^{1000} \\ x \equiv_{17} 2^{1000} \end{cases} \Leftrightarrow \begin{cases} x \equiv_2 0 \\ x \equiv_3 1 \\ x \equiv_{17} 1 \end{cases} \xLeftrightarrow^{CRT} x \equiv_{102} 52.$$

Binary method for exponentiation

To compute $a^p \% n$

- Write p in binary, i.e., as a sum of distinct powers of 2, $p = 2^{n_1} + \dots + 2^{n_s}$.
- Compute sufficiently many powers $a^{2^i} \% n$ by squaring
 - $a^{2^1} = a^2 \% n$,
 - $a^{2^2} = (a^{2^1})^2 \% n$ (square the previous number),
 - $a^{2^3} = (a^{2^2})^2 \% n$ (square the previous number),
 - $a^{2^4} = (a^{2^3})^2 \% n$ (square the previous number),
 - etc.
- Compute $a^p \% n$ as $a^{2^{n_1}} + \dots + a^{2^{n_s}} \% n$.

For instance, using binary exponentiation, we would compute certain powers of 3 as follows:

- $3^{63} = 3^{32+16+8+4+2+1} = 3^{32} \cdot 3^{16} \cdot 3^8 \cdot 3^4 \cdot 3^2 \cdot 3^1$
- $3^{53} = 3^{32+16+4+1} = 3^{32} \cdot 3^{16} \cdot 3^4 \cdot 3^1$
- $3^{39} = 3^{32+4+2+1} = 3^{32} \cdot 3^4 \cdot 3^2 \cdot 3^1$

We will see examples later.

Application of number theory: public key cryptosystems

Setting.

Two parties: Alice and Bob.

Bob wants to send Alice a secret message (number) m via a “public channel”, which means that everyone (e.g., his roommate, his internet provider, NSA) knows everything he sends. Bob wants only Alice to read m .

Third party. An **adversary** Eve wants to read the message m too.

- If there is no adversary, then there is no need to use encryption.
- Eve is an **active** adversary if she actively adds/modifies/reuses/erases messages between Alice and Bob.
- Eve is a **passive** adversary if she simply collects and analyses transactions between Alice and Bob and their public information.

Here we assume that Eve is passive.

RSA (Rivest–Shamir–Adleman, 1977)

Key generation (performed by Alice):

- Choose randomly two distinct prime numbers p and q .
- Compute $n = pq$ called Alice's **public modulus**.
- Compute $\varphi(n) = \varphi(p)\varphi(q)$.
- Choose a random integer $1 < e < \varphi(n)$ satisfying

$$\gcd(e, \varphi(n)) = 1.$$

The number e is called Alice's **public exponent**.

- Compute a number d satisfying $ed \equiv 1 \pmod{\varphi(n)}$, which implies that

$$ed = q \cdot \varphi(n) + 1 \text{ for some } q \in \mathbb{Z}.$$

The number d is called Alice's **private exponent**.

Finally, Alice publishes the numbers n and e together called **Alice's public key**.

Encryption (performed by Bob):

Bob's message is a number m satisfying $0 < m < n$ and $\gcd(m, n) = 1$.

- Compute $c = m^e \% n$, i.e., the remainder of division of m^e by n .

The number c is then sent to Alice.

Decryption (performed by Alice):

- Alice computes $c^d \% n$. The obtained number is m .

RSA (Rivest–Shamir–Adleman, 1977)

Key generation (performed by Alice):

- Choose randomly two distinct prime numbers p and q .
- Compute $n = pq$ called Alice's **public modulus**.
- Compute $\varphi(n) = \varphi(p)\varphi(q)$.
- Choose a random integer $1 < e < \varphi(n)$ satisfying

$$\gcd(e, \varphi(n)) = 1.$$

The number e is called Alice's **public exponent**.

- Compute a number d satisfying $ed \equiv 1 \pmod{\varphi(n)}$, which implies that

$$ed = q \cdot \varphi(n) + 1 \text{ for some } q \in \mathbb{Z}.$$

The number d is called Alice's **private exponent**.

Finally, Alice publishes the numbers n and e together called **Alice's public key**.

Encryption (performed by Bob):

Bob's message is a number m satisfying $0 < m < n$ and $\gcd(m, n) = 1$.

- Compute $c = m^e \% n$, i.e., the remainder of division of m^e by n .

The number c is then sent to Alice.

Decryption (performed by Alice):

- Alice computes $c^d \% n$. The obtained number is m .

RSA (Rivest–Shamir–Adleman, 1977)

Key generation (performed by Alice):

- Choose randomly two distinct prime numbers p and q .
- Compute $n = pq$ called Alice's **public modulus**.
- Compute $\varphi(n) = \varphi(p)\varphi(q)$.
- Choose a random integer $1 < e < \varphi(n)$ satisfying

$$\gcd(e, \varphi(n)) = 1.$$

The number e is called Alice's **public exponent**.

- Compute a number d satisfying $ed \equiv 1 \pmod{\varphi(n)}$, which implies that

$$ed = q \cdot \varphi(n) + 1 \text{ for some } q \in \mathbb{Z}.$$

The number d is called Alice's **private exponent**.

Finally, Alice publishes the numbers n and e together called **Alice's public key**.

Encryption (performed by Bob):

Bob's message is a number m satisfying $0 < m < n$ and $\gcd(m, n) = 1$.

- Compute $c = m^e \% n$, i.e., the remainder of division of m^e by n .

The number c is then sent to Alice.

Decryption (performed by Alice):

- Alice computes $c^d \% n$. The obtained number is m .

RSA: example

Key generation

- Choose random primes $p = 7$, $q = 13$.
- Then $n = 91$ and $\varphi(n) = 72$.
- We can choose $e = 5$ because $\gcd(72, 5) = 1$.
- Solving $5d \equiv_{72} 1$ we get $d = 29$.

Encryption: To encrypt $m = 10$ Bob compute $c = 10^5 \% 91$ which is 82.

Decryption: To decrypt the message Alice computes $82^{29} \% 91 = 10$:

- $82 \equiv_{91} -9$
- $82^2 \equiv_{91} 81 \equiv -10$
- $82^4 \equiv_{91} 100 \equiv 9$
- $82^8 \equiv_{91} 81 \equiv -10$
- $82^{16} \equiv_{91} 9$
- Since $29 = 16 + 8 + 4 + 1$ we have
 $82^{29} = 82^{16} 82^8 82^4 82 \equiv 9 \cdot (-10) \cdot 9 \cdot (-9) \equiv 90 \cdot 81 \equiv -1 \cdot (-10) \equiv 10$.

RSA: informal definition of security

RSA allows Bob to send private (encrypted) messages to Alice using her public key.

*We say that RSA is not secure in a presence of a passive adversary **Eve**, if Eve, given Alice's public key and the cipher c , can compute (in reasonable time) the message m .*

This definition of security is rather informal, but it gives a basic idea of computationally-secure encryption. Security is based on hardness of the following computational problem.

RSA problem: Given n, e, c such that

- $n = pq$ is a product of primes p and q ,
- $\gcd(e, \varphi(n)) = 1$, and
- $c = m^e \% n$ for some unknown m ,

compute m .

- RSA problem has no efficient solution for a conventional computer.
- RSA problem has an efficient solution for a quantum computer.

See: NIST Post-Quantum Cryptography

RSA: basic security analysis

Alice can compute m because she knows d (she generated it). Hence, d must be private. Furthermore:

- d is a solution of the congruence $de \equiv_{\varphi(n)} 1$. If $\varphi(n)$ is known to an adversary, then (since e is public) he can compute d . Thus, $\varphi(n)$ **must be private**!
- $\varphi(n) = \varphi(p)\varphi(q)$. If factorization of n (namely p and q) is known to the adversary, then he can compute $\varphi(n)$ and then compute d as explained above.

Thus, the numbers $p, q, \varphi(n), d$ must be private to Alice. Knowledge of any of those numbers gives the adversary access to information m . For RSA to be secure the adversary should not be able to compute those numbers (in a reasonable amount of time). In particular, the following computational problems must be hard.

The integer factorization problem: find $\text{PPF}(n)$ for a given number n .

Computing the values of φ : find $\varphi(n)$ for a given number n .

Both problems

- do not have efficient solutions on a conventional computer;
- have efficient solutions on a quantum computer.

A branch of math/cs that studies cryptographic primitives resistant to quantum attacks is called **post-quantum cryptography**.