2. Units. Euler function. CRT. RSA.

A. Ushakov

MA503, January 26, 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 if and only if $\gcd(a, n) \mid c$.
Furthermore, a particular solution can be found using the Euclidean algorithm.

(Find a solution or prove that a solution does not exist)

- $2x \equiv_5 3$.
- $6x \equiv_{14} 5$.
- $11x \equiv_{19} 8$.

Units modulo n

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

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

- $[a] \in U_n \Leftrightarrow \gcd(a, n) = 1$.
- *In particular,* $U_n = \{a \mid 0 \leq a \leq n-1, \quad \gcd(a, n) = 1\}$.

$$U_1 = \{0\}, \quad U_5 = \{1, 2, 3, 4\} \quad U_9 = \{1, 2, 4, 5, 7, 8\},$$
$$U_2 = \{1\}, \quad U_6 = \{1, 6\}, \quad U_{10} = \{1, 3, 7, 9\},$$
$$U_3 = \{1, 2\} \quad U_7 = \{1, 2, 3, 4, 5, 6\}, \quad U_{11} = \{1, \ldots, 10\},$$
$$U_4 = \{1, 3\} \quad U_8 = \{1, 3, 5, 7\}, \quad U_{12} = \{1, 5, 7, 11\}.$$

Finding multiplicative inverse modulo n

A multiplicative inverse of a modulo n is a number x satisfying

$$ax \equiv_n 1.$$

To find x we rewrite the congruence in an equivalent form

$$1 = \mathbf{a}x + q\mathbf{n},$$

with unknown x and q, and find any solution (x, q). The obtained value of x is a solution to the original congruence.

(Find multiplicative inverse)

- 2^{-1} *modulo* 3.
- 6^{-1} *modulo* 17.
- 11^{-1} *modulo* 27.

Euler function φ

Definition

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

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

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

For instance, $\mathrm{PPF}(360) = 2^3 \cdot 3^2 \cdot 5^1$ 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** $\varphi : \mathbb{N} \to \mathbb{N}$ is defined by $\varphi(n) = |U_n|$.

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

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

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

1. *Product of two units modulo n is a unit.*
2. *Multiplicative inverse of a unit modulo n is a unit.*

1. If a, b are units modulo n, then for some $x, y \in \mathbb{Z}$

$$ax \equiv_n 1 \quad \text{and} \quad by \equiv_n 1$$

and, hence, $(ab)(xy) \equiv_n 1$ and ab is a unit modulo n.

2. If x is a multiplicative inverse of a, then $ax \equiv_n 1$ and, hence, 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 $U_n = \{[x_1], \ldots, [x_k]\}$, where $k = \varphi(n)$.
- Consider the function $\varphi_a : U_n \to U_n$ defined by $[x_i] \overset{\varphi_a}{\mapsto} [ax_i]$.
- φ_a is injective because

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

- 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 $\{[x_1], \ldots, [x_k]\} = \{[ax_1], \ldots, [ax_k]\}$.
- Hence, $\prod_{i=1}^{k}[x_i] = \prod_{i=1}^{k}[ax_i]$ which implies that

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

Chinese remainder theorem

Theorem

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

$$\begin{cases} x \equiv_{n_1} c_1, \\ \ldots \\ x \equiv_{n_k} c_k, \end{cases} \tag{1}$$

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

Proof.

Existence of a solution:

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

Theorem

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

$$\begin{cases} x \equiv_{n_1} c_1, \\ \ldots \\ x \equiv_{n_k} c_k, \end{cases} \tag{1}$$

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

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, $x \equiv x_0 \mod n$ and $x \in [x_0]$. □

Examples

The proof (existence part) of CRT gives a solution to a system of congruence equations. For instance, consider:

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

- $n = 5 \cdot 6 = 30$
- $m_1 = \frac{n}{n_1} = 6$, $m_2 = \frac{n}{n_2} = 5$
- Solve $6x \equiv 1 \mod 5$ to get $d_1 \equiv_5 1$
- Solve $5x \equiv 1 \mod 6$ to get $d_2 \equiv_5 5$.
- Hence, the solution can be constructed as:

$$x_0 = 1 \cdot 6 \cdot 1 + 3 \cdot 5 \cdot 5 = 81$$

which, indeed, satisfies both congruences.

Examples

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

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

- Solve a subsystem

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

  Replace the first two congruences with $x \equiv 5 \mod 6$:

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

- Solve a subsystem

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

  Replace the first two congruences with $x \equiv 11 \mod 30$:

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

- Finally, we get $x \equiv 101 \mod 210$.

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

Straightforward approach to compute $a^p \,\%\, n$:

- *compute a^p*
- *find the remainder of division of a^p by n.*

E.g., $5^3\%13$ can be computed as $5^3 = 125 \equiv_{13} -5 \equiv_{13} 8$.

This is a bad approach as exponents grow very fast (try to compute 5^{1000}).

We can compute a sequence of powers mod n multiplying each time by a.

E.g. to compute $2^{10}\%11$ we could compute:

$$2^1 \equiv_{11} 2 \quad 2^2 \equiv_{11} 4 \quad 2^3 \equiv_{11} 8 \quad 2^4 \equiv_{11} 5 \quad 2^5 \equiv_{11} 10$$
$$2^6 \equiv_{11} 9 \quad 2^7 \equiv_{11} 7 \quad 2^8 \equiv_{11} 3 \quad 2^9 \equiv_{11} 6 \quad 2^{10} \equiv_{11} 1$$

The result 1 is not surprising as $\varphi(11) = 10$.

Using Fermat's little theorem for exponentiation

If the modulus n is small, then we can use tricks to simplify computations. E.g., we can notice that $5^2 \equiv_{13} -1$ and hence $5^{1000} = (5^2)^{500} \equiv_{13} (-1)^{500} = 1$.

Compute the following using Fermat's little theorem:

- 2^{1000} % 19
- 7^{1000} % 22
- 2^{1000} % 102

Binary method for exponentiation: example

Consider the following manipulations:

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

and notice that

- $3^2 = 3^2$,
- $3^4 = (3^2)^2$ **(square the previous number)**,
- $3^8 = (3^4)^2$ **(square the previous number)**,
- $3^{16} = (3^8)^2$ **(square the previous number)**,
- $3^{32} = (3^{16})^2$ **(square the previous number)**.

Compute values of the form 3^{2^n}. Finally, take an appropriate product. E.g.:

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

In general, to compute 3^k:

1. represent k as a sum of distinct powers of 2, $k = 2^{n_1} + \ldots + 2^{n_s}$;
2. $3^k = 3^{2^{n_1}+2^{n_2}+\ldots+2^{n_s}} = 3^{2^{n_1}} \cdot 3^{2^{n_2}} \cdot \ldots \cdot 3^{2^{n_s}}$.

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 $ed \equiv 1 \mod \varphi(n)$, 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 $ed \equiv 1 \mod \varphi(n)$, 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 $ed \equiv 1 \mod \varphi(n)$, 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 (Worked out 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 insecure in a presence of a passive adversary Eve)

if Eve can compute (in reasonable time) the message m given Alice's public key and the cipher c.

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* 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**.