Cryptography ECE5632 - Spring 2025

Lecture 7A

Dr. Farah Raad

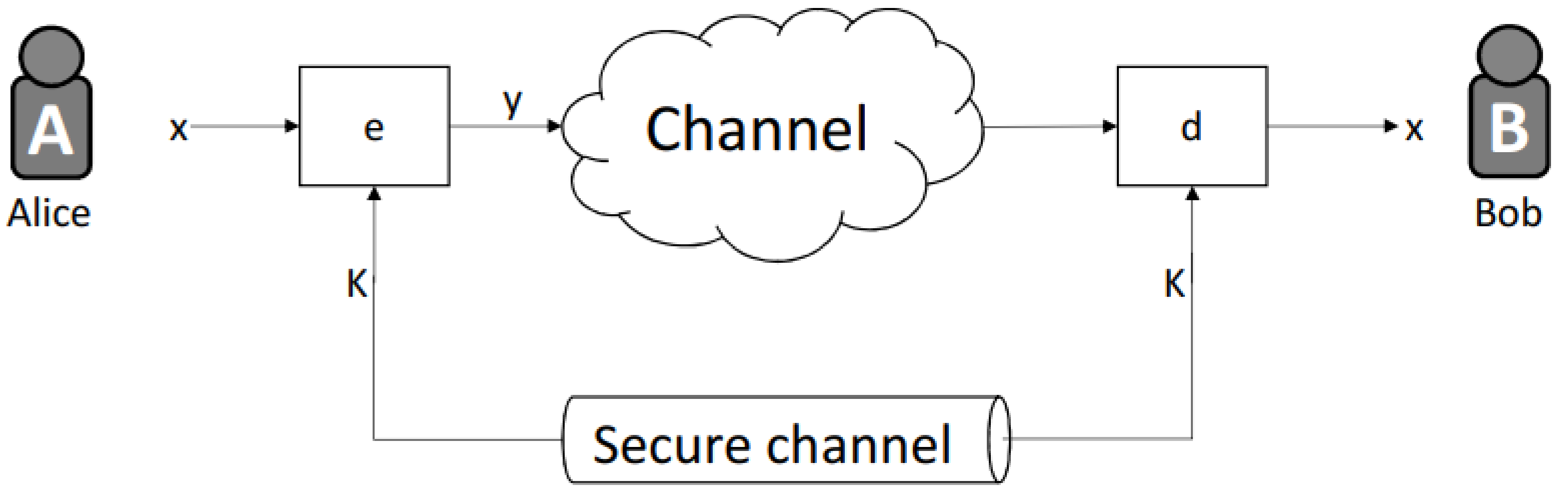
Lecture Topic

Introduction to Public-Key Cryptography (PKC)

Symmetric Cryptography Revisited

➢ Principle of Symmetric-Key encryption:

- The **same secret key** ***K*** is used for encryption and decryption
- Encryption and Decryption are very similar (or even identical) functions



X: plaintext
Y: ciphertext
K: key

❑ We have some problems; key distribution, number of keys, etc.

Symmetric Cryptography Revisited

➢ Symmetric Cryptography: Analogy

Safe with a strong lock, only Alice and Bob have a copy of the key

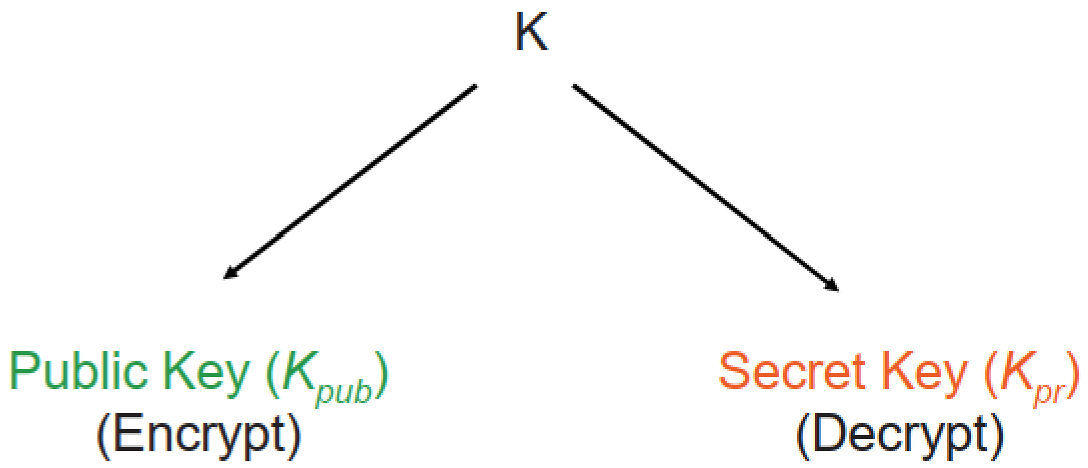
- Alice encrypts : locks message in the safe with her key
- Bob decrypts : uses his copy of the key to open the safe

Principles of Asymmetric Cryptography

➢ **Idea behind Asymmetric Cryptography**

Asymmetric (Public-Key) Cryptography

- ➢ Principle: “Split up” the key



- ✓ During the key generation, a key pair K_{pub} and K_{pr} is computed

Asymmetric (Public-Key) Cryptography

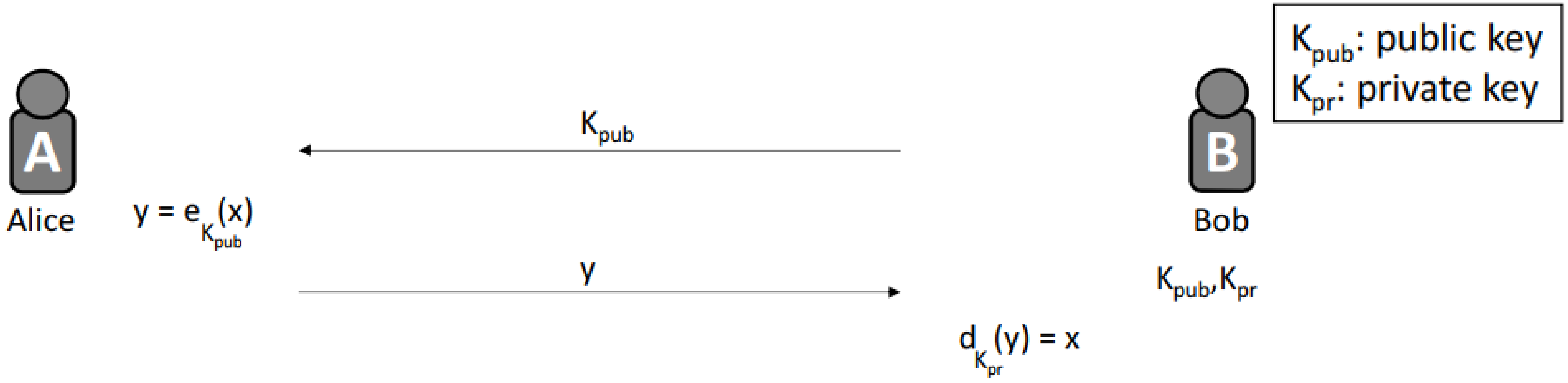
➢ Asymmetric Cryptography: Analogy

Safe with public lock and private lock:

- Alice deposits (encrypts) a message with the - *not secret* - public key K_{pub}
- Only Bob has the - *secret* - private key K_{pr} to retrieve (decrypt) the message

Asymmetric (Public-Key) Cryptography

➢ Basic protocol for Public-Key encryption:



✓ To study e() and d() .. **more Math is needed…**

Essential Number Theory for PKC

Euclidean Algorithm (EA)

➢ Compute the **greatest common divisor** ***gcd (*r_0*,* r_1*)*** of two integers r_0 and r_1

❖ gcd is **easy for small numbers**:

1. factor r_0 and r_1
2. gcd = highest common factor

Example 1 : $r_0 = 84, r_1 = 30$

$$r_0 = 84 = 2 \cdot 2 \cdot 3 \cdot 7$$
$$r_1 = 30 = 2 \cdot 3 \cdot 5$$

→ The gcd is the product of all common prime factors:

$2 \cdot 3 = 6 = gcd\ (30,84)$

❖ **But:** Factoring is complicated (and often infeasible) for large numbers

Euclidean Algorithm (EA)

Such method doesn't work with large numbers, i.e., the case of PKC. **We need the EA.**

e.g., 3^{2000}

Efficient
(faster, less complex)

Basic idea: $\gcd(r_0, r_1) = \gcd(r_0 \bmod r_1, r_1)$

$= \gcd(r_1, r_0 \bmod r_1)$

. . . we simply reduce the problem.

Example 1 : $r_0 = 84, r_1 = 30$

gcd(84, 30) = gcd(84 mod 30, 30) = gcd(**24**, 30)

= gcd(30 mod 24, 24) = gcd(**6**, 24) = 6

= gcd(24 mod 6, 6) = gcd(**0**, 6)

Terminate once a zero remainder is reached; gcd is the last remainder.

Same e.g., (illustrated)

$\overset{r_0}{84} = 2 \cdot \overset{r_1}{30} + \overset{r_2}{24}$

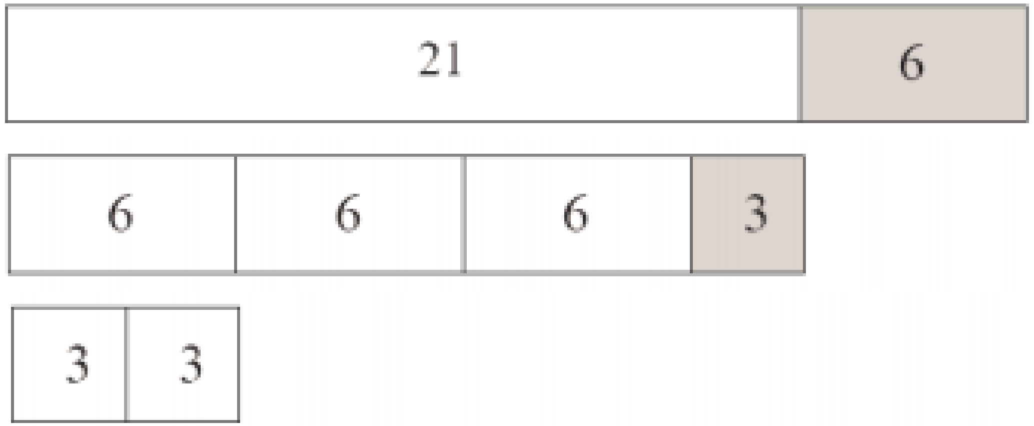
$30 = 1 \cdot 24 + \overset{r_3}{6}$ → gcd (84, 30) = 6

$24 = 4 \cdot 6 + 0$ . . . zero remainder reached.

Euclidean Algorithm (EA)

Example 2 : $\mathbf{r_0 = 27}$, $\mathbf{r_1 = 21}$

gcd (r_0, r_1) for r_0 = 27 and r_1 = 21



$\gcd(27, 21) = \gcd(1 \cdot 21 + 6, 21) = \gcd(21, 6)$

$\gcd(21, 6) = \gcd(3 \cdot 6 + 3, 6) = \gcd(6, 3)$

$\gcd(6, 3) = \gcd(2 \cdot 3 + 0, 3) = \gcd(3, 0) = 3$

- Note: very efficient method even for long numbers:
  The complexity grows **linearly** with the number of bits

Euclidean Algorithm (EA)

Example 3 : $\mathbf{r_0 = 973\ ,r_1 = 301}$

$\gcd(\overset{r_0}{973}, \overset{r_1}{301})$

$\overset{r_0}{973} = 3 \cdot \overset{r_1}{301} + \overset{r_2}{70}$

$301 = 4 \cdot 70 + \overset{r_3}{21}$

$70 = 3 \cdot 21 + \overset{r_4}{7}$ → $\gcd(973, 301) = 7$

$21 = 3 \cdot 7 + 0$ . . . zero remainder reached.

Pretty simple…

Extended Euclidean Algorithm (EEA)

Goal: rewrite $\gcd(r_0, r_1) = s \cdot r_0 + t \cdot r_1$

Why and **How**?

Why: To compute <u>modular inverses of large numbers</u>.

How: Using regular EA for the LHS & the extension

Extended Euclidean Algorithm (EEA)

- Extend the Euclidean algorithm to **find modular inverse** of r_1 mod r_0
- EEA computes *s*,*t*, and the gcd : $\gcd(r_0, r_1) = s \cdot r_0 + t \cdot r_1$
- Take the relation **mod r_0**

$$s \cdot r_0 + t \cdot r_1 = 1$$

$$s \cdot 0 + t \cdot r_1 \equiv 1 \bmod r_0$$

$$r_1 \cdot t \equiv 1 \bmod r_0$$

→ Compare with the definition of modular inverse**: *t* is the inverse of r_1 mod r_0**

- Note that *gcd* $(r_0, r_1) = 1$ in order for the inverse to exist

- **Recursive formulae** to calculate *s* and *t* in each step

  → „magic table" for *r*, *s*, *t* and a quotient *q* to derive the inverse with pen and paper

Extended Euclidean Algorithm

- How to compute the modular inverse using the Extended Euclidean Algorithm:

i	$q_i = \left\lfloor \frac{r_{i-2}}{r_{i-1}} \right\rfloor$	$s_i = s_{i-2} - q_{i-1} \cdot s_{i-1}$	$t_i = t_{i-2} - q_{i-1} \cdot t_{i-1}$	$r_i = r_{i-2} - q_{i-1} \cdot r_{i-1}$
0		$s_0 = 1$	$t_0 = 0$	r_0
1		$s_1 = 0$	$t_1 = 1$	r_1
2	$q_1 = \left\lfloor \frac{r_0}{r_1} \right\rfloor$	$s_2 = s_0 - q_1 \cdot s_1$	$t_2 = t_0 - q_1 \cdot t_1$	$r_2 = r_0 - q_1 \cdot r_1$
3	$q_2 = \left\lfloor \frac{r_1}{r_2} \right\rfloor$	$s_3 = s_1 - q_2 \cdot s_2$	$t_3 = t_1 - q_2 \cdot t_2$	$r_3 = r_1 - q_2 \cdot r_2$
⋮	⋮	⋮	⋮	⋮

For initialization (steps $i \in \{0,1\}$, cell values are predetermined as proven before.

For $i \geq 2$, compute the q_i, s_i, t_i, r_i columns.

For each iteration *i*, check:

If $r_i=1$ is reached, then **$\gcd(r_0,r_1) = r_i = 1$**. Then **mult. inverse of r_1 mod r_0 exists and equals t_i**. Stop.

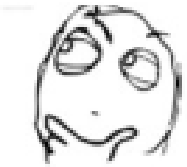
Else, if $r_i=0$ is reached, **then $\gcd(r_0,r_1) = r_{i-1}$**. Then **mult. inverse of r_1 mod r_0 doesn't exist**. Stop.

Extended Euclidean Algorithm (EEA)

$\gcd(r_0, r_1)$	$r_0 = q_1 \cdot r_1 + r_2$	$r_2 = s_2 \cdot r_0 + t_2 \cdot r_1$
$\gcd(r_1, r_2)$	$r_1 = q_2 \cdot r_2 + r_3$	$r_3 = s_3 \cdot r_0 + t_3 \cdot r_1$
	$\vdots$	$\vdots$
$\gcd(r_{l-2}, r_{l-1})$	$r_{l-2} = q_{l-1} \cdot r_{l-1} + r_l$	$r_l = \overset{s}{s_l} \cdot r_0 + \overset{t}{t_l} \cdot r_1 = \gcd(r_0, r_1)$
$\gcd(r_{l-1}, r_l)$	$r_{l-1} = q_l \cdot r_l + 0$	

r_l is the last remainder, i.e., the result of $\gcd(r_0, r_1)$.

How does that lead to modulo inverse?



To compute $a^{-1} \bmod n$:

$\gcd(n, a) = r_l = s \cdot n + t \cdot a = 1$ (condition for inverse existence)

Then $s \cdot 0 + t \cdot a \equiv 1 \bmod n$ (mod n for both sides)

$t \cdot a \equiv 1 \bmod n$

$t \equiv a^{-1} \bmod n$

Extended Euclidean Algorithm (EEA)

Example:

- Calculate the modular Inverse of 12 mod 67:
- From magic table follows $-5 \cdot 67 + 28 \cdot 12 = 1$
- Hence **28 is the inverse** of 12 mod 67.

i	q_{i-1}	r_i	s_i	t_i
2	5	7	1	-5
3	1	5	-1	6
4	1	2	2	-11
5	2	1	-5	**28**

- Check: $28 \cdot 12 = 336 \equiv 1 \bmod 67$ ✓

Multiplicative Inverse in GF(2^m)

Find the multiplicative inverse of x^3+x+1 in GF(2^4) with $P(x)=x^4+x+1$

Using EEA:

$x^4+x+1 = (x)(x^3+x+1) + (x^2+1)$ . . . $x^2+1 = (x^4+x+1) + (x)(x^3+x+1)$

$(x^3+x+1) = (x)(x^2+1) + 1$ . . . $1 = (x)(x^2+1) + (x^3+x+1)$

$= (x)(x^4+x+1) + (x^2+1)(x^3+x+1)$

$[(x)(x^4+x+1) + (x^2+1)(x^3+x+1)] \bmod P(x)$ → multiplicative inverse $= x^2+1$

Multiplicative Inverse in GF(2^m)

Example We are looking for the inverse of $A(x) = x^2$ in the finite field $GF(2^3)$ with $P(x) = x^3 + x + 1$. The initial values for the $t(x)$ polynomial are: $t_0(x) = 0$, $t_1(x) = 1$

Iteration	$r_{i-2}(x)$	$= [q_{i-1}(x)]\, r_{i-1}(x) + [r_i(x)]$	$t_i(x)$
2	$x^3 + x + 1$	$= [x]\, x^2 + [x + 1]$	$t_2 = t_0 - q_1 t_1 = 0 - x\,1 \equiv x$
3	x^2	$= [x]\,(x + 1) + [x]$	$t_3 = t_1 - q_2 t_2 = 1 - x\,(x) \equiv 1 + x^2$
4	$x + 1$	$= [1]\, x + [1]$	$t_4 = t_2 - q_3 t_3 = x - 1\,(1 + x^2)$
			$t_4 \equiv 1 + x + x^2$
5	x	$= [x]\, 1 + [0]$	Termination since $r_5 = 0$

$$A^{-1}(x) = t(x) = t_4(x) = x^2 + x + 1.$$

Multiplicative Inverse in GF(2^m)

Here is the check that $t(x)$ is in fact the inverse of x^2, where we use the properties that $x^3 \equiv x+1 \bmod P(x)$ and $x^4 \equiv x^2+x \bmod P(x)$:

$$\begin{aligned} t_4(x) \cdot x^2 &= x^4 + x^3 + x^2 \\ &\equiv (x^2+x) + (x+1) + x^2 \bmod P(x) \\ &\equiv 1 \bmod P(x) \end{aligned}$$

Euler's Phi Function

For PKC, it's important to know how many numbers in Z_m that are relatively prime to m.

Why and **how**?

Why: Will be clear later once we study actual PK cryptosystems.

How: Using Euler's Phi function simply counts these numbers.

Manually counting may work for small numbers.

e.g., manually count the numbers in Z_6 that are relatively prime to 6. → $\Phi(6) = 2$

For large numbers, we use Euler's Phi function.

Euler's Phi Function

- For PKC, it's important to know how many numbers in Z_m that are relatively prime to m.

Why and **how**?

Why: Will be clear later once we study actual PK cryptosystems.
How: Using Euler's Phi function simply counts these numbers.

- ***New problem,*** *important for public-key systems, e.g., RSA:*
  Given the set of the *m* integers {0, 1, 2, …, *m* -1},
- **How many** numbers in the set are **relatively prime to *m*** ?
  - **Answer:** **Euler's Phi function $\Phi(m)$**

Euler's Phi Function

- **Example** for the sets {0,1,2,3,4,5} (*m*=6), and {0,1,2,3,4} (*m*=5)

$$\gcd(0,6) = 6$$
$$\gcd(1,6) = 1 \leftarrow$$
$$\gcd(2,6) = 2$$
$$\gcd(3,6) = 3$$
$$\gcd(4,6) = 2$$
$$\gcd(5,6) = 1 \leftarrow$$

→ 1 and 5 relatively prime to *m*=6,

hence **$\Phi(6) = 2$**

$$\gcd(0,5) = 5$$
$$\gcd(1,5) = 1 \leftarrow$$
$$\gcd(2,5) = 1 \leftarrow$$
$$\gcd(3,5) = 1 \leftarrow$$
$$\gcd(4,5) = 1 \leftarrow$$

→ **$\Phi(5) = 4$**

- Testing one gcd per number in the set is **extremely slow for large *m*.**

Euler's Phi Function

- Manually counting may work for **small numbers**.

e.g., manually count the numbers in Z_6 that are relatively prime to 6. → $\Phi(6) = 2$

✓ For **large numbers**, we use Euler's Phi function.

Euler's Phi Function

- **If** canonical factorization of m known: $m = p_1^{e_1} \cdot p_2^{e_2} \cdot \ldots \cdot p_n^{e_n}$
  (where p_i primes and e_i positive integers)
- **then** calculate Phi according to the relation $\Phi(m) = \prod_{i=1}^{n} (p_i^{e_i} - p_i^{e_i - 1})$

- Phi especially easy for $e_i = 1$, e.g., $m = p \cdot q \rightarrow \Phi(m) = (p-1) \cdot (q-1)$

- **Example** $m = 899 = 29 \cdot 31$:
  Φ(899) = (29-1) · (31-1) = 28 · 30 **= 840**

- **Note:** Finding $\Phi(m)$ is computationally easy **if factorization of *m* is known**
  (otherwise the calculation of $\Phi(m)$ becomes computationally infeasible for large numbers)

Euler’s Phi Function

How to compute $\Phi(m)$ for a large m?

Let m have the following factorization form:

$$m = p_1^{e_1} \cdot p_2^{e_2} \cdot \ldots \cdot p_n^{e_n},$$

Where p_i are distinct prime numbers and e_i are positive integers, then

$$\Phi(m) = \prod_{i=1}^{n} (p_i^{e_i} - p_i^{e_i - 1}).$$

Euler's Phi Function

e.g. 1) compute $\Phi(m)$ for m = 240

m = 16·15 = $2^4 \cdot 3^1 \cdot 5^1$

= $p_1^{e_1}\, p_2^{e_2}\, p_3^{e_3}$

$$\Phi(240) = \prod_{i=1}^{3}(p_i^{e_i} - p_i^{e_i-1}) = (2^4-2^3)(3^1-3^0)(5^1-5^0)$$

$$= 8 \cdot 2 \cdot 4 = 64$$

e.g. 2) compute $\Phi(m)$ for m = 100

Euler’s Theorem

- Used in public-key cryptography.

Euler’s Theorem:

Let a and m be integers with gcd(a,m) = 1, then:

$$a^{\Phi(m)} \equiv 1 \text{ mod m}$$

- **e.g., Let's check with m = 12 and a = 5.**

$$\Phi(12) = \Phi(2^2 \cdot 3) = (2^2 - 2^1)(3^1 - 3^0) = (4-2)(3-1) = 4$$

$$5^{\Phi(12)} = 5^4 = 25^2 = 625 \equiv 1 \text{ mod } 12$$

Fermat's Little Theorem

➢ Fermat's Little Theorem:

- Given a **prime *p*** and an **integer *a***: $a^p \equiv a \pmod{p}$
- Can be rewritten as $a^{p-1} \equiv 1 \pmod{p}$
- **Use: Find modular inverse**, if p is prime. Rewrite to $a \cdot a^{p-2} \equiv 1 \pmod{p}$
- Comparing with definition of the modular inverse $a \cdot a^{-1} \equiv 1 \bmod m$

→ $a^{-1} \equiv a^{p-2} \pmod{p}$ is the modular inverse modulo a prime *p*

Fermat’s Little Theorem

➢ Fermat’s Little Theorem:

Let a be an integer and p be a prime,

then: $a^p \equiv a \bmod p$

so, $a^{p-1} \equiv 1 \bmod p$

or, $a \cdot a^{p-2} \equiv 1 \bmod p$

so, $a^{-1} \equiv a^{p-2} \bmod p$

✓ **e.g., Let’s check with p = 7 and a = 2**

$$a^{p-2} = 2^5 = 32 \equiv 4 \bmod 7$$

$$2 \cdot 4 \equiv 1 \bmod 7$$

Therefore, $2^{-1} \equiv 4 \bmod 7$

Fermat’s Little Theorem and Euler’s Theorem

- Fermat‘s little theorem = special case of Euler‘s Theorem
- for a prime ***p***: $\Phi(p) = (p^1 - p^0) = p - 1$

  → Fermat: $a^{\Phi(p)} = a^{p-1} \equiv 1 \pmod{p}$

Euler's Phi Function

Notes:

If p is prime, then $\Phi(p) = p - 1$

if p and q are prime, then $\Phi(pq) = \Phi(p) \times \Phi(q)$

How to prove that $\Phi(pq) = \Phi(p) \times \Phi(q)$?

Since Z_n has (pq-1) positive integers.

Since integers that are not relatively prime to n are {p,2p,…(q-1)p} and {q,2q,..,(p-1)q} …
i.e., (p-1) elements + (q-1) elements.

Then the number of integers in Z_n that are relatively prime to n = (pq-1) – [(p-1)+(q-1)]

i.e., pq – (p+q) +1

Then $\Phi(n)$=(p-1)x(q-1) = $\Phi(p) \times \Phi($q$)$

MSA UNIVERSITY
IN EGYPT SINCE 1996
Established by Dr.Nawal El Degwi
MSA UNIVERSITY
جامعة أكتوبر للعلوم الحديثة والآداب

Thank You!

See You next Lectures!!

Any Question?

THE FIRST BRITISH HIGHER EDUCATION IN EGYPT

26th July Mehwar Road Intersection with Wahat Road, 6th of October City, Egypt
Tel: +202383711146 **Fax:** +20238371543 **Postal code:** 12451
Email: info@msa.eun.eg **Hotline:** 16672 **Website:** www.msa.edu.eg