MATHEMATICS OF CRYPTOGRAPHY PART III

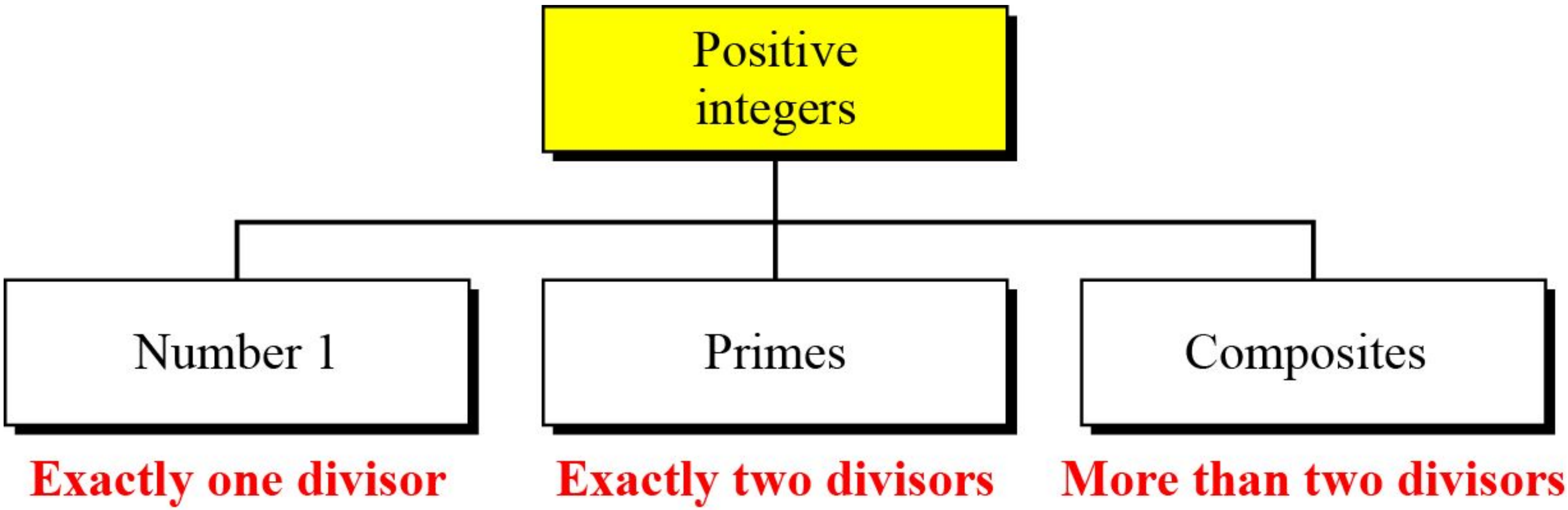
Primes and Related Congruence Equations

Objectives

- To introduce prime numbers and their applications in cryptography
- To discuss some primality test algorithms and their efficiencies.
- To discuss factorization algorithms and their applications in cryptography
- To discuss the Chinese remainder theorem and its application
- To introduce modular exponentiation and algorithm

Primes

Three groups of positive integers



A prime is divisible only by itself and 1.

The smallest prime????

Primes(cont.)

- Number of Primes

$$[n / (\ln n)] \quad < \quad \pi(n) \quad < \quad [n/(\ln n - 1.08366)]$$

- E.g. Find the number of primes less than 1,000,000.
  – The approximation gives the range 72,383 to 78,543. The actual number of primes is 78,498

Checking for Primeness

- Given a number n, how can we determine if n is a prime?
  - The answer is that we need to see if the number is divisible by primes less than $\sqrt{n}$
- Is 97 a prime?
  - The floor of $\sqrt{97} = 9$. The primes less than 9 are 2, 3, 5, and 7. We need to see if 97 is divisible by any of these numbers. It is not, so 97 is a prime.

Checking for Primeness(cont.)

- Is 301 a prime?
  - The floor of $\sqrt{301}$ = 17. We need to check 2, 3, 5, 7, 11, and 13. The numbers 2, 3, and 5 do not divide 301, but 7 does. Therefore 301 is not a prime.

Euler's Phi-Function

- *Euler's phi-function*, φ (n), which is sometimes called the *Euler's totient function* plays a very important role in cryptography.
- The function finds the number of integers that are both smaller than *n* and relatively prime to *n*
- The followings helps to find the value of φ (n).

1. $\phi(1) = 0$.
2. $\phi(p) = p - 1$ if p is a prime.
3. $\phi(m \times n) = \phi(m) \times \phi(n)$ if m and n are relatively prime.
4. $\phi(p^e) = p^e - p^{e-1}$ if p is a prime.

Euler's Phi-Function(cont.)

- We can combine all four rules to find the value of φ (n). For example, if n can be factored as

$$n = p_1^{e1} \times p_2^{e2} \times \ldots \times p_k^{ek}$$

- Then we combine the third and the fourth rule to find

$$\phi(n) = (p_1^{e_1} - p_1^{e_1 - 1}) \times (p_2^{e_2} - p_2^{e_2 - 1}) \times \cdots \times (p_k^{ek} - p_k^{ek - 1})$$

- The value of φ (n) for large composites can be found only if the number *n* can be factored into primes.

The difficulty of finding φ(n) depends on the difficulty of finding the factorization of n.
"n can be factored into primes"

Euler's Phi-Function(cont.)

1. $\phi(1) = 0.$
2. $\phi(p) = p - 1$ if p is a prime.
3. $\phi(m \times n) = \phi(m) \times \phi(n)$ if m and n are relatively prime.
4. $\phi(p^e) = p^e - p^{e-1}$ if p is a prime.

- Example 1
  - What is the value of $\varphi(13)$?
- Solution
  - Because 13 is a prime, $\varphi(13) = (13 - 1) = 12$.
- Example 2
  - What is the value of $\varphi(10)$?
- Solution
  - We can use the third rule: $\varphi(10) = \varphi(2) \times \varphi(5) = 1 \times 4 = 4$, because 2 and 5 are primes.

Euler's Phi-Function(cont.)

1. $\phi(1) = 0.$
2. $\phi(p) = p - 1$ if p is a prime.
3. $\phi(m \times n) = \phi(m) \times \phi(n)$ if m and n are relatively prime
4. $\phi(p^e) = p^e - p^{e-1}$ if p is a prime.

- Example 3
  - What is the value of $\varphi(240)$?
- Solution
  - We can write $240 = 2^4 \times 3^1 \times 5^1$. Then

    $\varphi(240) = (2^4 - 2^3) \times (3^1 - 3^0) \times (5^1 - 5^0) = 64$
- Example 4
  - Can we say that $\varphi(49) = \varphi(7) \times \varphi(7) = 6 \times 6 = 36$????
- Solution
  - No. The third rule applies when *m* and *n* are relatively prime. Here $49 = 7^2$. We need to use the fourth rule: $\varphi(49) = 7^2 - 7^1 = 42$.

Euler's Phi-Function(cont.)

- Example 5
  - What is the number of elements in $Z_{14}{}^*$?
- Solution
  - The answer is $\varphi(14) = \varphi(7) \times \varphi(2) = 6 \times 1 = 6$. The members are 1, 3, 5, 9, 11, and 13.

Interesting point: If $n > 2$, the value of $\varphi(n)$ is even.

Examples

If $n = 2020$, then $\Phi(n)$ is

A. 200

B. 400

C. 600

D. 800

prime factorization of 2,020 is $2^2 \times 5 \times 101$

Accepted Answers:

D.

Fermat's Little Theorem

- It plays important role in cryptography. It has two versions.
- First Version
  - If p is a prime and a is an integer such that p does not divide a, then

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

- Second Version
  - Removes the condition on a
  - It says that if p is prime and a is an integer,

$$a^{p} \equiv a \bmod p$$

<exponent and modulus are same>

Fermat's Little Theorem(cont.)

- Application- <exponentiation> it is helpful for quickly finding a solution to some exponentiation.
- Example 1
  - Find the result of 6^{10} mod 11.

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

$$a^{p} \equiv a \bmod p$$

- Solution
  - We have 6^{10} mod 11 = 1. This is the first version of Fermat's little theorem where p = 11.
- Example 2
  - Find the result of 3^{12} mod 11.
- Solution
  - Here the exponent (12) and the modulus (11) are not the same. With substitution this can be solved using Fermat's little theorem.

$$3^{12} \bmod 11 = (3^{11} \times 3) \bmod 11 = (3^{11} \bmod 11)\,(3 \bmod 11) = (3 \times 3) \bmod 11 = 9$$

Fermat's Little Theorem(cont.)

- Application- Multiplicative Inverses <if modulus is prime>
- If *p* is a prime and *a* is an integer such that *p* does not divide *a* .

$$a^{-1} \bmod p = a^{p-2} \bmod p$$

- The answers to multiplicative inverses modulo a prime can be found without using the extended Euclidean algorithm:

a. $8^{-1} \bmod 17 = 8^{17-2} \bmod 17 = 8^{15} \bmod 17 = 15 \bmod 17$

b. $5^{-1} \bmod 23 = 5^{23-2} \bmod 23 = 5^{21} \bmod 23 = 14 \bmod 23$

c. $60^{-1} \bmod 101 = 60^{101-2} \bmod 101 = 60^{99} \bmod 101 = 32 \bmod 101$

d. $22^{-1} \bmod 211 = 22^{211-2} \bmod 211 = 22^{209} \bmod 211 = 48 \bmod 211$

Euler's Theorem

- It is the generalization of Fermat's little theorem. The modulus in Euler's theorem is an integer not prime.
- First Version
  - If a and n are coprime,

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

- Second Version
  - Removes the condition that a and n should be coprime. If $n=p\times q$, $a<n$, and k is integer, then

$$a^{k \times \varphi(n) + 1} \equiv a \pmod{n}$$

The second version of Euler's theorem is used in the RSA cryptosystem

Euler's Theorem(cont.)

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

$$a^{k \times \varphi(n) + 1} \equiv a \pmod{n}$$

- Application- Exponentiation
- Example 1
  - Find the result of 6^{24} mod 35.
- Solution
  - We have 6^{24} mod 35 = $6^{\varphi(35)}$ mod 35 = 1.
- Example 2
  - Find the result of 20^{62} mod 77.
- Solution

If we let k = 1 on the second version, we have

20^{62} mod 77 = (20 mod 77) ($20^{\varphi(77) + 1}$ mod 77) mod 77

= (20)(20) mod 77 = 15.

Euler’s Theorem(cont.)

- Application-Multiplicative Inverses
  - Euler’s theorem can be used to find multiplicative inverses modulo a composite. If *n* and *a* are coprime, then

$$a^{-1} \bmod n = a^{\varphi(n)-1} \bmod n$$

Euler's Theorem(cont.)

- Example

$$a^{-1} \bmod n = a^{\varphi(n)-1} \bmod n$$

  - The answers to multiplicative inverses modulo a composite can be found without using the extended Euclidean algorithm.

a. $8^{-1} \bmod 77 = 8^{\phi(77)-1} \bmod 77 = 8^{59} \bmod 77 = 29 \bmod 77$

b. $7^{-1} \bmod 15 = 7^{\phi(15)-1} \bmod 15 = 7^{7} \bmod 15 = 13 \bmod 15$

c. $60^{-1} \bmod 187 = 60^{\phi(187)-1} \bmod 187 = 60^{159} \bmod 187 = 53 \bmod 187$

d. $71^{-1} \bmod 100 = 71^{\phi(100)-1} \bmod 100 = 71^{39} \bmod 100 = 31 \bmod 100$

Generating Primes

- Mersenne Primes- $\boxed{M_p = 2^p - 1}$
- If p in the above formula is a prime, then M_p was thought to be prime.

$$
\begin{array}{ll}
M_2 \ = 2^2 - 1 = 3 & \\
M_3 \ = 2^3 - 1 = 7 & \\
M_5 \ = 2^5 - 1 = 31 & \\
M_7 \ = 2^7 - 1 = 127 & \\
M_{11} = 2^{11} - 1 = 2047 & \textbf{Not a prime } (2047 = 23 \times 89) \\
M_{13} = 2^{13} - 1 = 8191 & \\
M_{17} = 2^{17} - 1 = 131071 &
\end{array}
$$

A number in the form $M_p = 2^p - 1$ is called a Mersenne number and may or may not be a prime.

Generating Primes(cont.)

- Fermat Primes $\boxed{F_n = 2^{2^n} + 1}$

$F_0 = 3 \quad F_1 = 5 \quad F_2 = 17 \quad F_3 = 257 \quad F_4 = 65537$

$F_5 = 4294967297 = 641 \times 6700417$ ***Not a prime***

Primality Testing

- Finding an algorithm to correctly and efficiently test a very large integer and output *a prime or a composite* has always been a challenge in number theory.
- Two types
  - Deterministic Algorithms <gives correct answer>
  - Probabilistic Algorithms <gives an answer that is correct most of the time, but not all of time>

Deterministic Algorithms

- Divisibility Algorithm

Algorithm 9.1 *Pseudocode for the divisibility test*

```
Divisibility_Test (n)                    // n is the number to test for primality
{
  r ← 2
  while (r < √n )
  {
   if (r | n) return "a composite"
   r ← r + 1
  }
  return "a prime"
}
```

Probabilistic Algorithms

- Fermat Test **If n is a prime, then $a^{n-1} \equiv 1 \bmod n$.**

If n is a prime, $a^{n-1} \equiv 1 \bmod n$
If n is a composite, it is possible that $a^{n-1} \equiv 1 \bmod n$

- Example
  - Does the number 561 pass the Fermat test?

Probabilistic Algorithms(cont.)

- Example
  - Does the number 561 pass the Fermat test?
- Solution
  - Use base 2

$$2^{561-1} = 1 \bmod 561$$

  - The number passes the Fermat test, but it is not a prime, because 561 = 33 × 17.

FACTORIZATION

Fundamental Theorem of Arithmetic

$$n = p_1^{e1} \times p_2^{e2} \times \cdots \times p_k^{ek}$$

- Greatest Common Divisor

$$a = p_1^{a_1} \times p_2^{a_2} \times \cdots \times p_k^{a_k} \qquad b = p_1^{b_1} \times p_2^{b_2} \times \cdots \times p_k^{bk}$$

$$\gcd(a, b) = p_1^{\min(a_1, b_1)} \times p_2^{\min(a_2, b_2)} \times \cdots \times p_k^{\min(a_k, b_k)}$$

- Least Common Multiplier- smallest integer that is multiple of both a&b

$$a = p_1^{a1} \times p_2^{a2} \times \cdots \times p_k^{ak} \qquad b = p_1^{b1} \times p_2^{b2} \times \cdots \times p_k^{bk}$$

$$\text{lcm}(a, b) = p_1^{\max(a_1, b_1)} \times p_2^{\max(a_2, b_2)} \times \cdots \times p_k^{\max(a_k, b_k)}$$

- Example- GCD & LCM of 16 and 64

$$\text{lcm}(a, b) \times \gcd(a, b) = a \times b$$

CHINESE REMAINDER THEOREM

- Used to solve a set of congruent equations with one variable but different moduli, which are relatively prime

$$
\begin{aligned}
x &\equiv a_1 \pmod{m_1} \\
x &\equiv a_2 \pmod{m_2} \\
&\cdots \\
x &\equiv a_k \pmod{m_k}
\end{aligned}
$$

- The above equations have a unique solution if the moduli are relatively prime

Continued…

- Example
  - The following is an example of a set of equations with different moduli:

$$
\begin{aligned}
x &\equiv 2 \pmod{3} \\
x &\equiv 3 \pmod{5} \\
x &\equiv 2 \pmod{7}
\end{aligned}
$$

  - The solution to this set of equations is given in the next section; for the moment, note that the answer to this set of equations is $x = 23$. This value satisfies all equations: $23 \equiv 2 \pmod{3}$, $23 \equiv 3 \pmod{5}$, and $23 \equiv 2 \pmod{7}$.

Continued…

- Solution To Chinese Remainder Theorem
  - Find $M = m_1 \times m_2 \times \dots \times m_k$. This is the common modulus.
  - Find $M_1 = M/m_1$, $M_2 = M/m_2$, …, $M_k = M/m_k$.
  - Find the multiplicative inverse of M_1, M_2, …, M_k using the corresponding moduli (m_1, m_2, …, m_k). Call the inverses M_1^{-1}, M_2^{-1}, …, M_k^{-1}.
  - The solution to the simultaneous equations is

$$\boxed{x = (a_1 \times M_1 \times M_1^{-1} + a_2 \times M_2 \times M_2^{-1} + \cdots + a_k \times M_k \times M_k^{-1}) \bmod M}$$

Continued...

- Example
  - Find the solution to the simultaneous equations:

$$
\begin{aligned}
x &\equiv 2 \pmod{3} \\
x &\equiv 3 \pmod{5} \\
x &\equiv 2 \pmod{7}
\end{aligned}
$$

- Solution: We follow the four steps.

1. $M = 3 \times 5 \times 7 = 105$

2. $M_1 = 105 / 3 = 35$, $M_2 = 105 / 5 = 21$, $M_3 = 105 / 7 = 15$

3. The inverses are $M_1^{-1} = 2$, $M_2^{-1} = 1$, $M_3^{-1} = 1$

4. $x = (2 \times 35 \times 2 + 3 \times 21 \times 1 + 2 \times 15 \times 1) \bmod 105 = 23 \bmod 105$

Continued…

- Example
  - Find an integer that has a remainder of 3 when divided by 7 and 13, but is divisible by 12.
- Solution ????

Continued…

- Example
  - Find an integer that has a remainder of 3 when divided by 7 and 13, but is divisible by 12.
- Solution
  - This is a CRT problem. We can form three equations and solve them to find the value of x.

$$x \equiv 3 \bmod 7$$
$$x \equiv 3 \bmod 13$$
$$x \equiv 0 \bmod 12$$

  - If we follow the four steps, we find x = 276. We can check that
    $276 \equiv 3 \bmod 7$, $276 \equiv 3 \bmod 13$ and 276 is divisible by 12 (the quotient is 23 and the remainder is zero).

Continued…

- Assume we need to calculate z = x + y where x = 123 and y = 334. These numbers can be represented as follows:

$$
\begin{array}{ll}
x \equiv 24 \pmod{99} & y \equiv 37 \pmod{99} \\
x \equiv 25 \pmod{98} & y \equiv 40 \pmod{98} \\
x \equiv 26 \pmod{97} & y \equiv 43 \pmod{97}
\end{array}
$$

- Adding each congruence in x with the corresponding congruence in y gives

$$
\begin{array}{lll}
x + y \equiv 61 \pmod{99} & \rightarrow & z \equiv 61 \pmod{99} \\
x + y \equiv 65 \pmod{98} & \rightarrow & z \equiv 65 \pmod{98} \\
x + y \equiv 69 \pmod{97} & \rightarrow & z \equiv 69 \pmod{97}
\end{array}
$$

- Now three equations can be solved using the Chinese remainder theorem to find z. One of the acceptable answers is z = 457.

Continued…

Secret Sharing scheme in cryptography **aims to distribute and later recover secret S among n parties**. Secret S is distributed in form of **shares** which are generated from secret. Without cooperation of k no. of parties, the secret cannot be reconstructed from shares directly. Consider the following example:

Say our secret is S. The shares for n=4 no. of parties are generated taking modulus 11,13,17 and 19. They are respectively 1,12,2 and 3 and given by following equations:

S Ξ 1 mod 11,

S Ξ 12 mod 13,

S Ξ 2 mod 17,

S Ξ 3 mod 19.

Now, from four possible sets of k=3 shares (as k shares are necessary to reconstruct the secret), consider one possible set {1, 12, 2} and recover the secret S from it.

Continued...

Solution: The problem can be solved by Chinese remainder theorem.

For the set {1,12,2}, the equations available are,

$S \equiv 1 \bmod 11,$

$S \equiv 12 \bmod 13,$

$S \equiv 2 \bmod 17,$

Now solving this equation using CRT, M=11 *13*17 = 2431,

M1 = 2431/11=221,

M2 = 2431/13=187,

M3=2431/17=143

$M1^{-1}$, $M2^{-1}$ **and** $M3^{-1}$ **can be calculated using Extended Euclidean Algorithm.**

$M1^{-1} = 1$

$M2^{-1} = 8$

$M3^{-1} = 5$

Now, secret S= ((1*221*1) + (12*187*8) + (2*143*5)) mod 2431

S = 155 mod 2431

EXPONENTIATION AND LOGARITHM

EXPONENTIATION AND LOGARITHM

- Exponentiation and logarithm are inverses of each other.
- a is called the base of the exponentiation or logarithm

$$\textbf{Exponentiation: } y = a^x \quad \rightarrow \quad \textbf{Logarithm: } x = \log_a y$$

EXPONENTIATION

- In cryptography, a common modular operation is **exponentiation.** That is we often need to calculate.

$$\mathbf{y = a^x \bmod n}$$

- The RSA cryptosystem, which uses exponentiation for both encryption and decryption with very large exponents.
- Unfortunately, most computer languages have no operator that can efficiently compute exponentiation, particularly when the exponent is very large.
- To make this type of calculation, we need more efficient algorithms.

EXPONENTIATION

- Fast Exponentiation
  - The idea behind the *square-and-multiply method*
- In traditional algorithms only *multiplication* is used to simulate exponentiation, but the fast exponentiation uses both *squaring* and *multiplication*.
- *square-and-multiply method -* treat the exponent as a binary number of n_b bits (x_0 to x_{nb-1})

Exponentiation

- Fast Exponentiation
  - The idea behind the square-and-multiply method

$$y = a^{\boxed{x_{n_b-1} \times 2^{n_b-1} + x_{n_b-2} \times 2^{n_b-2} + \cdots + x_1 \times 2^1 + x_0 \times 2^0}}$$

in which x_i is 0 or 1

$$y = \boxed{\boxed{a^{2^{n_b-1}}} \text{ or } \boxed{1}} \times \boxed{\boxed{a^{2^{n_b-2}}} \text{ or } \boxed{1}} \times \cdots \times \boxed{\boxed{a^2} \text{ or } \boxed{1}} \times \boxed{\boxed{a} \text{ or } \boxed{1}}$$

Example:

$$\boxed{y = a^9 = a^{1001_2} = a^8 \times 1 \times 1 \times a}$$

Continued…

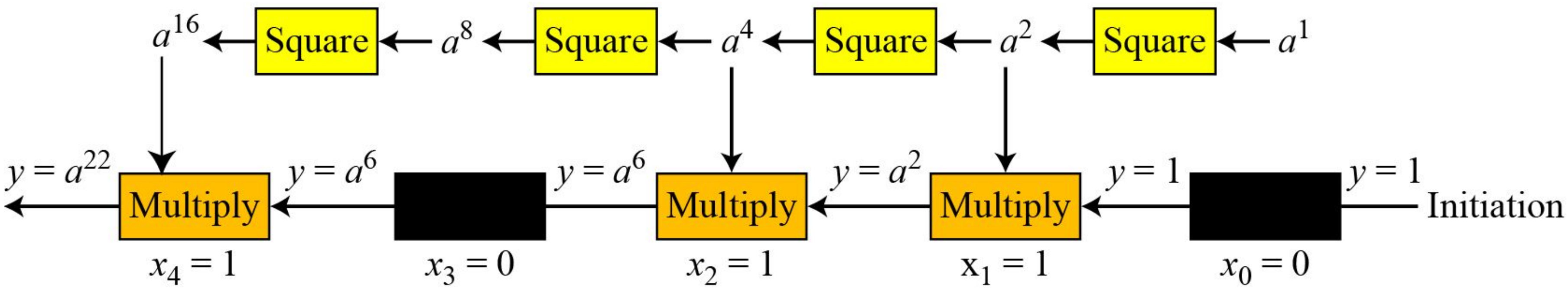
Algorithm 9.7 *Pseudocode for square-and-multiply algorithm*

```
Square_and_Multiply (a, x, n)
{
    y ← 1
    for (i ← 0 to n_b − 1)                 // n_b is the number of bits in x
    {
        if (x_i = 1)   y ← a × y  mod n    // multiply only if the bit is 1

        a ← a^2 mod n                      // squaring is not needed in the last iteration
    }
    return y
}
```

Continued…

- The process for calculating y = a^x
- In this case, x = 22 = $(10110)_2$ in binary.



Continued…

Table 9.3 *Calculation of 17^{22} mod 21*

i	x_i	*Multiplication (Initialization: $y = 1$)*		*Squaring (Initialization: $a = 17$)*
0	0		$\rightarrow$	$a = 17^2 \bmod 21 = 16$
1	1	$y = 1 \times 16 \bmod 21 = 16$	$\rightarrow$	$a = 16^2 \bmod 21 = 4$
2	1	$y = 16 \times 4 \bmod 21 = 1$	$\rightarrow$	$a = 4^2 \bmod 21 = 16$
3	0		$\rightarrow$	$a = 16^2 \bmod 21 = 4$
4	1	$y = 1 \times 4 \bmod 21 = 4$	$\rightarrow$	

Logarithm

- In cryptography we need to discuss modular logarithm.
- If we use exponentiation to encrypt or decrypt, the adversary can use logarithm to attack.
- We need to know how hard it is to reverse the exponentiation.

Logarithm(cont.)

- Order of the Group.
- Example:
  - What is the order of group G = <Z_{21}*, ×>?
    - |G| = φ (21) = φ (3) × φ (7) = 2 × 6 =12. There are 12 elements in this group: 1, 2, 4, 5, 8, 10, 11, 13, 16, 17, 19, and 20. All are relatively prime with 21.

Logarithm(cont.)

- Order of an element: The order of an element is the order of the cyclic group it generates.

- Example:
  - Find the order of all elements in $G = \langle Z_{10}*, \times \rangle$.
  - This group has only $\varphi(10) = 4$ elements: 1, 3, 7, 9.
  - **Lagrange Theorem**- The order of an element divides the order of the group. The only integers that divide 4 are 1, 2, and 4, which means in each case we need to check only these powers to find the order of the element.

a. $1^1 \equiv 1 \bmod (10) \rightarrow \text{ord}(1) = 1$.
b. $3^4 \equiv 1 \bmod (10) \rightarrow \text{ord}(3) = 4$.
c. $7^4 \equiv 1 \bmod (10) \rightarrow \text{ord}(7) = 4$.
d. $9^2 \equiv 1 \bmod (10) \rightarrow \text{ord}(9) = 2$.

Logarithm(cont.)

- Primitive roots
  - In the group $G = <Z_n*, \times>$, when the order of an element is the same as $\varphi(n)$, that element is called the primitive root of the group.
  - Example
    - There are no primitive roots in $G = <Z_8*, \times>$ because no element has the order equal to $\varphi(8) = 4$.

Logarithm(cont.)

- Example
  - the result of $a^i \equiv x \pmod 7$ for the group $G = \langle Z_7^*, \times \rangle$. In this group, $\varphi(7) = 6$.

Table 9.5 *Example 9.50*

		$i = 1$	$i = 2$	$i = 3$	$i = 4$	$i = 5$	$i = 6$
	$a = 1$	$\boxed{x: 1}$	x: 1	x: 1	x: 1	x: 1	x: 1
	$a = 2$	x: 2	x: 4	$\boxed{x: 1}$	x: 2	x: 4	x: 1
Primitive root →	$a = 3$	x: 3	x: 2	x: 6	x: 4	x: 5	$\boxed{x: 1}$
	$a = 4$	x: 4	x: 2	$\boxed{x: 1}$	x: 4	x: 2	x: 1
Primitive root →	$a = 5$	x: 5	x: 4	x: 6	x: 2	x: 3	$\boxed{x: 1}$
	$a = 6$	x: 6	$\boxed{x: 1}$	x: 6	x: 1	x: 6	x: 1

Logarithm(cont.)

The group $G = \langle Z_n^*, \times\rangle$ has primitive roots only if n is 2, 4, p^t, or $2p^t$. <p is an odd prime (not 2) and t is an integer>

For which value of n, does the group $G = \langle Z_n^*, \times\rangle$ have primitive roots: 17, 20, 38, and 50?

Solution

a. $G = \langle Z_{17}^*, \times\rangle$ has primitive roots, 17 is a prime.
b. $G = \langle Z_{20}^*, \times\rangle$ has no primitive roots.
c. $G = \langle Z_{38}^*, \times\rangle$ has primitive roots, $38 = 2 \times 19$ prime.
d. $G = \langle Z_{50}^*, \times\rangle$ has primitive roots, $50 = 2 \times 5^2$ and 5 is a prime.

Logarithm(cont.)

If the group $G = <Z_n^*, \times>$ has any primitive root, the number of primitive roots is $\varphi(\varphi(n))$.

Cyclic Group ***If g is a primitive root in the group, we can generate the set Z_n^* as $Z_n^* = \{g^1, g^2, g^3, \ldots, g^{\varphi(n)}\}$***

The group $G = <Z_{10}^*, \times>$ has two primitive roots because $\varphi(10) = 4$ and $\varphi(\varphi(10)) = 2$. It can be found that the primitive roots are 3 and 7. The following shows how we can create the whole set Z_{10}^* using each primitive root.

$g = 3 \rightarrow$	$g^1 \bmod 10 = 3$	$g^2 \bmod 10 = 9$	$g^3 \bmod 10 = 7$	$g^4 \bmod 10 = 1$
$g = 7 \rightarrow$	$g^1 \bmod 10 = 7$	$g^2 \bmod 10 = 9$	$g^3 \bmod 10 = 3$	$g^4 \bmod 10 = 1$

The group $G = <Z_n^*, \times>$ is a cyclic group if it has primitive roots.
The group $G = <Z_p^*, \times>$ is always cyclic.

Logarithm(cont.)

The idea of Discrete Logarithm

Properties of $G = \langle Z_p^*, \times\rangle$:

1. ***Its elements include all integers from 1 to $p - 1$.***

2. ***It always has primitive roots.***

3. ***It is cyclic. The elements can be created using g^x where x is an integer from 1 to $\varphi(n) = p - 1$.***

4. ***The primitive roots can be thought as the base of logarithm.***

Logarithm(cont.)

Solution to Modular Logarithm Using Discrete Logs

Tabulation of Discrete Logarithms

Table 9.6 *Discrete logarithm for* $\mathbf{G} = \langle \mathbf{Z}_7^*, \times \rangle$

y	*1*	*2*	*3*	*4*	*5*	*6*
$x = \mathrm{L}_3\, y$	6	2	1	4	5	3
$x = \mathrm{L}_5\, y$	6	4	5	2	1	3

Logarithm(cont.)

Find x in each of the following cases:

a. $4 = 3^x \pmod 7$.

b. $6 = 5^x \pmod 7$.

Solution

We can easily use the tabulation of the discrete logarithm:

a. $4 = 3^x \bmod 7 \rightarrow x = L_3 4 \bmod 7 = 4 \bmod 7$

b. $6 = 5^x \bmod 7 \rightarrow x = L_5 6 \bmod 7 = 3 \bmod 7$