Mathematics for Cryptography: prime number and related congruence equtions

Dr. B C Dhara

Department of Information Technology

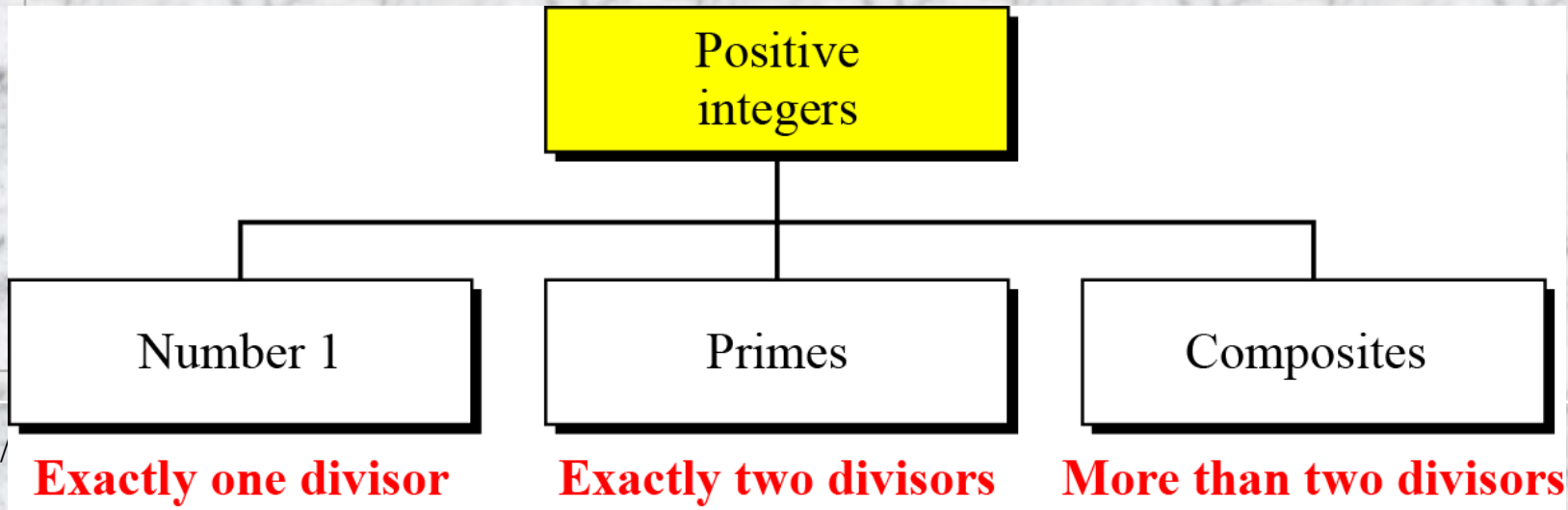
Jadavpur University

Objectives

- Prime numbers and applications in cryptography
- Primality test algorithms and their efficiencies
- Factorization algorithms and their applications in cryptography
- Chinese remainder theorem and its application
- Quadratic congruence
- Modular exponentiation and logarithm

Primes

- *Asymmetric-key cryptography uses primes extensively*
- *A positive integer is prime if and only if it is divisible by exactly two positive integers, 1 and itself*
  - *Otherwise the number is composite*



Prime: example

List the primes smaller than 10.

Solution

There are four primes less than 10: 2, 3, 5, and 7. It is interesting to note that the percentage of primes in the range 1 to 10 is 40%. The percentage decreases as the range increases.

Coprimes

- Two positive integers a and b are relatively prime or coprime, if gcd(a,b)=1
  - 1 is coprime to all positive integers
  - If p is a prime, it is coprime to 1 to p-1
  - Z_n^* is the set of positive integers coprime to n
  - $Z_p^*=\{1,2,\ldots,p-1\}$

Cardinality of primes

- Is there a finite number of primes or is the list infinite? **There is an infinite number of primes**

As a trivial example, assume that the only primes are in the set {2, 3, 5, 7, 11, 13, 17}. Here P = 510510 and P + 1 = 510511. However, 510511 = 19 × 97 × 277; none of these primes were in the original list. Therefore, there are three primes greater than 17.

number

Euclid's Proof. Suppose that $p_1 = 2 < p_2 = 3 < \cdots < p_r$ are all the primes. Let $P = p_1 p_2 \cdots p_r + 1$ and let p be a prime dividing P; then p cannot be any of $p_1, p_2, \ldots, p_r$, otherwise p would divide the difference $P - p_1 p_2 \cdots p_r = 1$, which is impossible. So this prime p is still another prime, and $p_1, p_2, \ldots, p_r$ would not be all the primes. □

on,

i.e., q > p ➔ p cannot be largest prime number

Number of Primes

- Number of prime numbers less or equal to n is

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

- Gauss discovered lower limit
- Lagrange discovered upper limit

Find the number of primes less than 1,000,000.

Solution
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 all primes less than* $\sqrt{n}$

  This is inefficient, but go to start

Is 97 a prime?
The floor of $\sqrt{97} = 9$. The primes less than 9 are 2, 3, 5, and 7. 97 is not divisible by any of these numbers, so 97 is a prime.

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

Sieving method

- Find all prime numbers less than 100, say?

	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100

Euler's phi-function, $\phi(n)$

- *Euler's phi-function, $\phi(n)$, which is sometimes called the Euler's totient function plays a very important role in cryptography*
- Z_n^* contains the number smaller than n and relatively prime to 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, $\phi(n)$ (contd…)

We can combine the above four rules to find the value of $\phi(n)$. For example, if n can be factored as

$$n = p_1^{e_1} \times p_2^{e_2} \times \ldots \times p_k^{e_k}$$

then we combine the third and the fourth rule to find

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

The difficulty of finding ϕ(n) depends on the difficulty of finding the factorization of *n*.

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

Euler's phi-function, $\phi(n)$: example

What is the value of $\phi(13)$? ➔ $\phi(13) = (13 - 1) = 12$

What is the value of $\phi(10)$? ➔ $\phi(10) = \phi(2) \times \phi(5) = 1 \times 4 = 4$

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

Can we say that $\phi(49) = \phi(7) \times \phi(7) = 6 \times 6 = 36$?

No. The third rule applies when *m* and *n* are relatively prime. Here $49 = 7^2$. We need to use the fourth rule: $\phi(49) = 7^2 - 7^1 = 42$.

What is the value of $\phi(240)$? ➔ $240 = 2^4 \times 3^1 \times 5^1$. Then,

$$\phi(240) = (2^4 - 2^3) \times (3^1 - 3^0) \times (5^1 - 5^0) = 64$$

Fermat's Little Theorem

- First version: p is a prime number *a* does not divides p then

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

- Second version: (remove the restriction on *a*) p is a prime number *a* is an integer then

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

- Useful to find exponent

Fermat's Little Theorem (contd…)

Find the result of $6^{10} \bmod 11 \rightarrow 6^{10} \bmod 11 = 1$, this is the first version of Fermat's little theorem where $p = 11$

Find the result of $3^{12} \bmod 11$.

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

Euler's Theorem

- A generalization of Fermat's little theorem
- *First Version: gcd(a,n)=1*

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

- *Second version: relaxed the condition gcd(a,n)=1*

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

- *Useful to find exponent*

Application of Euler's Theorem

Find the result of $6^{24} \bmod 35$ ➔ $6^{24} \bmod 35 = 6^{\phi(35)} \bmod 35 = 1$

Find the result of $20^{62} \bmod 77$ ➔ $\phi(77) = 6 \times 10 = 66$

$20^{62} \bmod 77 = (20 \bmod 77)$ **$(20^{\phi(77)+1} \bmod 77)$** $\bmod 77$ **[2nd version]**

$= (20)(20) \bmod 77 = 15$

Multiplicative Inverses

Euler's theorem can be used to find multiplicative inverses ***modulo a composite***

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

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*

- A formula that could generate a prime

$$\mathbf{M_p = 2^p - 1}$$

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

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: *Fermat Primes*

$$F_n = 2^{2^n} + 1$$

$F_0 = 3$
$F_1 = 5$
$F_2 = 17$
$F_3 = 257$
$F_4 = 65537$
$F_5 = 4294967297 = 641 \times 6700417$ *Not a prime*

PRIMALITY TESTING

- How a large prime number is to be produced?
  - We can use Fermat's or Mersenne's method, but they may fail
  - Choose a large random number and test weather it is prime or not.
    - This is a challenging task
    - Two types of algorithms deal with this problem
      - Deterministic algorithm and probabilistic algorithm and some are hybrid
      - Deterministic gives correct answer but less efficient
      - Probabilistic gives correct answer most 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"
}
```

1. The loop iterates $\sqrt{n}$ times
2. In each iteration: number of operations 2 (division and increment)
3. If we assume only bit-operation, number of operations is $O(\sqrt{2^{n_b}}) = O(2^{n_b/2})$, an exponential complexity

- ***The algorithm can be improved by***
  - ***considering odd numbers only***
  - ***A table of primes between 2 and $\sqrt{n}$***

Divisibility Algorithm: example

Assume n has 200 bits. What is the number of bit operations needed to run the divisibility-test algorithm?

Solution

The bit-operation complexity of this algorithm is $2^{n_b/2}$. This means that the algorithm needs 2^{100} bit operations. On a computer capable of doing 2^{30} bit operations per second, the algorithm needs 2^{70} seconds to do the testing (forever).

Probabilistic algorithms

- A probabilistic algorithm does not give any guarantee about the correctness of the result
  - The probability of error can be made so small that it is almost certain that the answer returned by the algorithm

The probability of mistake can be reduced:

if we run the algorithm more than once with different parameters or using different methods

composite

- If the tested number is prime, algorithm definitely returns a

if the algorithm runs m times, the probability of error may reduced to ε^m

- If the tested number is actually composite, it returns a composite with probability 1-ε or it may return prime with probability ε

Fermat Test

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

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 $561 = 33 \times 17$.

Square Root Test

If n is a prime, $\sqrt{1} \bmod n = \pm 1$.
If n is a composite, $\sqrt{1} \bmod n = \pm 1$ and possibly other values.

What are the square roots of 1 mod n if n is 7 (a prime)?

Solution
The only square roots are 1 and −1. We can see that

$$1^2 = 1 \bmod 7 \qquad (-1)^2 = 1 \bmod 7$$
$$2^2 = 4 \bmod 7 \qquad (-2)^2 = 4 \bmod 7$$
$$3^2 = 2 \bmod 7 \qquad (-3)^2 = 2 \bmod 7$$

Note that we don't have to test 4, 5 and 6 because $4 = -3 \bmod 7$, $5 = -2 \bmod 7$ and $6 = -1 \bmod 7$.

Square Root Test (contd…)

What are the square roots of 1 mod *n* if *n* is 8 (a composite)?

Solution

There are four solutions: 1, 3, 5, and 7 (which is −1). We can see that

$$1^2 = 1 \bmod 8 \qquad (-1)^2 = 1 \bmod 8$$
$$3^2 = 1 \bmod 8 \qquad 5^2 = 1 \bmod 8$$

If $x^2=1 \bmod n$ for $x \neq \pm 1$, number is composite

Square Root Test (contd…)

What are the square roots of 1 mod *n* if *n* is 17 (a prime)?

Solution: There are only two solutions: 1 and −1

$1^2 = 1 \bmod 17$	$(-1)^2 = 1 \bmod 17$
$2^2 = 4 \bmod 17$	$(-2)^2 = 4 \bmod 17$
$3^2 = 9 \bmod 17$	$(-3)^2 = 9 \bmod 17$
$4^2 = 16 \bmod 17$	$(-4)^2 = 16 \bmod 17$
$5^2 = 8 \bmod 17$	$(-5)^2 = 8 \bmod 17$
$6^2 = 2 \bmod 17$	$(-6)^2 = 2 \bmod 17$
$(7)^2 = 15 \bmod 17$	$(-7)^2 = 15 \bmod 17$
$(8)^2 = 13 \bmod 17$	$(-8)^2 = 13 \bmod 17$

Square Root Test (contd…)

What are the square roots of 1 mod *n* if *n* is 22 (a composite)?

Solution

Surprisingly, there are only two solutions, +1 and −1, although 22 is a composite.

$$1^2 = 1 \bmod 22$$
$$(-1)^2 = 1 \bmod 22$$

According to the method 22 is prime.

This test when tell us that number is composite, it is correct

Difficult to do the test

Miller-Rabin Test

- Combines the Fermat test and the square root test to find a strong pseudoprime (a prime with a very high probability)
- $n-1 = m \times 2^k$ (m is odd)
- Idea of Fermat test on base 'a'

$$a^{n-1} = a^{m \times 2^k} = [a^m]^{2^k} = [a^m]^{2^{2^{\cdot^{\cdot^{\cdot^{2}}}}}} \quad (k \text{ times})$$

Miller-Rabin Test (contd…)

- $a^{n-1} \pmod n$ can be computed in one step
- Here, it is done in k+1 steps
  - Advantage: in each step the square root test can be applied and it the test fails then STOP and declare n as composite
  - In testing:
    - if number is 2 only even prime
    - If other even number composite
    - Otherwise, number is odd. So, n-1 (even) can be expressed as $n-1= m2^k$ for $k>0$

This method returns either ***composite*** or ***probably prime***

Miller-Rabin Test (contd…)

- Select a random integer from [2 n-2]
- Compute $T = a^m \pmod n$
  - if $T = \pm 1 \bmod n$, n is probably prime and STOP
  - Else for i=1 to k-1
    - $T = T^2 \bmod n$
    - If T = +1, n is a composite number STOP
    - If T=-1, n is probably prime and STOP
  - n is composite

Pseudocode for Miller-Robin test

Algorithm 9.2 *Pseudocode for Miller-Rabin test*

```
Miller_Rabin_Test (n, a)                    // n is the number; a is the base.
{
    Find m and k such that n − 1 = m × 2^k
    T ← a^m mod n
    if (T = ± 1)   return "a prime"
    for (i ← 1 to k − 1)                    // k − 1 is the maximum number of steps.
    {
        T ← T^2 mod n
        if (T = +1) return "a composite"
        if (T = −1) return "a prime"
    }
    return "a composite"
}
```

Example: Miller-Robin test

Does the number 561 pass the Miller-Rabin test?

Solution

$561 - 1 = 35 \times 2^4$, which means $m = 35$, $k = 4$, and select $a = 2$.

Initialization:	$T = 2^{35} \bmod 561 = 263 \bmod 561$	
$k = 1$:	$T = 263^2 \bmod 561 = 166 \bmod 561$	
$k = 2$:	$T = 166^2 \bmod 561 = 67 \bmod 561$	
$k = 3$:	$T = 67^2 \bmod 561 = +1 \bmod 561$	$\rightarrow$ **a composite**

561= 3 * 187

Example: Miller-Robin test (contd…)

We already know that 27 is not a prime. Let us apply the Miller-Rabin test.

Solution

$27 - 1 = 13 \times 2^1$, which means that $m = 13$, $k = 1$, and let $a = 2$

In this case, because $k - 1 = 0$, we should do only the initialization step: $T = 2^{13} \bmod 27 = 11 \bmod 27$

However, because the algorithm never enters the loop, it returns a composite.

Example: Miller-Robin test (contd…)

We know that 61 is a prime, let us see if it passes the Miller-Rabin test.

Solution

We use base 2.

$61 - 1 = 15 \times 2^2 \rightarrow m = 15 \quad k = 2 \quad a = 2$

Initialization: $T = 2^{15} \bmod 61 = 11 \bmod 61$

$k = 1$ $T = 11^2 \bmod 61 = -1 \bmod 61 \rightarrow$ **a prime**

$n = 252601$, $n-1 = 2^3 \cdot 31575$.
Choose $a = 85132$.

$$a^{31575} \equiv 191102 \pmod{n}$$
$$a^{2 \cdot 31575} \equiv 184829 \pmod{n}$$
$$a^{2^2 \cdot 31575} \equiv 1 \pmod{n}$$

Conclusion: n is ***composite***.
(184829 is a square root of 1, mod n, different from ± 1.)

$n = 3057601$, $n-1 = 2^6 \cdot 47775$.
Choose $a = 99908 \pmod{n}$.

$$a^{47775} \equiv 1193206 \pmod{n}$$
$$a^{2 \cdot 47775} \equiv 2286397 \pmod{n}$$
$$a^{2^2 \cdot 47775} \equiv 235899 \pmod{n}$$
$$a^{2^3 \cdot 47775} \equiv 1 \pmod{n}$$

Conclusion: n is ***composite***.
(235899 is a square root of 1, mod n, different from ± 1.)

$n = 95721889$, $n-1 = 2^5 \cdot 2991309$.
Choose $a = 21906436$.

$$a^{2991309} \equiv 373440 \pmod{n}$$
$$a^{2 \cdot 2991309} \equiv 86363216 \pmod{n}$$
$$a^{2^2 \cdot 2991309} \equiv 93382930 \pmod{n}$$
$$a^{2^3 \cdot 2991309} \equiv 31803553 \pmod{n}$$
$$a^{2^4 \cdot 2991309} \equiv a^{(n-1)/2} \equiv 63099174 \pmod{n}$$

Conclusion: n *is* ***composite***.

$n = 104717$, $n-1 = 2^2 \cdot 26179$.
Choose $a = 96152$.

$$a^{26179} \equiv 1 \pmod{n}$$

Conclusion: n is ***probably prime***.

$n = 577757$, $n-1 = 2^2 \cdot 144439$.
Choose $a = 314997 \pmod{n}$.

$$a^{144439} \equiv 373220 \pmod{n}$$
$$a^{2 \cdot 144439} \equiv 577756 \equiv -1 \pmod{n}$$

Conclusion: n is ***probably prime***.

$n = 101089$, $n-1 = 2^5 \cdot 3159$.
Choose $a = 5$.

$$a^{3159} \equiv 101088 \equiv -1 \pmod{n}$$

Conclusion: n is ***probably prime***.

$n = 280001$, $n-1 = 2^6 \cdot 4375$.
Choose $a = 105532$.

$$a^{4375} \equiv 236926 \pmod{n}$$
$$a^{2 \cdot 4375} \equiv 168999 \pmod{n}$$
$$a^{2^2 \cdot 4375} \equiv 280000 \equiv -1 \pmod{n}$$

Conclusion: n is ***probably prime***.

Recommended Primality Test

- *The most popular primality test is a combination of the divisibility test and the Miller-Rabin test.*

Example: Recommended Primality Test

The number 4033 is a composite (37 × 109). Does it pass the recommended primality test?

Solution

1. Perform the divisibility tests first. The numbers 2, 3, 5, 7, 11, 17, and 23 are not divisors of 4033.
2. Perform the Miller-Rabin test with a base of 2, $4033 - 1 = 63 \times 26$, which means m is 63 and k is 6.

Initialization: $T \equiv 2^{63} \pmod{4033} \equiv 3521 \pmod{4033}$

$k = 1$ $\quad T \equiv T^2 \equiv 3521^2 \pmod{4033} \equiv -1 \pmod{4033}$ $\rightarrow$ **Passes**

No comments

Example: Recommended Primality Test (contd…)

3. But we are not satisfied. We continue with another base, 3.

Initialization: $T \equiv 3^{63} \pmod{4033} \equiv 3551 \pmod{4033}$

$k = 1$	$T \equiv T^2 \equiv 3551^2 \pmod{4033} \equiv 2443 \pmod{4033}$
$k = 2$	$T \equiv T^2 \equiv 2443^2 \pmod{4033} \equiv 3442 \pmod{4033}$
$k = 3$	$T \equiv T^2 \equiv 3442^2 \pmod{4033} \equiv 2443 \pmod{4033}$
$k = 4$	$T \equiv T^2 \equiv 2443^2 \pmod{4033} \equiv 3442 \pmod{4033}$
$k = 5$	$T \equiv T^2 \equiv 3442^2 \pmod{4033} \equiv 2443 \pmod{4033}$ **→Failed (composite)**

AKS primality test

- Given an integer n (≥ 2) and integer a coprime to n, n is prime if and only if the polynomial congruence relation $(x+a)^n \equiv x^n+a \bmod n$
- **AKS primality test (Agrawal–Kayal–Saxena primality test)** is **deterministically correct** for any general number
- The maximum running time of the algorithm can be expressed as a polynomial over the number of digits in the target number, $O((\log_2 n_b)^{12})$

AKS primality test (contd...)

- The algorithm is guaranteed to distinguish deterministically whether the target number is prime or composite
- The correctness of AKS is not conditional on any subsidiary unproven hypothesis

Assume *n* has 200 bits. What is the number of bit operations needed to run the AKS algorithm?

Solution

This algorithm needs only $(\log_2 200)^{12} = 39{,}547{,}615{,}483$ bit operations. On a computer capable of doing 1 billion bit operations per second, the algorithm needs only 40 seconds.

FACTORIZATION

- *Factorization plays a very important role in the security of several public-key cryptosystems*

Topics discussed in this section:

Fundamental Theorem of Arithmetic
Factorization Methods
Fermat Method
Pollard $p - 1$ Method
Pollard rho Method
More Efficient Methods

Fundamental Theorem of Arithmetic

- **Fundamental theorem of arithmetic (FTA)**, also called the **unique factorization theorem** or the **unique-prime-factorization theorem**, states that
- Every integer greater than 1either is a prime number itself or can be represented as the product of prime numbers; moreover, this representation is unique, up to (except for) the order of the factors
  - For example, $1200 = 2^4 \times 3^1 \times 5^2 = 5\times2\times5\times2\times3\times2\times2 = ...$

FTA (contd...)

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

Greatest Common Divisor

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

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

Least Common Multiplier

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

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

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

Factorization methods

- *Trial Division Method*

Algorithm 9.3 *Pseudocode for trial-division factorization*

```
Trial_Division_Factorization (n)          // n is the number to be factored
{
    a ← 2
    while (a ≤ √n )
    {
        while (n mod a = 0)
        {
            output a                      // Factors are output one by one
            n = n / a
        }
        a ← a + 1
    }
    if (n > 1) output n                   // n has no more factors
}
```

Exponential complexity algorithm

Example: *Trial Division Method*

Use the trial division algorithm to find the factors of 1233.

Solution

We run a program based on the algorithm and get the following result.

$$1233 = 3^2 \times 137$$

Use the trial division algorithm to find the factors of 1523357784.

Solution

We run a program based on the algorithm and get the following result.

$$1523357784 = 2^3 \times 3^2 \times 13 \times 37 \times 43987$$

Fermat Method

- this method factorizes ***n*** into two integers ***a*** and ***b*** (not necessary prime) so that ***n = a×b***
- Try to find ***a*** and ***b*** close to each other
- If ***a*** $\geq \sqrt{n}$ then ***b*** $\leq \sqrt{n}$

Pseudocode for Fermat factorization

Algorithm 9.4 *Pseudocode for Fermat factorization*

```
Feramat_Factorization (n)          // n is the number to be factored
{
    x ← √n                         // smallest integer greater than √n
    while (x < n)
    {
    w ← x^2 − n
    if (w is perfect square)  y ← √w;  a ← x+y;  b ← x−y;   return a and b
    x ← x + 1
    }
}
```

Complexity of the method is close to subexponential

Pollard p – 1 Method

- This method finds a prime factor p of a number n based on the condition that p-1 has no factor larger than B
  - $p = \gcd(2^{B!} - 1, n)$

Algorithm 9.5 *Pseudocode for Pollard p −1 factorization*

```
Pollard_ (p − 1) _Factorization (n, B)
{                    // n is the number to be factored
   a ← 2
   e ← 2
   while (e ≤ B)
   {
      a ← a^e mod n
      e ← e + 1
   }
   p ← gcd (a −1, n)
   if 1 < p < n    return p
   return failure
}
```

Complexity: $O(2^{n_b})$, n_b is the number of bits in B

This algorithm may fail
Probability of success is very small unless B is very close to $\sqrt{n}$

Example: Pollard's p-1 methods

- Factor 1403 using pollard's p-1 method
- We take a=2 and evaluate $2^{k!}$ mod 1403 for k=2,3,…, then find gcd($2^{k!}$-1,1403)

$2^{2!} \equiv 4$ mod 1403	**gcd(3,1403)=1, a trivial**
$2^{3!} = 4^3 \equiv 64$ mod 1403	gcd(63,1403)=1, a trivial
$2^{4!} = 64^4 \equiv 142$ mod 1403	gcd(141,1403)=1, a trivial
$2^{5!} = 142^5 \equiv 794$ mod 1403	gcd(793,1403)=61, a non-trivial 1403 = 61 × 23

- 61 is a factor

Example: Pollard's p-1 methods

- Factor 2993 using pollard's p-1 method

$2^{2!} \equiv 4 \bmod 2993$	gcd(3,2993)=1, a trivial
$2^{3!} = 4^3 \equiv 64 \bmod 2993$	gcd(63,2993)=1, a trivial
$2^{4!} = 64^4 \equiv 1451 \bmod 2993$	gcd(1450,2993)=1, a trivial
$2^{5!} = 142^5 \equiv 1395 \bmod 2993$	gcd(1394,2993)=41, a non-trivial $2993 = 41 \times 73$

- 41 is a factor

Pollard rho Method

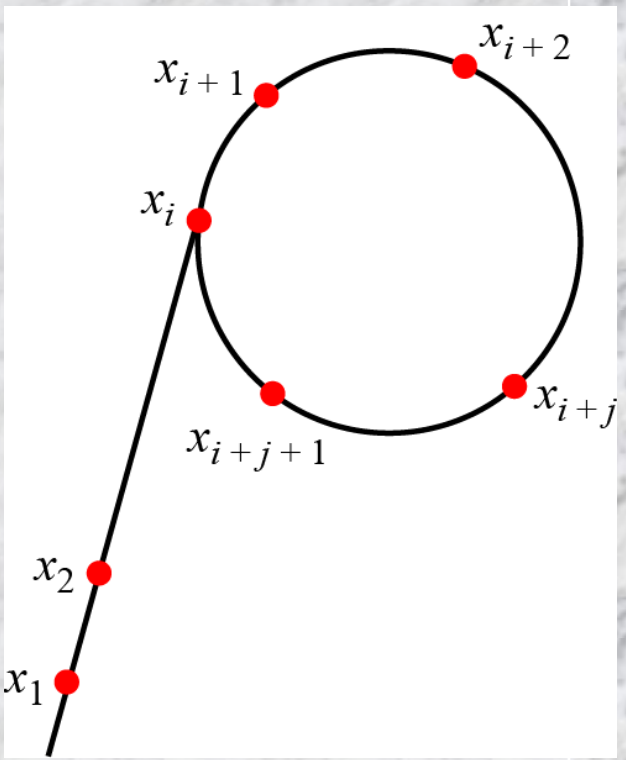
- Suppose n is the number to be factored, n =pq, where p is a prime factor
  - Note that p is unknown
- consider two distinct numbers x1 and x2 taken from Z_n such that x1 ≡ x2 mod p
- p | (x1 –x2) and p | n ➔ p | gcd(x1-x2, n)
- p is unknown, if we compute gcd(x_i'-x_j', n) for all pair $1 \le i' < j' \le r$, for some r, we will find nontrivial factor

Pollard rho Method (contd…)

- Select a small random number x_1 as seed
- Compute x2 so that n does not divides x_1-x_2
  - $x_2=x_1^2+1$
- If gcd(x_1 - x_2, n) is not trivial we get answer, STOP
- If gcd ==1, select another pair of number and repeat the process until we get nontrivial gcd
- List of seeds gives a repeated sequence of number on a circle, as figure in next slide

Pollard rho Method (contd…)

- The figure looks like a ‘rho’
- To decrease the number of iterations:
  - Start with (x_0, x_0) and computes $(x_1, x_2), (x_2, x_4), (x_3, x_6), \ldots, (x_i, x_{2i})$ using $x_{i+1}=f(x_i)$



Pseudocode for Pollard rho method and Complexity

- Complexity: $O(2^{nb/4})$

Algorithm 9.6 *Pseudocode for Pollard rho method*

```
Pollard_ rho _Factorization (n, B)          // n is the number to be factored
{
    x ← 2
    y ← 2
    p ← 1
    while (p = 1)
    {
        x ← f(x) mod n
        y ← f (f (y) mod n) mod n
        p ← gcd (x − y, n)
    }
    return p                                // if p = n, the program has failed
}
```

Example: Pollard rho method

Find factors of 434617. The result is 709 (434617 = 709 × 613).

Table 9.2 *Values of x, y, and p in Example 9.33*

x	y	p
2	2	1
5	26	1
26	23713	1
677	142292	1
23713	157099	1
346589	52128	1
142292	41831	1
380320	68775	1
157099	427553	1
369457	2634	1
52128	63593	1
102901	161353	1
41831	64890	1
64520	21979	1
68775	16309	709

Some efficient methods

Quadratic Sieve

The method uses a sieving procedure to find the value of x^2 mod n.

$$O(e^C), \text{ where } C \approx (\ln n \ln\ln n)^{1/2}$$

Number Field Sieve

The method uses a sieving procedure in an algebraic ring structure to find $x^2 \equiv y^2$ mod n.

$$O(e^C) \text{ where } C \approx 2 (\ln n)^{1/3} (\ln\ln n)^{2/3}$$

Example

Assume that there is a computer that can perform 230 (almost 1 billion) bit operations per second. What is the approximate time required for this computer to factor an integer of 100 decimal digits using one of the following methods?

a. Quadratic sieve method b. Number field sieve method

Solution

A number with 100 decimal digits has almost 300 bits ($n = 2^{300}$). $\ln(2^{300}) = 207$ and $\ln\ln(2^{300}) = 5$.

a. $(207)^{1/2} \times (5)^{1/2} = 14 \times 2.23 \approx 32$ e^{32} $(e^{32}) / (2^{30}) \approx 20$ hours.

b. $(207)^{1/3} \times (5)^{2/2} = 6 \times 3 \approx 18$. e^{18} $(e^{18}) / (2^{30}) \approx 6$ seconds.

CHINESE REMAINDER THEOREM

- *The Chinese remainder theorem (CRT) is used to solve a set of congruent equations with one variable but different moduli, which are relatively prime, as shown below:*

$$x \equiv a_1 \pmod{m_1}$$
$$x \equiv a_2 \pmod{m_2}$$
$$\cdots$$
$$x \equiv a_k \pmod{m_k}$$

Example

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

$$x \equiv 2 \pmod 3$$
$$x \equiv 3 \pmod 5$$
$$x \equiv 2 \pmod 7$$

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 $23 \equiv 2 \pmod 7$.

Solution: Chinese Remainder Theorem

1. Find $M = m_1 \times m_2 \times \ldots \times m_k$. This is the common modulus.
2. Find $M_1 = M/m_1$, $M_2 = M/m_2$, …, $M_k = M/m_k$.
3. Find the multiplicative inverse of M_i, respect to modulo m_i, for $1 \le i \le k$
   Call the inverses M_1^{-1}, M_2^{-1}, …, M_k^{-1}.
4. The solution to the simultaneous equations is

$$x \equiv a_1 \pmod{m_1}$$
$$x \equiv a_2 \pmod{m_2}$$
$$\ldots$$
$$x \equiv a_k \pmod{m_k}$$

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

Example: CRT

Find the solution to the simultaneous equations:

$$x \equiv 2 \pmod 3$$
$$x \equiv 3 \pmod 5$$
$$x \equiv 2 \pmod 7$$

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

Example: CRT

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.

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

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

Example: CRT

Assume we need to calculate $z = x + y$ where $x = 123$ and $y = 334$, but our system accepts only numbers less than 100. 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

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

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

QUADRATIC CONGRUENCE

- Linear congruence $ax \equiv c \pmod{n}$ or $ax + b \equiv c \pmod{n}$
- *In cryptography, we also need to discuss quadratic congruence—that is, equations of the form* $a_2x^2 + a_1x + a_0 \equiv 0 \pmod{n}$. *We limit our discussion to quadratic equations in which* $a_2 = 1$ *and* $a_1 = 0$, *that is equations of the form*

$$x^2 \equiv a \pmod{n}.$$

Topics discussed in this section:

Quadratic Congruence Modulo a Prime
Quadratic Congruence Modulo a Composite

Quadratic Congruence Modulo a Prime

Example 1:
The equation $x^2 \equiv 3 \pmod{11}$ has two solutions, $x \equiv 5 \pmod{11}$ and $x \equiv -5 \pmod{11}$. But note that $-5 \equiv 6 \pmod{11}$, so the solutions are actually 5 and 6. Also note that these two solutions are incongruent.

Example 2:
The equation $x^2 \equiv 2 \pmod{11}$ has no solution. No integer x can be found such that its square is 2 mod 11.

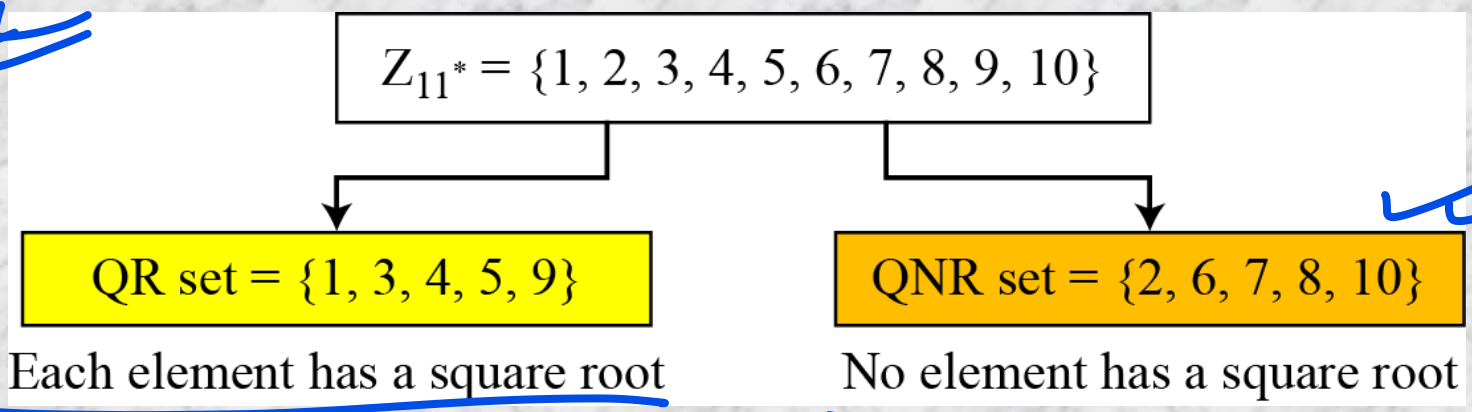
Quadratic Residues and Nonresidue
In the equation $x^2 \equiv a \pmod{p}$, a is called a ***quadratic residue (QR)*** *if the equation has two solutions; a is called* ***quadratic nonresidue (QNR)*** *if the equation has no solutions.*

QR-QNR example

If p is prime, Z_p^*, contains half QR and half QNR

There are 10 elements in Z_{11}^*. Exactly five of them are quadratic residues and five of them are nonresidues. In other words, Z_{11}^* is divided into two separate sets, QR and QNR, as shown in Figure 9.4.

Figure 9.4 ***Division of Z_{11}^* elements into QRs and QNRs***



Euler's Criterion to check QR

- *If $a^{(p-1)/2} \equiv 1 \pmod{p}$, a is a quadratic residue modulo p*
- *b. If $a^{(p-1)/2} \equiv -1 \pmod{p}$, a is a quadratic nonresidue modulo p*

To find out if 14 or 16 is a QR in Z_{23}^*, we calculate:

$14^{(23-1)/2} \bmod 23 \rightarrow 22 \bmod 23 \rightarrow -1 \bmod 23$ nonresidue

$16^{(23-1)/2} \bmod 23 \rightarrow 16^{11} \bmod 23 \rightarrow 1 \bmod 23$ residue

Solving Quadratic Equation Modulo a Prime

- Euler criteria can say if an integer a is a QR or QNR in Z_p^*.
  - Cannot find solution to $x^2 \equiv a \pmod{n}$
  - a prime can be either p=4k+1 or p=4k+3
  - Consider the case p=4k+3 and if a is QR then

$$x \equiv a^{(p+1)/4} \pmod{p} \quad \text{and} \quad x \equiv -a^{(p+1)/4} \pmod{p}$$

Example

Solve the following quadratic equations:

a. $x^2 \equiv 3 \pmod{23}$
b. $x^2 \equiv 2 \pmod{11}$
c. $x^2 \equiv 7 \pmod{19}$

Solutions

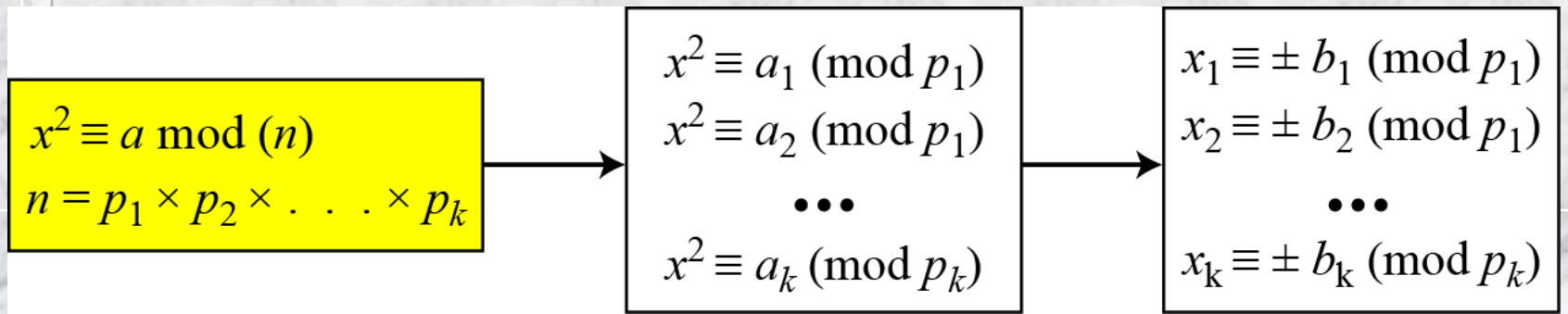
a. $x \equiv \pm 16 \pmod{23}$ $\sqrt{3} \equiv \pm 16 \pmod{23}$.

b. There is no solution for $\sqrt{2}$ in Z11.

c. $x \equiv \pm 11 \pmod{19}$. $\sqrt{7} \equiv \pm 11 \pmod{19}$.

Quadratic Congruence Modulo a Composite

- Can be solved by a set of congruence modulo a prime
- We have factorization of n and then solve each decompose equation (if solvable)
  - k pairs of solution
    - 2^k equation sets can be solved by CRT and obtained 2^k solutions of x



Quadratic Congruence Modulo a Composite

Assume that $x^2 \equiv 36 \pmod{77}$. We know that $77 = 7 \times 11$. We can write $x^2 \equiv 36 \pmod 7 \equiv 1 \pmod 7$ and $x^2 \equiv 36 \pmod{11} \equiv 3 \pmod{11}$

The answers are $x \equiv +1 \pmod 7$, $x \equiv -1 \pmod 7$, $x \equiv +5 \pmod{11}$, and $x \equiv -5 \pmod{11}$. Four sets of equations out of these:

Set 1: $x \equiv +1 \pmod 7$ $\quad x \equiv +5 \pmod{11} \longrightarrow$ -6

Set 2: $x \equiv +1 \pmod 7$ $\quad x \equiv -5 \pmod{11} \longrightarrow$ -27

Set 3: $x \equiv -1 \pmod 7$ $\quad x \equiv +5 \pmod{11} \longrightarrow$ 27

Set 4: $x \equiv -1 \pmod 7$ $\quad x \equiv -5 \pmod{11} \longrightarrow$ 6

The answers are $x = \pm 6$ and ± 27.

Solving a quadratic congruence modulo a composite is as hard as factorization of the modulus.

Exponentiation and Logarithm

- Exponentiation and Logarithm are inverse of each other

Exponentiation: $y = a^x$ $\rightarrow$ **Logarithm:** $x = \log_a y$

Topics discussed in this section:

Exponentiation: modular

Logarithm: modular

Fast exponentiation

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

in which x_i is 0 or 1

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

Example:

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

We need a^{2^0}, a^{2^1}, a^{2^2}, a^{2^3}, …, a^{2^i}, … // square part

If x_i =0 will be multiplied by 1

If x_i =1 will be multiplied by a^{2^i}

} multiplication

Fast exponentiation (contd…)

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
}
```

Example: Fast Exponentiation

The process to calculate $y = a^x$ (for simplicity, modulus is not shown)

In this case, $x = 22 = (10110)_2$ in binary. The exponent has five bits

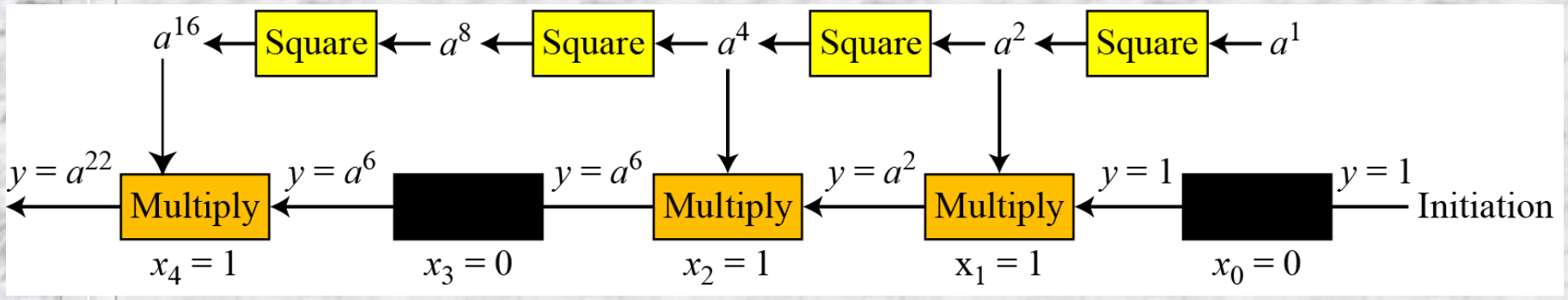


Table 9.3 *Calculation of 17^{22} mod 21*

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

Logarithm

In cryptography, we also need to discuss modular logarithm.

- Exhaustive search
  - To solve $x = \log_a y \pmod n$, continuously compute $y = a^x \bmod n$ until we find the given y

Algorithm 9.8 *Exhaustive search for modular logarithm*

```
Modular_Logarithm (a, y, n)
{
    for (x = 1 to n −1)                          // k is the number of bits in x
    {
        if (y ≡ a^x mod n) return x
    }
    return failure
}
```

Order of the Group

What is the order of group G = $<Z_{21}*, \times>$? $|G| = \phi(21) = \phi(3) \times \phi(7) = 2 \times 6 = 12$. 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.

Order of an Element

Find the order of all elements in G = $<Z_{10}^*, \times>$.

Solution

This group has only $\phi(10) = 4$ elements: 1, 3, 7, 9. We can find the order of each element by trial and error.

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. $9^2 \equiv 1 \bmod (10) \rightarrow \text{ord}(9) = 2$.

Euler's Theorem
If $a \in G$, *then* $a^{\phi(n)} = 1 \bmod n$

$a^i = 1 \bmod n$ holds when $i = \phi(n)$, it also holds for some $i < \phi(n)$

Relation $a^i = 1 \bmod n$ holds at least once

Order of an Element

- $G = \langle Z_8^*, \times \rangle$.

Euler's Theorem
If $a \in G$, *then* $a^{\phi(n)} = 1 \bmod n$

$a^i = 1 \bmod n$ holds when $i = \phi(n)$, it also holds for some $i < \phi(n)$

Relation $a^i = 1 \bmod n$ holds at least once

Table 9.4 *Finding the orders of elements in Example 9.48*

	$i = 1$	$i = 2$	$i = 3$	$i = 4$	$i = 5$	$i = 6$	$i = 7$
$a = 1$	**x: 1**	x: 1	x: 1	x: 1	x: 1	x: 1	x: 1
$a = 3$	x: 3	**x: 1**	x: 3	x: 1	x: 3	x: 1	x: 3
$a = 5$	x: 5	**x: 1**	x: 5	x: 1	x: 5	x: 1	x: 5
$a = 7$	x: 7	**x: 1**	x: 7	x: 1	x: 7	x: 1	x: 7

Primitive Roots

- *In the group G = <Z_n*, ×>, when the order of an element is the same as $\phi(n)$, that element is called the primitive root of the group*

There are no primitive roots in G = <Z_8*, ×> because no element has the order equal to $\phi(8) = 4$. The order of elements are all smaller than 4.

The result of $a^i \equiv x \pmod 7$ for the group G = <Z_7*, ×>. In this group, $\phi(7) = 6$.

	$i = 1$	$i = 2$	$i = 3$	$i = 4$	$i = 5$	$i = 6$
$a = 1$	x: 1	x: 1	x: 1	x: 1	x: 1	x: 1
$a = 2$	x: 2	x: 4	x: 1	x: 2	x: 4	x: 1
Primitive root → $a = 3$	x: 3	x: 2	x: 6	x: 4	x: 5	x: 1
$a = 4$	x: 4	x: 2	x: 1	x: 4	x: 2	x: 1
Primitive root → $a = 5$	x: 5	x: 4	x: 6	x: 2	x: 3	x: 1
$a = 6$	x: 6	x: 1	x: 6	x: 1	x: 6	x: 1

Primitive Roots (contd…)

The group G = $<Z_n^*, \times>$ has primitive roots only if *n* is 2, 4, p^t, or $2p^t$.

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

Solution

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

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

Primitive Roots (contd…)

- Given a group G = $<Z_n^*, \times>$, how can we check whether ***a*** is a primitive root of G?
- Given a group G = $<Z_n^*, \times>$, how can we check all primitive roots of G?
- Given a group G = $<Z_n^*, \times>$, how can we select a primitive root?

Cyclic group

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

Table 9.5 *Example 9.50*

	$i = 1$	$i = 2$	$i = 3$	$i = 4$	$i = 5$	$i = 6$
$a = 1$	$x: 1$	$x: 1$	$x: 1$	$x: 1$	$x: 1$	$x: 1$
$a = 2$	$x: 2$	$x: 4$	$x: 1$	$x: 2$	$x: 4$	$x: 1$
Primitive root → $a = 3$	$x: 3$	$x: 2$	$x: 6$	$x: 4$	$x: 5$	$x: 1$
$a = 4$	$x: 4$	$x: 2$	$x: 1$	$x: 4$	$x: 2$	$x: 1$
Primitive root → $a = 5$	$x: 5$	$x: 4$	$x: 6$	$x: 2$	$x: 3$	$x: 1$
$a = 6$	$x: 6$	$x: 1$	$x: 6$	$x: 1$	$x: 6$	$x: 1$

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* $\phi(n) = p - 1$.

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

Solution to Modular Logarithm Using Discrete Logs

- *Tabulation of Discrete Logarithms: table is pre-calculated and saved*

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

Find

a. 4

b. 6

Table 9.5 *Example 9.50*

	$i = 1$	$i = 2$	$i = 3$	$i = 4$	$i = 5$	$i = 6$
$a = 1$	x: 1	x: 1	x: 1	x: 1	x: 1	x: 1
$a = 2$	x: 2	x: 4	x: 1	x: 2	x: 4	x: 1
Primitive root → $a = 3$	x: 3	x: 2	x: 6	x: 4	x: 5	x: 1
$a = 4$	x: 4	x: 2	x: 1	x: 4	x: 2	x: 1
Primitive root → $a = 5$	x: 5	x: 4	x: 6	x: 2	x: 3	x: 1
$a = 6$	x: 6	x: 1	x: 6	x: 1	x: 6	x: 1

Solution to Modular Logarithm Using Discrete Logs

- *Using Properties of Discrete Logarithms*

Table 9.7 *Comparison of traditional and discrete logarithms*

Traditional Logarithm	*Discrete Logarithms*
$\log_a 1 = 0$	$L_g 1 \equiv 0 \pmod{\phi(n)}$
$\log_a (x \times y) = \log_a x + \log_a y$	$L_g(x \times y) \equiv (L_g x + L_g y) \pmod{\phi(n)}$
$\log_a x^k = k \times \log_a x$	$L_g x^k \equiv k \times L_g x \pmod{\phi(n)}$

Tabulation and properties of discrete logarithms cannot be used when n is very large

The discrete logarithm problem has the same complexity as the factorization problem.