# Number-Theoretic Algorithms (RSA and related algorithms)

Chapter 31, CLRS book

# Outline

- Modular arithmetic
- RSA encryption scheme
- Miller-Rabin algorithm (a probabilistic algorithm)

# Modular Arithmetic

# Integers

- $a \mid b$: $a$ divides $b$, $a$ is a divisor of $b$.
- $\gcd(a,b)$: greatest common divisor of $a$ and $b$.
- Coprime or relatively prime: $\gcd(a,b) = 1$.
- Euclid's algorithm: compute $\gcd(a,b)$.
- Extended Euclid's algorithm: compute integers $x$ and $y$ such that $ax + by = \gcd(a,b)$.

# Integers modulo $n$

- Let $n \geq 2$ be an integer.
- Definition: $a$ is congruent to $b$ modulo $n$, written $a \equiv b \mod n$, if $n \mid (a-b)$, i.e., $a$ and $b$ have the same remainder when divided by $n$.
- Note: $a \equiv b \bmod n$ and $a = b \bmod n$ are different.
- Definition: $[a]_n = \{x \in Z : x \equiv a \bmod n\}$.
- $[a]_n$ is called a residue class modulo $n$, and $a$ is a representative of that class.

- There are exactly $n$ residue classes modulo $n$ :

  $[0],\ [1],\ [2],\ \ldots,\ [n-1].$

- If $x \in [a],\ y \in [b],$ then $x + y \in [a+b]$ and $x \cdot y \in [a \cdot b]$.

- Define addition and multiplication for residue classes:

$$[a] +_n [b] = [a+b]$$

$$[a] \cdot_n [b] = [a \cdot b].$$

# Group

- A group, denoted by $(G,*)$, is a set $G$ with a binary operation $*$ such that
  1. $\forall x, y \in G,\ x * y \in G$ (closure)
  1. $x*(y*z) = (x*y)*z$ (associativity)
  2. $\exists\, e \in G$ s.t. $\forall x \in G,\ e*x = x*e = x$ (identity)
  3. $\forall x \in G,\ \exists\, y \in G$ s.t. $x*y = y*x = e$ (inverse)
- A group $(G,*)$ is abelian if $\forall x, y \in G,\ x*y = y*x$.
- Examples: $(Z,+)$, $(Q,+)$, $(Q\setminus\{0\},\ \times)$, $(R,+)$, $(R\setminus\{0\},\ \times)$.

- Define $Z_n = \{[0],\ [1],\ ...,\ [n-1]\}$.
- Or, more conveniently, $Z_n = \{0,\ 1,\ ...,\ n-1\}$.
- $(Z_n, +)$ forms an abelian additive group.
- For $a, b \in Z_n$,
  - $a + b = (a + b) \bmod n$. (Or, $[a] + [b] = [a + b] = [a + b \bmod n]$.)
  - 0 is the identity element.
  - The inverse of $a$, denoted by $-a$, is $n - a$.
- When doing addition/substraction in $Z_n$, just do the regular addition/substraction and reduce the result modulo $n$.
  - In $Z_{10}$, $5 + 5 + 9 + 4 + 6 + 2 + 8 + 3 = ?$

- $(Z_n, *)$ is not a group, because $0^{-1}$ does not exist.
- Even if we exclude 0 and consider only $Z_n^+ = Z_n \setminus \{0\}$, $(Z_n^+, *)$ is not necessarily a group; some $a^{-1}$ may not exist.
- For $a \in Z_n$, $a^{-1}$ exists if and only if $\gcd(a, n) = 1$.

- Let $Z_n^* = \{a \in Z_n : \ \gcd(a, n) = 1\}$.
- $(Z_n, *)$ is an abelian multiplicative group.
- $a * b = ab \mod n$.
  - $a * b = ab \mod n$.
  - 1 is the identity element.
  - The inverse of $a$, written $a^{-1}$, can be computed by the Extended Euclidean Algorithm.
- For example, $Z_{12}^* = \{1, 5, 7, 11\}$. $5 * 7 = 35 \bmod 12 = 11$.
- Q: How many elements are there in $Z_n^*$ ?

- Euler's totient function:

$$\varphi(n) = \left|Z_n^*\right|$$

$$= \left|\left\{a : \ a \in Z_n \text{ and } \gcd(a,n) = 1\right\}\right|$$

- Facts:

1. $\varphi(p^e) = (p-1)p^{e-1}$ for prime $p$
2. $\varphi(ab) = \varphi(a)\varphi(b)$ if $\gcd(a,b) = 1$

- Let $G$ be a (multiplicative) finite group.
- Lagrange's theorem: For any element $a \in G$, $a^{|G|} = e$.
- Corollary: For any element $a \in G$, $a^m = a^{m \bmod |G|}$.

- Euler's theorem:

  If $a \in Z_n^*$ (for any $n > 1$), then $a^{\varphi(n)} = 1$ in $Z_n^*$.

- Fermat's little theorem:

  If $a \in Z_p^*$ ($p$ a prime), then $a^{\varphi(p)} = a^{p-1} = 1$ in $Z_p^*$.

# Example: $n = 15$

- $Z_{15}^{*} = \{1, 2, 4, 7, 8, 11, 13, 14\}$
- $\left|Z_{15}^{*}\right| = \varphi(15) = \varphi(3) \times \varphi(5) = 2 \times 4 = 8$
- $$\begin{array}{l|cccccccc} a \in Z_{15}^{*}: & 1 & 2 & 4 & 7 & 8 & 11 & 13 & 14 \\ \hline \text{ord}(a): & 1 & 4 & 2 & 4 & 4 & 2 & 4 & 2 \end{array}$$
- $\text{ord}(a)$: smallest integer $k$ such that $a^{k} = 1$.
- $a^{\varphi(n)} = a^{8} = 1$
- $13^{816243240481} = ?$

# Algorithms

- $\gcd(a,b)$
- $a^{-1} \bmod n$
- $a^k \bmod n$
- Running time: $O(\log^3 n)$
- Here we assume $a, b \in Z_n$.

# Euclid's Algorithm

- Given $n > a > b \geq 0$, compute $\gcd(a,b)$. $(a,b \in Z_n)$
- Theorem: If $b = 0$, $\gcd(a,b) = a$.

  If $b > 0$, $\gcd(a,b) = \gcd(b, a \bmod b)$
- Euclid$(a,b)$

  if $b = 0$

  then return$(a)$

  else return $\big($Euclid$(b, a \bmod b)\big)$
- The number of recursive calls to Euclid is $O(\log n)$.
- Computing $a \bmod b$ takes $O(\log^2 n)$.

# Extended Euclidean Algorithm

Given $a > b \geq 0$, compute $x, y$ such that $d = \gcd(a,b) = ax + by$.

Example: $\gcd(299, 221) = ?$

$$299 = 1 \times 221 + 78$$
$$221 = 2 \times 78 + 65$$
$$78 = 1 \times 65 + 13$$
$$65 = 5 \cdot 13 + 0$$

$$\begin{aligned}\gcd(229,221) = 13 &= 78 - 65 \\ &= 78 - (221 - 2 \times 78) = 3 \cdot 78 - 221 \\ &= 3 \times (299 - 1 \cdot 221) - 221 \\ &= 3 \times 299 - 4 \times 221\end{aligned}$$

# Extended Euclidean Algorithm

Given $a > b \geq 0$, compute $d, x, y$ such that $\gcd(a,b) = d = ax + by$.

Extended - Euclid$(a,b)$

if $b = 0$ then

return$(a, 1, 0)$

else

$(d', x', y') \leftarrow \left(\text{Extended - Euclid}(b, a \bmod b)\right)$

$(d, x, y) \leftarrow \left(d', y', x' - \lfloor a/b \rfloor y'\right)$

return$(d, x, y)$

# Correctness Proof

- If $b = 0, \quad \gcd(a,b) = a = a \cdot 1 + b \cdot 0.$

  The returned answer $(a,1,0)$ is correct.

- If $(d', x', y')$ is correct,

  $\Rightarrow \gcd(b,\ a \bmod b) = d' = b \cdot x' + (a \bmod b) \cdot y'$

  $\Rightarrow \gcd(b,\ a \bmod b) = d' = b \cdot x' + \left(a - \lfloor a/b \rfloor \cdot b\right) \cdot y'$

  $\Rightarrow \gcd(a,\ b) = d' = a \cdot y' + b \cdot \left(x' - \lfloor a/b \rfloor y'\right)$

  $\Rightarrow (d,\ x,\ y) \leftarrow \left(d',\ y',\ x' - \lfloor a/b \rfloor y'\right)$ is correct

# How to compute $a^{-1} \bmod n$?

- Compute $a^{-1}$ in $Z_n^*$.
- $a^{-1}$ exists if and only if $\gcd(a,n) = 1$.
- Use extended Euclidean algorithm to find $x$, $y$ such that $ax + ny = \gcd(a,n) = 1 \quad (\text{in } Z)$

$$\Rightarrow [a][x] + [n][y] = [1]$$

$$\Rightarrow [a][x] = [1] \qquad (\text{since } [n] = [0])$$

$$\Rightarrow [a]^{-1} = [x].$$

- Note: may omit [ ], but reduce everything modulo $n$.

## Example

- Compute $15^{-1} \mod 47$.
- Using extended Euclidean algorithm, we obtain

  $\gcd(15, 47) = 1 = 15 \times 22 - 47 \times 7$

  $15^{-1} \mod 47 = 22$

  That is, $15^{-1} = 22$ in $Z_{47}^{*}$

# Algorithm: Square-and-Multiply($x$, $c$, $n$)

Comment: compute $x^c \bmod n$, where $c = c_k c_{k-1} \ldots c_0$ in binary.

$z \leftarrow 1$

for $i \leftarrow k$ downto 0 do

    $z \leftarrow z^2 \bmod n$

    if $c_i = 1$ then $z \leftarrow (z \cdot x) \bmod n$

return $(z)$

Note: $x^c = \begin{cases} \left(x^{\lfloor c/2 \rfloor}\right)^2 & \text{if c is even} \\ \left(x^{\lfloor c/2 \rfloor}\right)^2 \cdot x & \text{if } c \text{ is odd} \end{cases}$

# Example: $11^{23} \bmod 187$

$23 = 10111_b$

$z \leftarrow 1$

$z \leftarrow z^2 \cdot 11 \bmod 187 = 11$ (square and multiply)

$z \leftarrow z^2 \bmod 187 = 121$ (square)

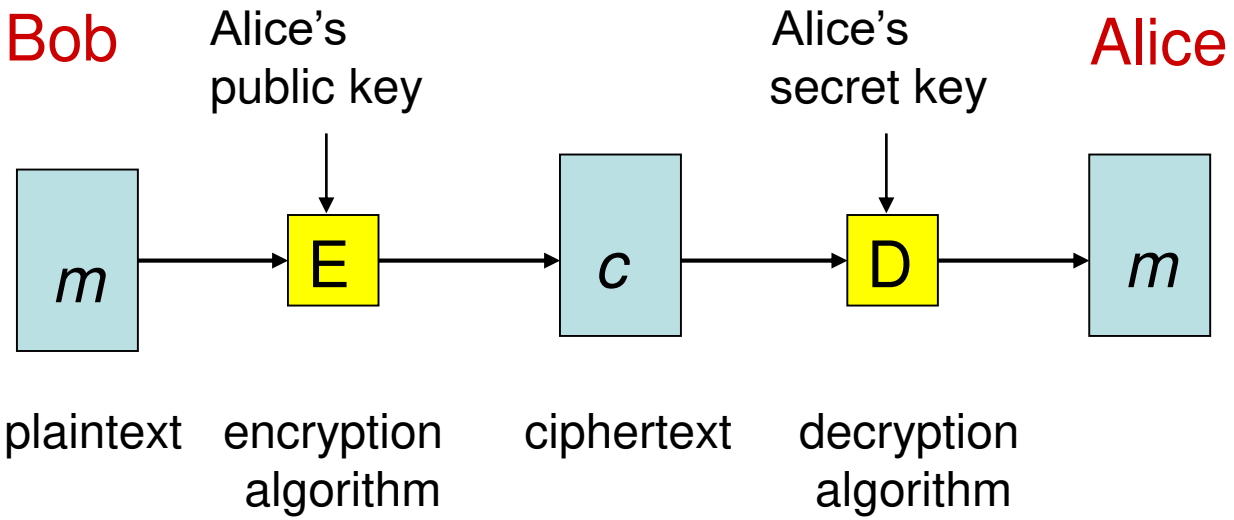
$z \leftarrow z^2 \cdot 11 \bmod 187 = 44$ (square and multiply)

$z \leftarrow z^2 \cdot 11 \bmod 187 = 165$ (square and multiply)

$z \leftarrow z^2 \cdot 11 \bmod 187 = 88$ (square and multiply)

# RSA Encryption

# Public-key Encryption



# The RSA Cryptosystem

- By Rivest, Shamir & Adleman of MIT in 1977.
- Best known and most widely used public-key scheme.
- Based on the assumed one-way property of modular powering:

$$f : x \to x^e \mod n \qquad \text{(easy)}$$

$$f^{-1} : x^e \to x \mod n \qquad \text{(hard)}$$

- In turn based on the hardness of integer factorization.

# Idea behind RSA

It works in group $Z_n^*$. Let $x \in Z_n^*$ be a message.

Encryption (easy): $x \xrightarrow{\text{RSA}} x^e$

Decryption (hard): $x \xleftarrow{\text{RSA}^{-1}} x^e$

Decryption (easy with "trapdoor"): $x \xleftarrow{\text{RSA}^{-1}} x^e$

Looking for a "trapdoor": $(x^e)^d = x.$

If $d$ is a number such that $ed \equiv 1 \bmod \varphi(n),$ then

$ed = k\varphi(n) + 1$ for some $k,$ and

$$(x^e)^d = x^{ed} = x^{\varphi(n)k+1} = \left(x^{\varphi(n)}\right)^k \cdot x = 1 \cdot x = x.$$

# RSA Cryptosystem

- Key generation:

  (a) Choose large primes $p$ and $q$, and let $n := pq$.

  (b) Choose $e$ $(1 < e < \varphi(n))$ coprime to $\varphi(n)$, and compute $d := e^{-1} \bmod \varphi(n)$. $(ed \equiv 1 \bmod \varphi(n).)$

  (c) Public key: $pk = (n, e)$. Secret key: $sk = (n, d)$.

- Encryption: $E_{pk}(x) := x^e \bmod n$, where $x \in Z_n^*$.

- Decryption: $D_{sk}(y) := y^d \bmod n$, where $y \in Z_n^*$.

# RSA Example: Key Setup

- Select two primes: $p = 17,\ q = 11.$
- Compute the modulus $n = pq = 187.$
- Compute $\varphi(n) = (p-1)(q-1) = 160.$
- Select $e$ between 0 and 160 such that $\gcd(e,160) = 1.$

  Say $e = 7.$
- Compute $d = e^{-1} \bmod \varphi(n) = 7^{-1} \bmod 160 = 23$

  (using extended Euclid's algorithm).
- Public key: $pk = (e,\ n) = (7,\ 187).$
- Secret key: $sk = (d,\ n) = (23,\ 187).$

# RSA Example: Encryption & Decryption

- Suppose $m = 88$.
- Encryption: $c = m^e \bmod n = 88^7 \bmod 187 = 11$.
- Decryption: $m = c^d \bmod n = 11^{23} \bmod 187 = 88$.
- When computing $11^{23} \bmod 187$, we do not first compute $11^{23}$ and then reduce it modulo 187.
- Rather, use square-and-multiply, and reduce intermediate results modulo 187 whenever they get bigger than 187.

# Attacks on RSA

# Attacks on RSA

- There are many attacks on RSA:
  - brute-force key search
  - mathematical attacks
  - timing attacks
  - chosen ciphertext attacks
- The most important one is integer factorization:

  If the adversary can factor $n$ into $pq$. Then he can calculate $\varphi(n) = (p-1)(q-1)$ and the secret key $d = e^{-1} \bmod \varphi(n)$.

# Integer Factorization

- A difficult problem.
- More and more efficient algorithms have been developed.
- In 1977, RSA challenged researchers to decode a ciphertext encrypted with a modulus $n$ of 129 digits (428 bits). Prize: \$100. RSA thought it would take quadrillion years to break the code using fastest algorithms and computers of that time. Solved in 1994.
- In 1991, RSA put forward more challenges (called RSA numbers), with prizes, to encourage research on factorization.

# RSA Numbers

- Each RSA number is a semiprime. (A number is semiprime if it is the product of two primes.)
- There are two labeling schemes.
  - by the number of decimal digits:
    RSA-100, ..., RSA-500, RSA-617.
  - by the number of bits:
    RSA-576, 640, 704, 768, 896, 1024, 1536, 2048.

# RSA Numbers which have been factored

- RSA-100 (332 bits), 1991, 7 MIPS-year, Quadratic Sieve.
- RSA-110 (365 bits), 1992, 75 MIPS-year, QS.
- RSA-120 (398 bits), 1993, 830 MIPS-year, QS.
- RSA-129 (428 bits), 1994, 5000 MIPS-year, QS.
- RSA-130 (431 bits), 1996, 1000 MIPS-year, GNFS.
- RSA-140 (465 bits), 1999, 2000 MIPS-year, GNFS.
- RSA-155 (512 bits), 1999, 8000 MIPS-year, GNFS.
- RSA-160 (530 bits), 2003, Lattice Sieve.
- RSA-576 (174 digits), 2003, Lattice Sieve.
- RSA-640 (193 digits), 2005, Lattice Sieve.
- RSA-200 (663 bits), 2005, Lattice Sieve.

RSA-200 =

27,997,833,911,221,327,870,829,467,638,

722,601,621,070,446,786,955,428,537,560,

009,929,326,128,400,107,609,345,671,052,

955,360,856,061,822,351,910,951,365,788,

637,105,954,482,006,576,775,098,580,557,

613,579,098,734,950,144,178,863,178,946,

295,187,237,869,221,823,983.

## Remark

- In light of current factorization technologies, RSA recommends using an $n$ of 1024-2048 bits.

# Generating large primes

To set up an RSA cryptosystem,

we need two large primes p and q.

## How to generate a large prime number?

- Generate a random odd number $n$ of desired size.
- Test if $n$ is prime.
- If not, discard it and try a different number.

# Primality test: Is $n$ a prime?

- Can it be solved in polynomial time?
- A long standing open problem until 2002.
- AKS(Agrawal, Kayal, Saxena): $O\left((\log n)^{12+\varepsilon}\right)$.
  - Later improved by others to $O\left((\log n)^{10.5}\right)$, and then to $O\left((\log n)^{6+\varepsilon}\right)$.
- In practice, Miller-Rabin's probabilistic algorithm is still the most popular --- much faster, $O\left((\log n)^{3}\right)$.

# Miller-Rabin primality test: Is $n$ a prime?

- Using some characteristic property of prime numbers:
  - $n$ is prime $\Leftrightarrow$ $\forall a \in 2..\sqrt{n},\ a$ does not divide $n$.
- Miller-Rabin's idea: look for some property $P(a)$ s.t.
  - $n$ is prime $\Rightarrow$ For all $a \in Z_n^*,\ P(a) = true$

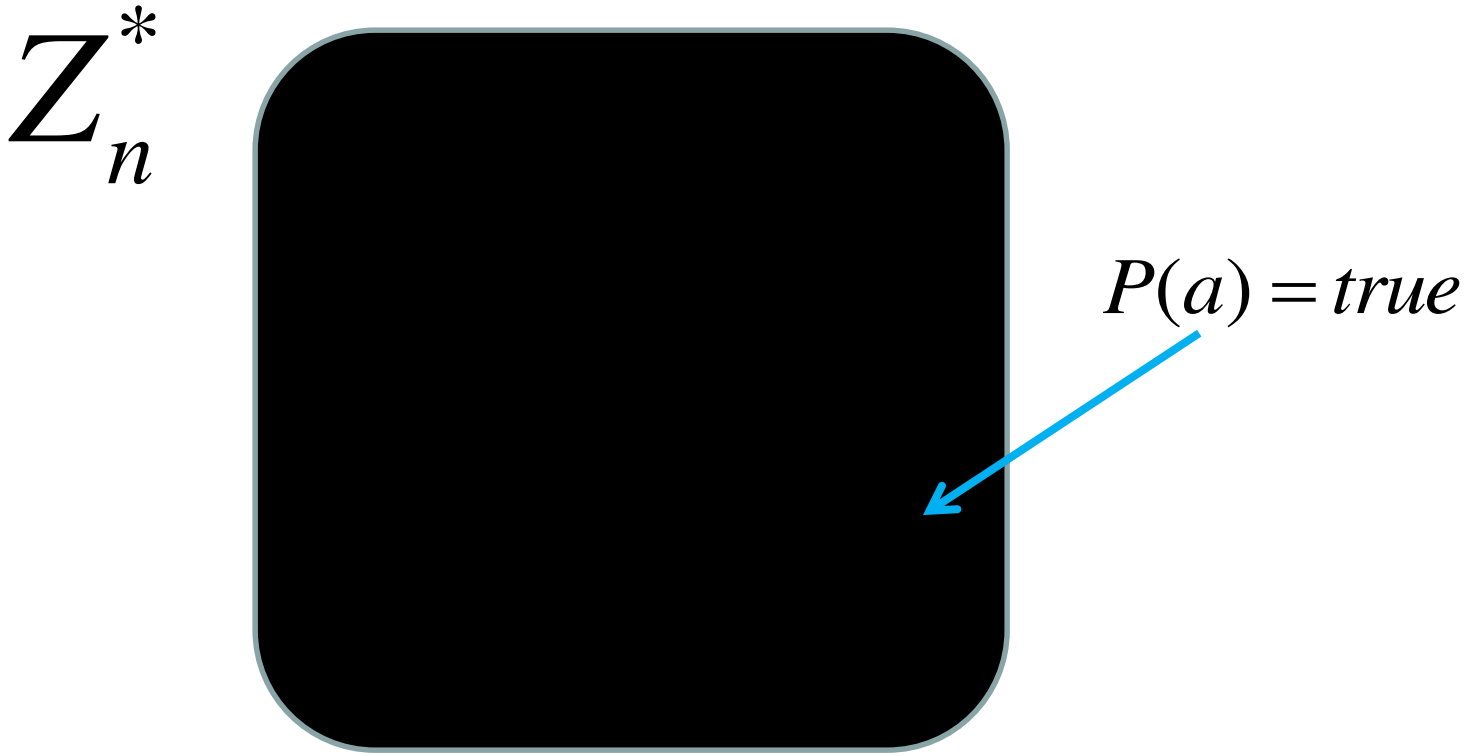
    $n$ not prime $\Rightarrow$ For at most a portion $1/k$ of elements

    $$a \in Z_n^*,\ P(a) = true$$
- Algorithm: Randomly pick $t$ elements $a \in Z_n^*$.

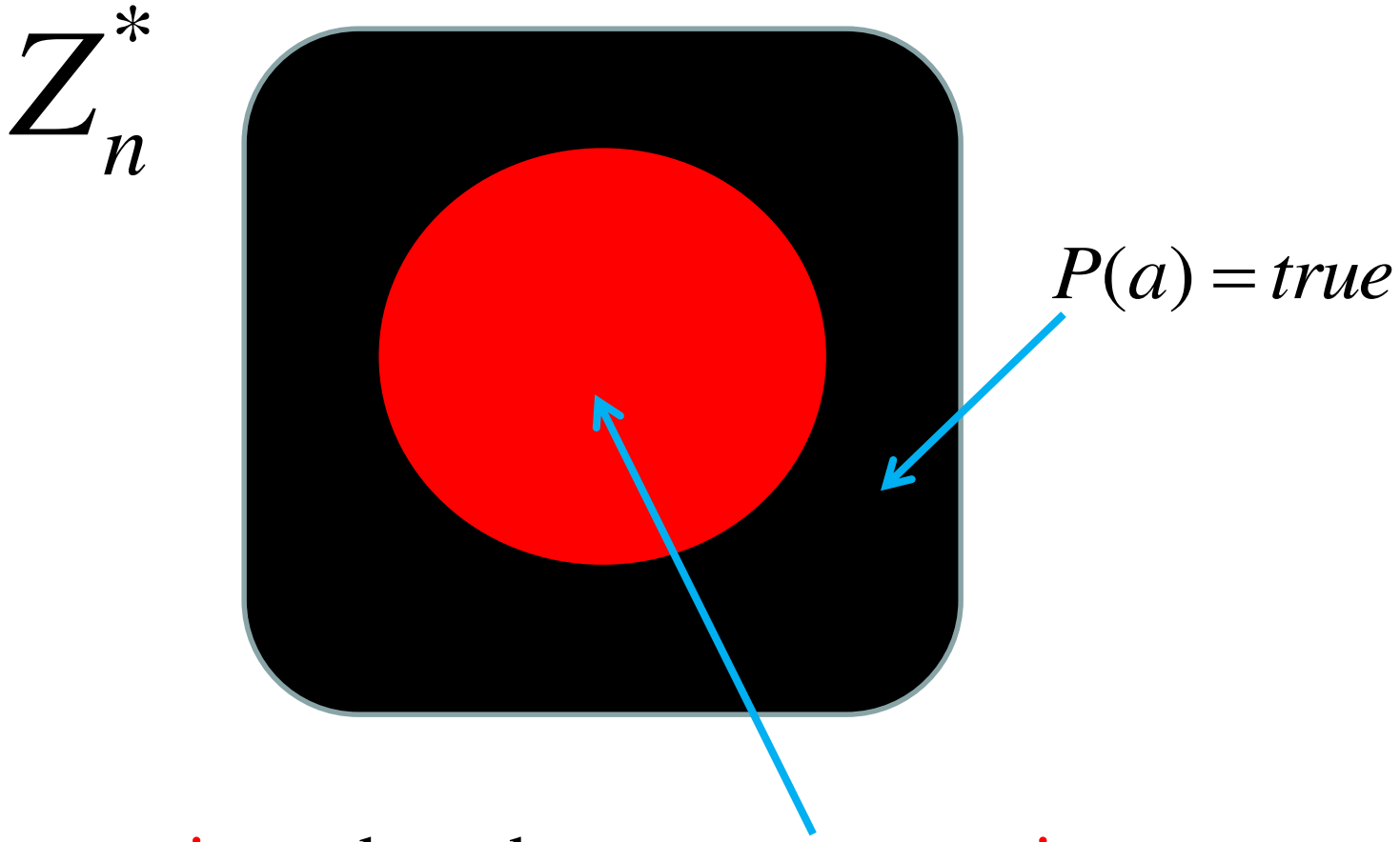
  If $P(a)$ is true for all of them then return prime

  else return composite.
- A "prime" answer may be incorrect with probability

  $\leq (1/k)^t$




If $n$ is prime, then for all $a \in Z_n^*$, $P(a)$ is true.




If $n$ is not prime, then there are strong witnesses,

which are elements $a \in Z_n^*$ s.t $P(a) = false$.

Say, at most $1/k$ of $Z_n^*$ are black.

## The property $P(a)$

- Write $n-1 = u2^k$, where $u$ is odd.
- Let $P(a) = \begin{cases} a^u \equiv 1 \mod n \text{ or} \\ a^{u2^i} \equiv -1 \mod n \text{ for some } i,\ 0 \le i \le k-1 \end{cases}$
- Consider the sequence

$$a^u,\ a^{u2},\ a^{u2^2},\ \ldots,\ a^{u2^{k-1}}$$

- If $n$ is prime, then $P(a) = true$ for all $a \in Z_n^*$.
- If $n$ is an odd composite and not a prime power, then at most one half of the elements $a \in Z_n^*$ are black (i.e., $P(a) = true$).
- A composite number $n$ is a prime power if $n = p^e$ for some prime $p$ and integer $e \geq 2$; a perfect power if $n = k^e$ for some integer $k$ and $e \geq 2$.)

# Algorithm: Miller-Rabin primality test

- Input: integer $n > 2$ and parameter $t$
- Output: a decision as to whether $n$ is prime or composite

1. if $n$ is even, return "composite"
2. if $n$ is a perfect power, return "composite"
3. for $i := 1$ to $t$ do

   choose a random integer $a$, $2 \le a \le n-1$

   if $\gcd(a,n) \ne 1$, return "composite"

   if $a$ is a strong witness, return "composite"
4. return ("prime")

## Analysis: Miller-Rabin primality test

- If the algorithm answers "composite", it is always correct.
- If the algorithm answers "prime", it may or may not be correct.
- The algorithm gives a wrong answer if $n$ is composite but the algorithm fails to find a strong witness in $t$ iterations.
- This may happen with probability at most $2^{-t}$.
- Actually, at most $4^{-t}$, by a more sophisticated analysis.

# Monte Carlo algorithms

- A Monte Carlo algorithm is a probabilistic algorithm
  - which always gives an answer
  - but sometimes the answer may be incorrect.
- A Monte Carlo algorithm for a decision problem is yes-biased if its "yes" answer is always correct but a "no" answer may be incorrect with some error probability.
- A $t$-iteration Miller-Rabin is a "composite"-biased Monte Carlo algorithm with error probability at most $1/4^t$.

# Las Vegas algorithms

- A Las Vegas algorithm is a probabilistic algorithm
  - which may sometimes fail to give an answer
  - but never gives an incorrect one
- A Las Vegas algorithm can be converted into a Monte Carlo algorithm.