Public-Key Cryptography and RSA

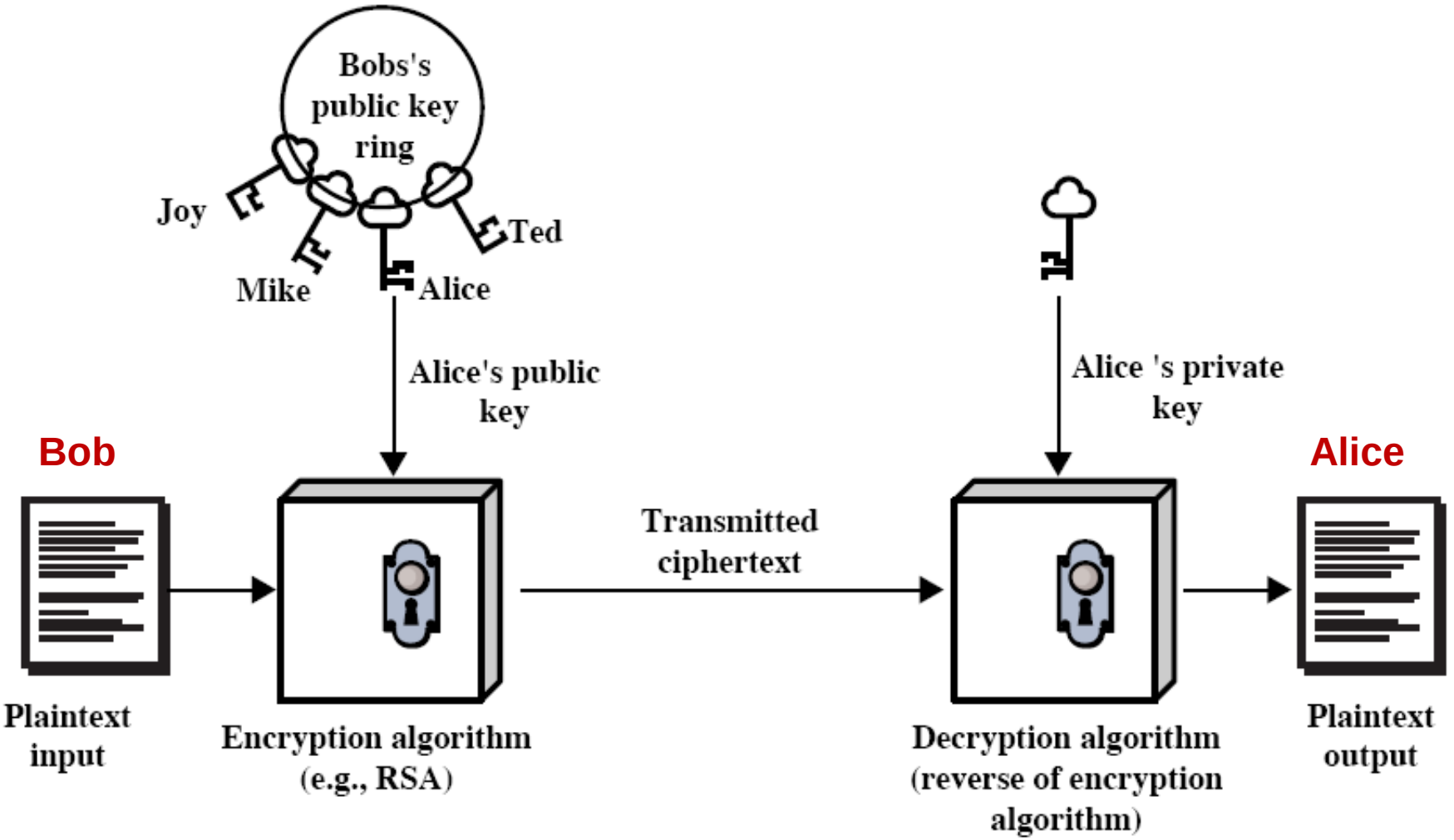
CSE 651: Introduction to Network Security

Abstract

- We will discuss
  - The concept of public-key cryptography
  - RSA algorithm
  - Attacks on RSA
- Suggested reading:
  - Sections 4.2, 4.3, 8.1, 8.2, 8.4
  - Chapter 9

Public-Key Cryptography

- Also known as asymmetric-key cryptography.
- Each user has a pair of keys: a public key and a private key.
- The public key is used for encryption.
  - The key is known to the public.
- The private key is used for decryption.
  - The key is only known to the owner.


Bobs's public key ring
Joy
Ted
Mike
Alice
Alice's public key
Alice 's private key
Bob
Alice
Transmitted ciphertext
Plaintext input
Encryption algorithm (e.g., RSA)
Decryption algorithm (reverse of encryption algorithm)
Plaintext output

Why Public-Key Cryptography?

- Developed to address two main issues:
  - key distribution
  - digital signatures
- Invented by Whitfield Diffie & Martin Hellman 1976.

One-way function with trapdoor

Easy: $x \xrightarrow{f} y$

Hard: $x \xleftarrow{f^{-1}} y$

Easy: $x \xleftarrow[\text{trapdoor}]{f^{-1}} y$

Use trapdoor as the private key.

- Many public-key cryptosystems are based on trapdoor one-way functions.

Modular Arithmetic

Mathematics used in RSA

(Sections 4.2, 4.3, 8.1, 8.2, 8.4)

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: computes $\gcd(a,b)$.
- Extended Euclid's algorithm: computes integers x and y such that $ax + by = \gcd(a,b)$.

Euclidean Algorithm

Comment: compute $\gcd(a,b)$, where $a > b > 1$.

$r_0 := a$

$r_1 := b$

for $i := 1, 2, \ldots$ until $r_{n+1} = 0$

$r_{i+1} := r_{i-1} \bmod r_i$

return (r_n)

Extended Euclidean Algorithm: Example

$\gcd(299, 221) = ?$

$$299 = 1 \cdot 221 + 78$$

$$221 = 2 \cdot 78 + 65$$

$$78 = 1 \cdot 65 + 13$$

$$65 = 5 \cdot 13 + 0$$

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

Group

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

Integers modulo n

- Let $n \geq 2$ be an integer.
- Def: 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.
- Def: $[a]_n = \{\text{all integers congruent to } a \text{ modulo } n\}$.
- $[a]_n$ is called a residue calss modulo n, and a is a representative of that class.

- $[a]_n = [b]_n$ if and only if $a \equiv b \bmod n$.
- 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:

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

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

- 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 $\gcd(a, n) = 1$.
- $\gcd(a, n) = 1 \Leftrightarrow ax + ny = 1$ for some integers x and y

$$\Leftrightarrow [a] \cdot [x] + [n] \cdot [y] = [1] \text{ in } Z_n$$

$$\Leftrightarrow [a] \cdot [x] = [1] \text{ in } Z_n$$

$$\Leftrightarrow [a]^{-1} = [x] \text{ in } Z_n$$

- 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 computated 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^*?

How many elements are there in Z_n^*?

- Euler's totient function:

$$\varphi(n) = \left|Z_n^*\right| = \left|\left\{a : \ 1 \le a \le n, \ \gcd(a,n) = 1\right\}\right|$$

- Facts:

1. $\varphi(p) = (p-1)$ for prime p.
2. $\varphi(pq) = \varphi(p)\varphi(q)$ if $\gcd(p,q) = 1$.

- Let G be a (multiplicative) finite group.
- The order of $a \in G$, written ord(a), is the smallest positive integer t such that $a^t = e$. (e, identity element.)
- The order of G, ord(G), is the number of elements in G.
- Example: Consider Z_{15}^*
  - $\text{ord}(Z_{15}^*) = \left|Z_{15}^*\right| = \varphi(15)$
  - $\text{ord}(8) = 4,$ since $8^2 = 64 \bmod 15 = 4$

$$8^3 = (8^2 \cdot 8) \bmod 15 = (4 \cdot 8) \bmod 15 = 2$$

$$8^4 = (8^2 \cdot 8^2) \bmod 15 = (4 \cdot 4) \bmod 15 = 1$$

Example: $n = 15$

- $Z_{15}^{*} = \{1, 2, 4, 7, 8, 11, 13, 14\}$
- $\left|Z_{15}^{*}\right| = \varphi(15) = \varphi(3) \times \varphi(5) = 2 \times 4 = 8$
- $$\begin{array}{l|cccccccc} a \in Z_{15}^{*}: & 1 & 2 & 4 & 7 & 8 & 11 & 13 & 14 \\ \hline \text{ord}(a): & 1 & 4 & 2 & 4 & 4 & 2 & 4 & 2 \end{array}$$
- For all $a \in Z_{15}^{*}$, we have $a^{\varphi(15)} = a^{8} = 1$

- Theorem: For any element $a \in G$, $\text{ord}(a) \mid \text{ord}(G)$.
- Corollary: For any element $a \in G$, $a^{\text{ord}(G)} = e$.
- Fermat's little theorem:

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

  If $a \in Z_n^*$, then $a^{\varphi(n)} = 1$. $(\because \text{ord}(Z_n^*) = \varphi(n).)$

  That is, for any integer a relatively prime to n,

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

The Chinese Remainder Problem

- A problem described in an ancient Chinese arithmetic book.
- Problem: We have a number of things, but we do not know exactly how many. If we count them by threes we have two left over. If we count them by fives we have three left over. If we count them by sevens we have two left over. How many things are there?

$$x \equiv 2 \bmod 3, \quad x \equiv 3 \bmod 5, \quad x \equiv 2 \bmod 7$$

$$x = ?$$

Chinese remainder theorem

If integers $n_1,\ldots,n_k$ are pairwise coprime,

then the system of congruences

$$\begin{cases} x \equiv a_1 \bmod n_1 \\ x \equiv a_2 \bmod n_2 \\ \quad\vdots \\ x \equiv a_k \bmod n_k \end{cases}$$

has a unique solution in Z_N :

$$x_0 = \sum_{i=1}^{k} a_i N_i \left(N_i^{-1} \bmod n_i \right) \mod N$$

where $N = n_1 n_2 \ldots n_k$, and $N_i = N/n_i$.

Furthermore, every integer $x \in \left[x_0 \right]_N$ is a solution.

Example: Chinese remainder theorem

Suppose

$$x \equiv 1 \mod 3$$

$$x \equiv 6 \mod 7$$

$$x \equiv 8 \mod 10$$

By the Chinese remainer theorem, the solution is:

$$x \equiv 1 \times 70 \times (70^{-1} \bmod 3) + 6 \times 30 \times (30^{-1} \bmod 7) + 8 \times 21 \times (21^{-1} \bmod 10)$$

$$\equiv 1 \times 70 \times (1^{-1} \bmod 3) + 6 \times 30 \times (2^{-1} \bmod 7) + 8 \times 21 \times (1^{-1} \bmod 10)$$

$$\equiv 1 \times 70 \times 1 + 6 \times 30 \times 4 + 8 \times 21 \times 1 \mod 210$$

$$\equiv 958 \mod 210$$

$$\equiv 118 \mod 210$$

Chinese remainder theorem (another version)

- $N = n_1 n_2 \cdots n_k$ (the numbers n_i are pairwise coprime)
- There is a one-to-one correspondence :

$$Z_N \quad \leftrightarrow \quad Z_{n_1} \times \cdots \times Z_{n_k} \qquad \left(\text{also, } Z_N^* \leftrightarrow Z_{n_1}^* \times \cdots \times Z_{n_2}^*\right)$$

$$A \quad \leftrightarrow \quad (a_1, \ \ldots, a_k), \ \text{where } A \in Z_N \text{ and } a_i = A \bmod n_i$$

- $A \quad \rightarrow \quad ?$
- $? \quad \leftarrow \quad (a_1, \ \ldots, a_k)$

- One-to-one correspondence :

$$Z_N \quad \leftrightarrow \quad Z_{n_1} \times \cdots \times Z_{n_k}$$

$$A \quad \leftrightarrow \quad (a_1, \ \ldots, a_k)$$

- Operations in Z_N can be performed individually in each Z_{n_i}.

If

$$\begin{cases} A \quad \leftrightarrow \quad (a_1, \ \ldots, a_k) \\ B \quad \leftrightarrow \quad (b_1, \ \ldots, b_k) \end{cases}$$

then

$$A \pm B \quad \leftrightarrow \quad (a_1 \pm b_1, \ \ldots, a_k \pm b_k)$$

$$A \times B \quad \leftrightarrow \quad (a_1 \times b_1, \ \ldots, a_k \times b_k)$$

$$A \div B \quad \leftrightarrow \quad (a_1 \div b_1, \ \ldots, a_k \div b_k) \quad \text{if } B \in Z_N^*$$

$$\uparrow \qquad\qquad \uparrow \qquad\qquad \uparrow$$

$$\text{mod } N \qquad \text{mod } n_1 \qquad \text{mod } n_k$$

Example: Chinese remainder theorem

- Suppose we want to compute 8×11 in Z_{15}.
- $Z_{15} \leftrightarrow Z_3 \times Z_5 \quad \left(Z_{15}^* \leftrightarrow Z_3^* \times Z_5^*\right)$

  $8 \leftrightarrow (2,\ 3) = (8 \bmod 3,\ 8 \bmod 5)$

  $11 \leftrightarrow (2,\ 1) = (11 \bmod 3,\ 11 \bmod 5)$

  $8 \times 11 \leftrightarrow (2 \times 2,\ 3 \times 1) = (1,\ 3).$
- $x \leftrightarrow (1,\ 3)$
- Solve $\begin{cases} x \equiv 1 \bmod 3 \\ x \equiv 3 \bmod 5 \end{cases} \Rightarrow x = 13$

Important Problems

- $\gcd(a, b)$,
- $a^k \bmod n$,
- $a^{-1} \bmod n$
- Can be done in $O(\log^3 n)$ time.

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 $ax + ny = \gcd(a, n) = 1$

  $\Rightarrow ax = 1$ (because $ny = 0$ in Z_n)

  $\Rightarrow a^{-1} = x$.
- Note: every computation is reduced modulo n.

Example

- Compute $15^{-1} \mod 47$.

$47 = 15 \times 3 + 2$ (divide 47 by 15; remainder $= 2$)

$15 = 2 \times 7 + 1$ (divide 15 by 2; remainder $= 1$)

$$\begin{aligned} 1 &= 15 - 2 \times 7 && (\mod 47) \\ &= 15 - (47 - 15 \times 3) \times 7 && (\mod 47) \\ &= 15 \times 22 - 47 \times 7 && (\mod 47) \\ &= 15 \times 22 && (\mod 47) \end{aligned}$$

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

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:

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

$$f^{-1} : y \to \sqrt[e]{y} \mod n \qquad \text{(hard)}$$

Idea behind RSA

It works in group Z_n^*.

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

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

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

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

Setting up an RSA Cryptosystem

- A user wishing to set up an RSA cryptosystem will:
  - Choose a pair of public/private keys: (PU, PR).
  - Publish the public (encryption) key.
  - Keep secret the private (decryption) key.

RSA Key Setup

- Select two large primes p and q at random.
- Compute $n = pq$. Note: $\varphi(n) = (p-1)(q-1)$.
- Select an encryption key e satisfying $1 < e < \varphi(n)$ and $\gcd(e, \varphi(n)) = 1$. (i.e., $e \in Z^*_{\varphi(n)}$, $e \neq 1$.)
- Compute the descryption key: $d = e^{-1} \bmod \varphi(n)$.
  - $ed \equiv 1 \bmod \varphi(n)$.
  - d is the inverse of e mod $\varphi(n)$.
- Public key: $PU = (n, e)$. Private key: $PR = (n, d)$.
- Important: p, q, and $\varphi(n)$ must be kept secret.

RSA Encryption and Decryption

- Suppose Bob is to send a secret message m to Alice.
- To encrypt, Bob will
  - obtain Alice's public key $PU_{\text{Alice}} = \{e, n\}$.
  - encrypt m as $c = m^e \bmod n$.
  - Note: $m \in Z_n^*$.
- To decrypt the ciphertext c, Alice will compute
  - $m = c^d \bmod n$, using her private key $PR_{\text{Alice}} = \{d, n\}$.
- What key will Alice use to encrypt her reply to Bob?

Why RSA Works

- The settig of RSA is the group $\left(Z_n^*, \square\right)$:
  - Plaintexts and ciphertexts are elements in Z_n^*.
  - Recall: $Z_n^* = \{x: \ 0 < x < n, \ \gcd(x, n) = 1 \}$.
  - Z_n^* has $\varphi(n)$ elements. (The group Z_n^* has order $\varphi(n)$.)
  - In group $\left(Z_n^*, \square\right)$, for any $x \in Z_n^*$, we have $x^{\varphi(n)} = 1$.
  - We have chosen e, d such that $ed \equiv 1 \mod \varphi(n)$, i.e., $ed = k\varphi(n) + 1$ for some positive integer k.
  - For $x \in Z_n^*$, $\left(x^e\right)^d = x^{ed} = x^{k\varphi(n)+1} = \left(x^{\varphi(n)}\right)^k x = x$.

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 $\gcd(e,160) = 1.$

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

  (using extended Euclid's algorithm).
- Public key: $PU = (e,\ n) = (7,\ 187).$
- Private key: $PR = (d,\ n) = (23,\ 187).$

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, when conmputing 11^{23}, reduce the intermediate results modulo 187 whenever they get bigger than 187.

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 $z \leftarrow (z \times x) \bmod n$

} i.e., $z \leftarrow \left(z \times x^{c_i}\right) \bmod n$

return (z)

Note: At the end of iteration i, $z = x^{c_k \ldots c_i}$.

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)

Encryption Key *e*

- To speed up encryption, small values are usually used for *e*.
- Popular choices are $3,\ 17 = 2^4 + 1,\ 65537 = 2^{16} + 1.$ These values have only two 1's in their binary representation.
- There is an interesting attack on small *e*.

Low encryption exponent attack

- A message m sent to e users who employ the same encryption exponent e is not protected by RSA.
- Say, $e = 3,$ and Bob sends a message m to three receipients encrypted as:

$$c_1 = m^3 \bmod n_1,\ \ c_2 = m^3 \bmod n_2,\ \ c_3 = m^3 \bmod n_3.$$

- Eve intercepts the three ciphertexts, and recovers m:
  - $m^3 \equiv c_1 \bmod n_1,\ \ m^3 \equiv c_2 \bmod n_2,\ \ m^3 \equiv c_3 \bmod n_3.$
  - By CRT, $m^3 \equiv c \bmod n_1 n_2 n_3$ for some $c < n_1 n_2 n_3$.
  - Also, $m^3 < n_1 n_2 n_3$. So, $m^3 = c,$ and $m = \sqrt[3]{c}$.

Decryption Key d

- One may be tempted to use a small d to speed up decryption.
- Unfortunately, that is risky.
- Wiener's attack: If $d < n^{1/4}/3$ and $p < q < 2p$, then the decryption exponent d can be computed from (n, e). (The condition $p < q < 2p$ often holds in practice.)
- CRT can be used to speed up decryption by four times.

Speeding up Decryption by CRT

- Multiplying two numbers of k bits takes $O(k^2)$ time.
- $n = pq$. $n \approx$ 1024-2058 bits. $p, q \approx$ half the size.
- Decryption: $c^d \bmod n$.
- Instead of computing $c^d \bmod n$ directly, we
  - compute $c_1 = c \bmod p$ and $c_2 = c \bmod q$
  - compute $m_1 = (c_1)^d \bmod p$ and $m_2 = (c_2)^d \bmod q$
  - recover the plaintext by solving $\begin{cases} x \equiv m_1 \bmod p \\ x \equiv m_2 \bmod q \end{cases}$

Security of RSA

- Four categories of attacks on RSA:
  - brute-force key search
    (infeasible given the large key space)
  - mathematical attacks
  - timing attacks
  - chosen ciphertext attacks

Mathematical Attacks

- Factor n into pq. Then $\varphi(n) = (p-1)(q-1)$ and $d = e^{-1} \bmod \varphi(n)$ can be calculated easily.
- Determine $\varphi(n)$ directly. Equivalent to factoring n. Knowing $\varphi(n)$ will enable us to factor n by solving

$$\begin{cases} n = pq \\ \varphi(n) = (p-1)(q-1) \end{cases}$$

- Determine d directly. Best known algorithms are not faster than those for factoring n. Besides, if d is known, then n can be factored with high probability.

Integer Factorization

- A difficult problem, but more and more efficient algorithms have been developed.
- In 1977, RSA challenged researchers to decode a ciphertex encrypted with a key (n) of 129 digits (428 bits). Prize: \$100. Would take quadrillion years using best algorithms of that time.
- In 1991, RSA put forward more challenges, 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.

Remarks

- In light of current factorization technoligies,
  RSA recommends that n be of 1024-2048 bits.

- Encrypting messages $m \in Z_n \setminus Z_n^*$ is insecure.
  - Such an m is not relatively prime to n, i.e., $\gcd(m,n) > 1$.
  - By computing $\gcd(m,n)$, one will be able to factor $n = pq$.
  - $\gcd(m,n) > 1 \Rightarrow \gcd(c,n) > 1$, where $c = m^e \bmod n$

- Question: what is the probability that $m \in Z_n \setminus Z_n^*$?

Timing Attacks

- Paul Kocher in mid-1990's demonstrated that a snooper can determine a private key by keeping track of how long a computer takes to decipher messages.
- RSA decryption: $c^d \bmod n$.
- Countermeasures:
  - Use constant decryption time
  - Add a random delay to decryption time
  - Blinding: modify the ciphertext c to c' and compute $(c')^d \bmod n$.

Blinding in Some of RSA Products

- RSA encryption has a homomorphism property:

$$\mathrm{RSA}(m_1 \times m_2) = \mathrm{RSA}(m_1) \times \mathrm{RSA}(m_2).$$

- To decrypt a ciphertext $c_m = \mathrm{RSA}(m)$:
  - Generate a random secret message r.
  - Encrypt r as $c_r = \mathrm{RSA}(r)$.
  - Multiply the two ciphertexts: $c_{mr} = c_m c_r = \mathrm{RSA}(mr)$.
  - Decrypting c_{mr} yields a value equal to mr.
  - Multiplying that value by r^{-1} yields m.
- Note: all calculations are done in Z_n^* (i.e., modulo n).

Chosen-Ciphertext Attacks

- RSA's homomorphism property is the basis of a simple chosen-ciphertext attack.
- The attacker intercepts a ciphertext c_m.
- An oracle can decrypt ciphertexts, except c_m, for you.
- To launch a chosen-ciphertext attack:
  - Generate a message, say, $r = 2$.
  - Encrypt r as $c_r = \text{RSA}(r)$.
  - Multiply the two ciphertexts: $c = c_m c_r = \text{RSA}(mr)$.
  - Ask the oracle to decrypt c, yielding a value equal to mr.
  - Multiplying that value by r^{-1} yields the plaintext m.

Small message space attack

- If the message space is small. The adversary can encrypt all messages and compare them with the intercepted ciphertext.
- For instance, if the message is known to be a 56-bit DES key, or a social security number.

Textbook RSA

- The RSA we have described is the basic or textbook RSA, susceptible to many attacks.
- In real world, RSA is not used that way.
- RSA PKCS #1 (now in version 2.1) is a specification by RSA Labs specifying how to implement RSA.

Padded RSA as in PKCS #1 v.1.5

- PKCS: Public Key Cryptography Standard.
- Let (n, e, d) give a pair of RSA keys.
- Let k denote the length of n in bytes (e.g., $k = 216$).
- To encrypt a message m:
  - pad m so that $m' = 00 \| 02 \| r \| 00 \| m$ (k bytes)
  - where $r = 8$ or more random bytes $\neq 00$.
  - original message m must be $\leq k - 11$ bytes.
  - the ciphertext is $c := \mathrm{RSA}(m') = (m')^e \bmod n$.
- This format makes RSA resistant to many of the aforementioned attacks Q: Which ones?

RSA PKCS #1 (v.1.5)

- A padded message is called PKCS conforming if it has the specified format:

  00 □ 02 □ padding string □ 00 □ original message.

- PKCS #1 implementations usually send you (sender) an error message if $\mathrm{RSA}^{-1}(c)$ is not PKCS conforming.
- There was a famous chosen-ciphertext attack taking advantage of these error messages.
- PKCS #1 (v 2.1) now uses a scheme called Optimal Asymmetric Encryption Padding (OAEP).

Bleichenbacher's chosen-ciphertext attack

- A message is called PKCS conforming if it has the specified format:

  00 ▯ 02 ▯ padding string ▯ 00 ▯ original message.

- PKCS #1 implementations usually send you (sender) an error message if $\mathrm{RSA}^{-1}(c)$ is not PKCS conforming.
- It is just like you have an Oracle which, given c, answers whether or not $\mathrm{RSA}^{-1}(c)$ is PKCS conforming.
- Bleichenbacher's attack takes advange of such an Oracle.

- Given $c = \text{RSA}(m)$, Eve tries to find m.
  - (Assume m is PKCS conforming.)
- How can Oracle help?
  - Recall that RSA is homomorphic:

    $\text{RSA}(a \cdot b) = \text{RSA}(a) \cdot \text{RSA}(b)$ (computated in Z_n^*)
  - Given $\text{RSA}(m)$, Eve can compute $\text{RSA}(m \cdot s)$ for any $s \in Z_n^*$.
  - She can ask the Oracle,

    Is $ms \in Z_n^*$ PKCS conforming?

    (That is, is $ms \bmod n$ PKCS conforming?)
- Why is this info useful?

- Recall PKCS Format (k bytes):

  $00 \square 02 \square$ padding string $\square 00 \square$ original message

- Let $B = 00 \square 01 \square 0^{8(k-2)} = 2^{8(k-2)}$ (as a binary integer)
- Then, $2B = 00 \square 02 \square 0^{8(k-2)}$ and $3B = 00 \square 03 \square 0^{8(k-2)}$.
- If m is PKCS conforming $\Rightarrow$ $2B < m < 3B$.
- If, in addition, $ms \bmod n$ is PKCS conforming

  $\Rightarrow$ $2B < ms \bmod n < 3B$

  $\Rightarrow$ $2B + tn < ms < 3B + tn$ for some t

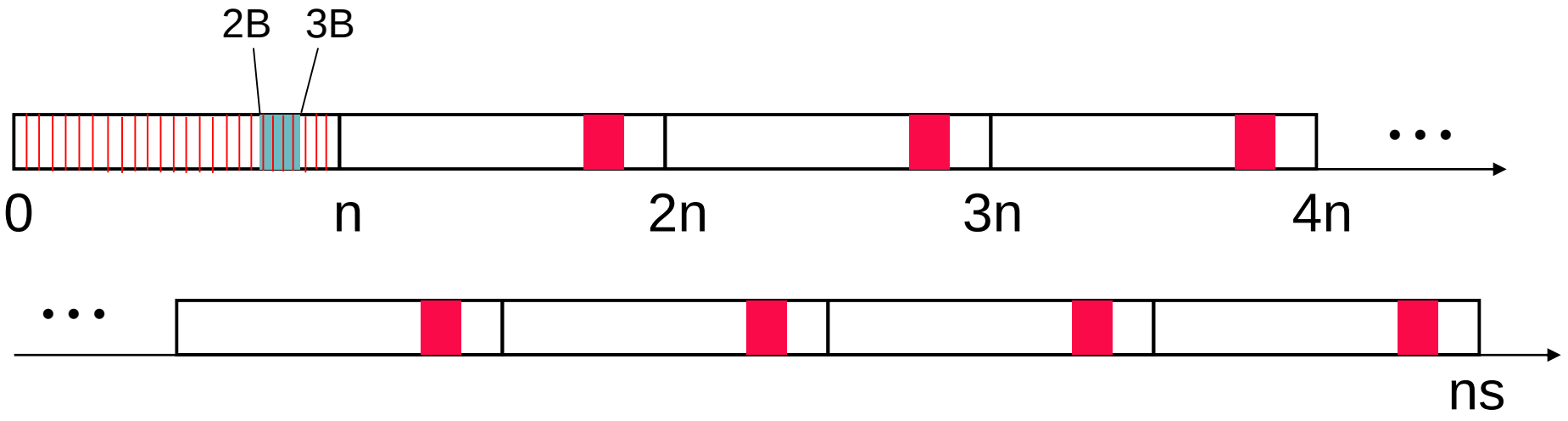
  $\Rightarrow$ $2B/s + t \cdot n/s < m < 3B/s + t \cdot n/s$ for some t

- If m is PKCS conforming $\Rightarrow$ m is in the blue area.
- If $ms \bmod n$ is also PKCS conforming

  $\Rightarrow$ $ms \bmod n$ is in the blue area

  $\Rightarrow$ ms is in the red areas

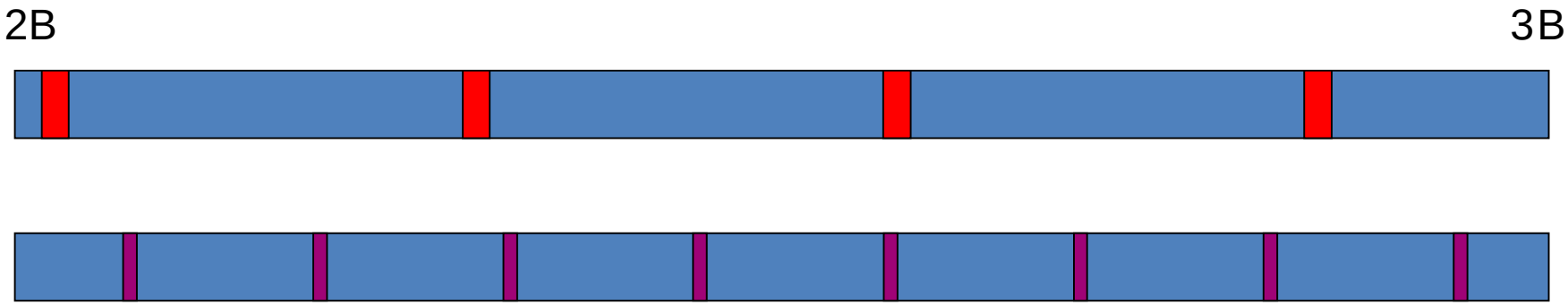
  $\Rightarrow$ m is in the red lines.
- Thus, m is in the red lines of the blue area.



- Let's focus on the blue area, (2B, 3B).
- If m is PKCS conforming $\Rightarrow$ m is in the blue area.
- If $ms \bmod n$ is also PKCS conforming

  $\Rightarrow$ m is in red areas/lines
- If $ms' \bmod n$ is also PKCS conforming

  $\Rightarrow$ m is in purple areas/lines
- So, $m \in \text{blue} \cap \text{red} \cap \text{purple}$



- So, starting with the fact that m is PKCS conforming, Eve finds a sequence of integers $s_1, s_2, s_3, \ldots$ such that

  $2s_{i-1} \le s_i$ and

  $ms_i \bmod n$ is PKCS conforming.
- To find s_i, randomly choose an $s \ge 2s_{i-1}$, and ask the oracle whether $ms \bmod n$ is PKCS conforming. If not, then try a different s.
- This way, Eve can repeatedly narrow down the area containing m, and eventually finds m.
- For n having 1024 bits, it takes roughly 1 million accesses to the oracle in order to find $s_1, s_2, s_3, \ldots$

RSA PKCS #1 (v.2.1)

- The current version of PKCS #1 (v 2.1) uses a scheme called Optimal Asymmetric Encryption Padding (OAEP).
- $$m \xrightarrow{\text{OAEP}} m' := m \oplus G(r) \,\square\, r \oplus h\left(m \oplus G(r)\right)$$
  $$\xrightarrow{\text{RSA}} c := \left(m'\right)^e \bmod n$$
- G : pseudorandom generator
- h : hash function
- r : random

Public-Key Applications

- Three categories of applications:
  - encryption/decryption (provide secrecy)
  - digital signatures (provide authentication)
  - key exchange (of session keys)
- Public-key cryptosystems are slower than symmetric-key systems.
- So, mainly used for digital signatures and key exchange.

- RSA: basic RSA, textbook RSA
- Padded-RSA: PKCS #1 v.1.5
- Original message
- Padded message

.