Public Key Cryptography and the RSA Encryption Algorithm

Dr. Demetrios Glinos
University of Central Florida

CIS3360 - Security in Computing

Readings

- "Computer Security: Principles and Practice", 3rd Edition, by William Stallings and Lawrie Brown
    - Section 21.3

Outline

- Public-Key Cryptography Introduction
- Why Public-Key Cryptography?
- Public Key Encryption/Decryption Process
- Security of Public Key Schemes
- Euler's Totient Function $\phi(n)$
- RSA Key Setup
- RSA Operation
- RSA Example: Key Setup
- RSA Example: Operation
- Modulo Reduction Review

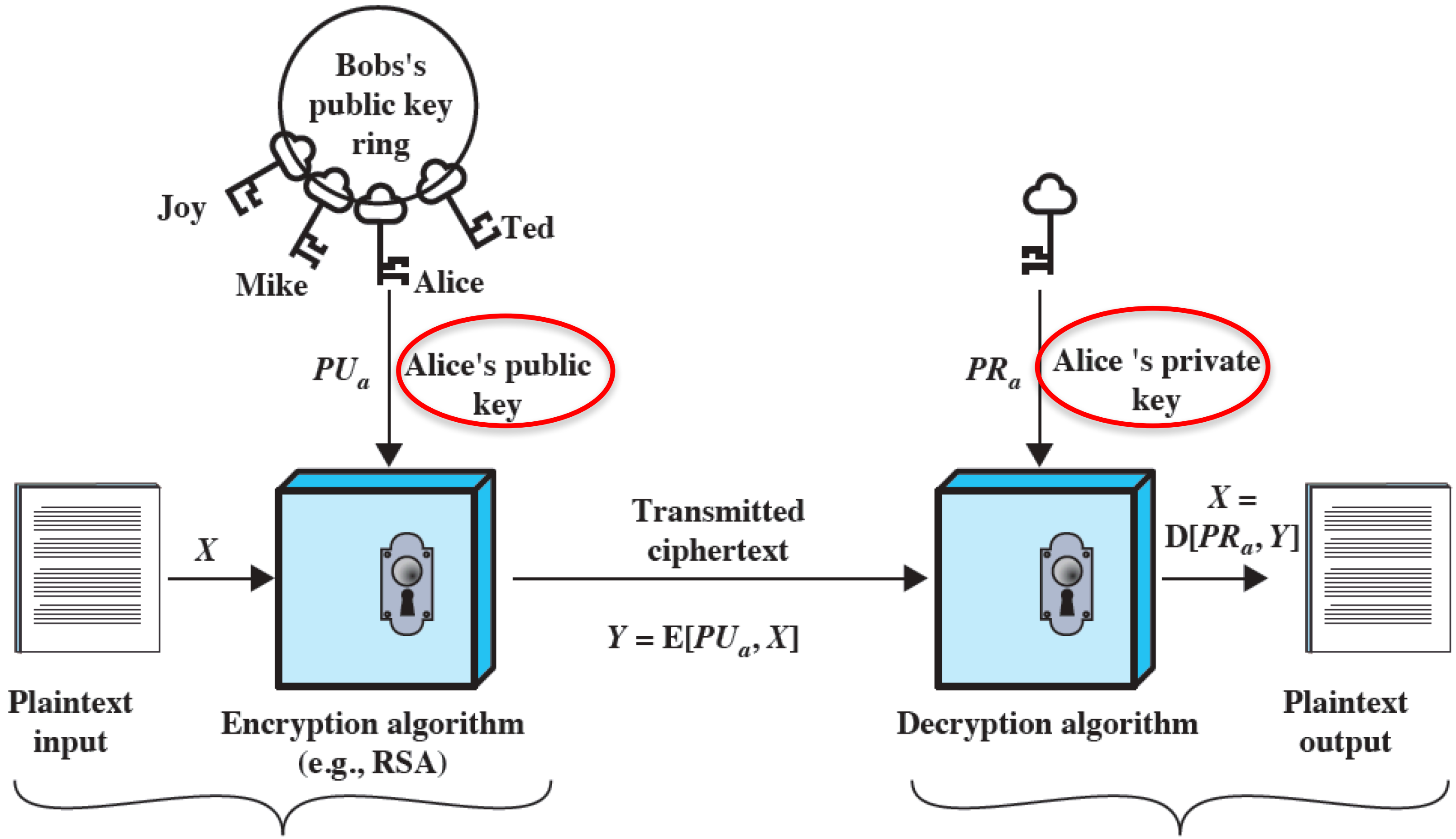
Public-Key Cryptography Introduction

- Probably most significant advance in the 3,000 year history of cryptography
- Uses **two** keys – a **public key** and a **private key**
- It is ***asymmetric*** since sender and receiver use different keys
- Based on a clever application of number theoretic concepts
- In practice, it complements **but does not replace** symmetric cryptographic methods

Why Public-Key Cryptography?

- Developed to address two key issues:
  - **key distribution/exchange** – how to have secure communications in general without having to trust a **Key Distribution Center (KDC)** with your key
  - **digital signatures** – how to verify that a message arrives intact (unmodified) from the claimed sender
- Public invention due to *Whitfield Diffie & Martin Hellman* at Stanford University in 1976
  - known earlier in classified community
  - described in a classified report in 1970 by James Ellis (UK CESG) - and subsequently declassified.
  - There is also a claim that the NSA knew of the concept in the mid-60's.

Public-Key Encryption/Decryption Model



(a) Encryption with public key

source: Fig. 2.6(a), p. 57

Security of Public Key Schemes

- as for symmetric schemes, brute force **exhaustive search** attack is always theoretically possible
- but keys used are usually too large (>512bits)
- *security relies on the difficulty of computing a modular inverse without knowing the modulus*
  - this task is impractical **if** large numbers are used
- since security requires the use of **very large numbers** and **complex calculations** are involved, public key methods are ***slow*** compared to private (symmetric) key schemes
  - this is why general practice is to use public key methods for negotiating keys and for digital signatures, but use symmetric methods for data communication

Euler's Totient Function $\phi(n)$ (part 1)

- *Euler's Totient, which we denote by* ***ϕ (n)*** is defined as the *number* of *positive integers less than n (or 1 if n=1)* that are *relatively prime to n*
  - $\phi(n)$ is a ***count*** of a set of numbers, i.e., how many of them there are
  - *1 is always in the set; 0 is never in the set*
- Notation:
  - we use braces to enclose the elements of a set, e.g., { 2, 5, 13 }
  - we use vertical bars around a set denote its count, e.g., |{2, 5, 13}|
- Some totient values:

$$\phi(1) = 1 = |\{1\}|$$
$$\phi(2) = 1 = |\{1\}|$$
$$\phi(3) = 2 = |\{1, 2\}|$$
$$\phi(4) = 2 = |\{1, 3\}|$$
$$\phi(7) = 6 = |\{1, 2, 3, 4, 5, 6\}|$$
$$\phi(10) = 4 = |\{1, 3, 7, 9\}|$$

Euler’s Totient Function $\phi(n)$ (part 2)

- If p is a *prime*, then $\phi(p) = p - 1$
- If p and q are *both primes*, and $p \neq q$, then $\phi(pq) = \phi(p)\phi(q)$
  - In other words, the totient of a product of two primes is the product of their totients
- Example 1: $p = 2, q = 5$; $pq = 10$ ← p and q are both prime
  $\phi(10) = \phi(2)\phi(5) = (1)(4) = 4$ ← result is same as before
- Example 2: $p = 5, q = 7$; $pq = 35$
  $\phi(35) = \phi(5)\phi(7) = (4)(6) = 24$ ← we don’t need to find them to know how many there are!

Euler's Totient Function $\phi(n)$ (part 3)

- If p and q are *relatively prime*, then $\phi(pq) = \phi(p)\phi(q)$
  - i.e., the result for primes also works for relatively prime numbers

- Example: $\phi(110) = \phi(11)\,\phi(10)$
  $= (10)\,(4) = 40$

  Note that $\phi(11) = 10$ since 11 is prime

  We also know $\phi(10) = \phi(2)\phi(5) = (1)(4) = 4$ from before

RSA Algorithm Key Setup

- Developed by Ron **R**ivest, Adi **S**hamir, and Len **A**dleman, who won the Turing award in 2002 for this
- Select two large prime numbers at random: **p, q**
- Compute **$n = pq$** and **$\phi(n) = (p-1)(q-1)$**
- **Select** a random *encryption key* **e** that satisfies both of these conditions:

  $$1 < e < \phi(n)$$

  $\gcd(e, \phi(n)) = 1$ ← *e and $\phi(n)$ are relatively prime*

- **Find** *decryption key* **d** in the range **$0 \le d \le \phi(n)$** using the Extended Euclidean Algorithm, satisfying the condition:

  $ed = 1 \bmod \phi(n)$ ← *e and d are inverses modulo $\phi(n)$*

- Publish **public encryption key**: **PU = { e, n }**
- Keep secret **private decryption key**: **PR = { d, n }**

RSA Operation

- The message is divided into blocks (pad the last one, if necessary), which are interpreted as unsigned binary numbers
  - the blocks must be sized so that the largest numeric value is less than the modulus **n** (which is part of both keys)
  - i.e., For all message blocks M, we have $0 \leq M < n$

  - Alice **encrypts** a single message block **M** and sends it to Bob:
  - she uses **Bob's public key** **PU_B={ e, n }**
  - Alice encrypts by computing: **$C = M^e \bmod n$**

- Bob **decrypts** the ciphertext C for the block:
  - Bob uses **Bob's private key** **PR_B ={ d, n }**
  - Bob decrypts by computing: **$M = C^d \bmod n$**
  - NOTE that C was already blocked at the encryption end

RSA Example - Key Setup

1. Alice selects primes, for example: ***p* = 17** and ***q* = 11**

1. Alice computes ***n* = *pq* = (17)(11) = 187**

2. Alice computes ***ϕ(n)* = (*p*–1)(*q*-1) = (16)(10) = 160**

3. Alice selects **e** so that **gcd(e,160) = 1**; suppose she chooses ***e* = 7**

4. Alice uses Extended Euclidean Algorithm to find **d** satisfying ***de* = 1 mod 160** and ***d* < 160**

*(We computed this value previously; it is **d = 23** since (23)(7) mod 160 = 161 mod 160 = 1)*

6. Alice publishes ***her public key*** **PU = { e, n } = { 7, 187 }**

   and keeps secret ***her private key*** **PR = { d, n } = { 23, 187 }**

RSA Example - Operation

- Encryption/decryption for the example on the previous slide

- Assume Alice has generated these public and private RSA keys, and has published the public key:

PU = { 7, 187 }

PR = { 23, 187 }

- And suppose Bob wishes to encrypt the message M = 88 (which is less than 187) and send it to Alice.

  - Bob encrypts using Alice's public key: $C = 88^{7} \bmod 187 = 11$

  - Alice decrypts using her private key: $M = 11^{23} \bmod 187 = 88$

Modulo Reduction Review

- To perform RSA encryption and decryption by hand using our calculators, we use **modulo reduction**
- Suppose we wish to compute: **$7^5 \bmod 11$**
- We expand and simplify repeatedly until we have the desired result:

$$
\begin{aligned}
7^5 \bmod 11 &= (\, 7^4 \bmod 11 \,)(\, 7^1 \bmod 11 \,) \bmod 11 \\
&= (\, 7^2 \bmod 11 \,)(\, 7^2 \bmod 11 \,)\ (\, 7 \bmod 11 \,) \bmod 11 \\
&= (\, 49 \bmod 11)\ (\, 49 \bmod 11)\ (\, 7 \bmod 11 \,) \bmod 11 \\
&= (\, 5 \bmod 11 \,)(\, 5 \bmod 11 \,)(\, 7 \bmod 11 \,) \bmod 11 \\
&= (\, 25 \bmod 11 \,)\ (\, 7 \bmod 11 \,) \bmod 11 \\
&= (\, 3 \bmod 11 \,)(\, 7 \bmod 11 \,) \bmod 11 \\
&= 21 \bmod 11 \\
&= 10
\end{aligned}
$$