Public-Key (Asymmetric) Ciphers

ENCS4320 - Applied Cryptography

Dr. Ahmed I. A. Shawahna

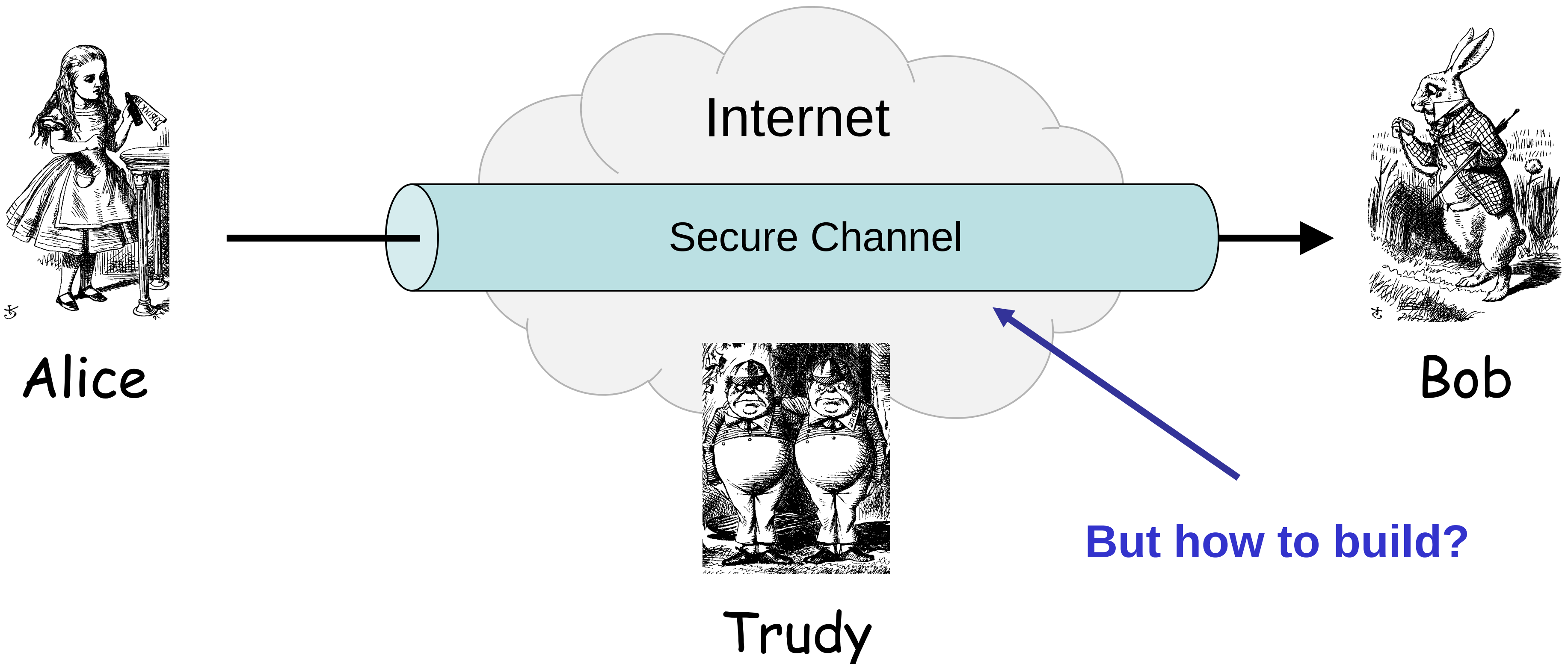
Electrical and Computer Engineering Department

Birzeit University

Presentation Outline

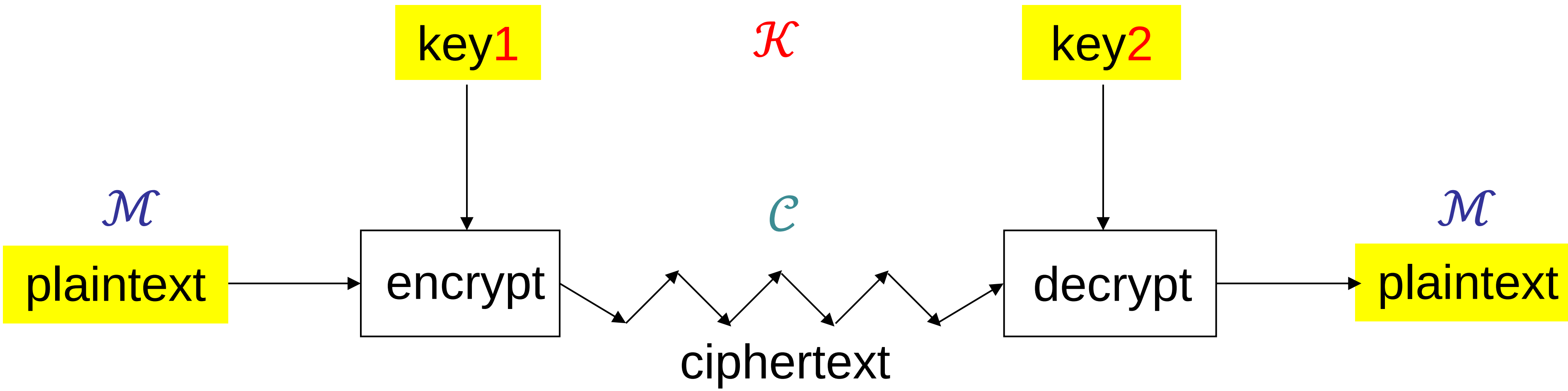
- **Motivation**
- The Public-Key Revolution
- Principles Behind Public-Key Ciphers
- Essential Number Theory for Public-Key Algorithms
- Essential Group Theory for Public-Key Algorithms
- Public-key Examples

Basic Goals of Cryptography: Review



- **Security goals:**
  - Data privacy: adversary should not be able to **read** message *M*
  - Data integrity: adversary should not be able to **modify** message *M*
  - Data authenticity: message *M* really **originated** from Alice

Cryptographic Schemes

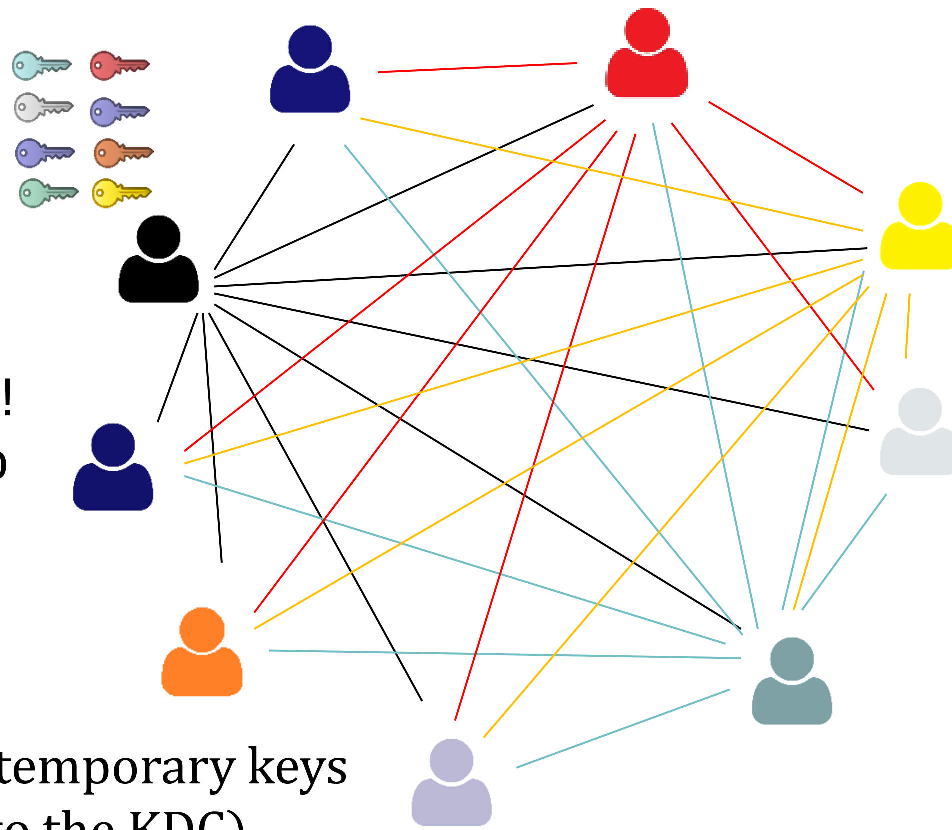


Crypto	Keys
Symmetric Key	key1 = key2
Public Key	key1 ≠ key2

How do keys get distributed?

Issues with Symmetric-key Ciphers

- Symmetric-key ciphers (e.g., AES or 3DES) are very secure, fast, and widespread. **But**:
  - Number of keys: In a network, each pair of users requires a unique key
    - N users in the network require $N(N-1)/2$ keys, $\mathcal{O}(N^2)$, with each user storing $(N-1)$ keys!!
    - Difficult to store and manage so many keys securely



- Partial solution:
  - key distribution centers (**KDC**)
    - One central authority hands out temporary keys
    - $\mathcal{O}$(N) (long-term) keys needed (to the KDC)
    - Might be a feasible solution in a single organization
    - But, single point of failure, and
    - What about the internet?

Issues with Symmetric-key Ciphers

- Symmetric-key ciphers (e.g., AES or 3DES) are very secure, fast, and widespread. **But**:
  - Key distribution problem: Secret key must be transported securely
  - Cheating: Alice or Bob can cheat each other, because they have identical keys!!!
    - Repudiation by Alice
    - Fabrication by Bob

Next …

- Motivation
- **The Public-Key Revolution**
- Principles Behind Public-Key Ciphers
- Essential Number Theory for Public-Key Algorithms
- Essential Group Theory for Public-Key Algorithms
- Public-key Examples

Basic Goals of Cryptography

	Message Privacy	Message Integrity / Authentication
Symmetric Keys	Symmetric Encryption (private-key encryption)	Message Authentication Codes (MAC)
Asymmetric Keys	Asymmetric Encryption (public-key encryption)	Digital Signatures

(Key exchange)

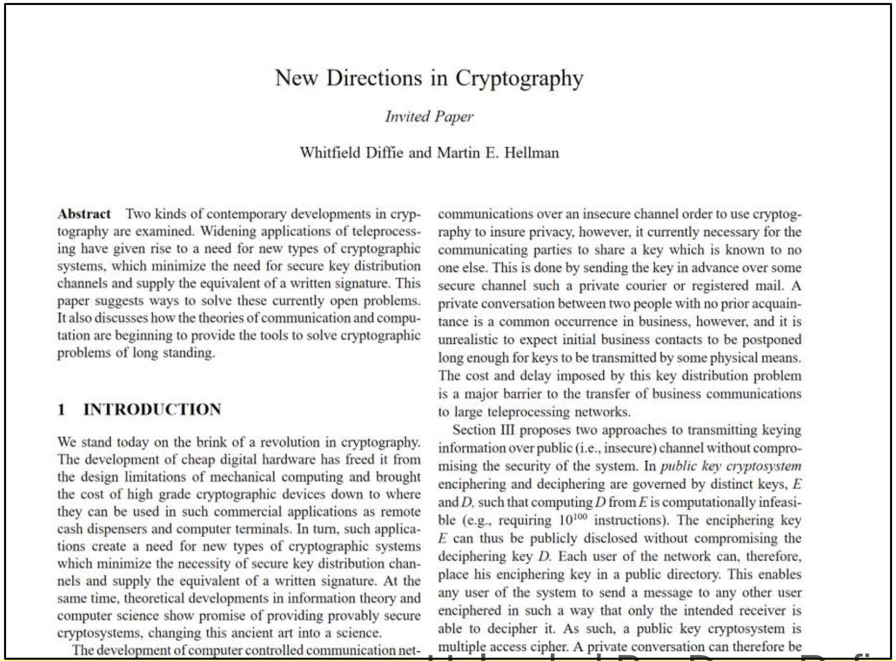
Diffie-Hellman Key Exchange – Idea

Public-Key Encryption

Diffie-Hellman Key Exchange

- Discovered in the 1970's
- A "key exchange" algorithm
  - Allows two parties to establish a shared secret (shared symmetric key) without ever having met
  - **Not** for encrypting or signing
- Diffie & Hellman paper also introduced:
  - Public-key encryption
  - Digital signatures

Ralph Merkle Whitfield Diffie
Martin Hellman



New Directions in Cryptography

Invited Paper

Whitfield Diffie and Martin E. Hellman

Abstract Two kinds of contemporary developments in cryptography are examined. Widening applications of teleprocessing have given rise to a need for new types of cryptographic systems, which minimize the need for secure key distribution channels and supply the equivalent of a written signature. This paper suggests ways to solve these currently open problems. It also discusses how the theories of communication and computation are beginning to provide the tools to solve cryptographic problems of long standing.

1 INTRODUCTION

We stand today on the brink of a revolution in cryptography. The development of cheap digital hardware has freed it from the design limitations of mechanical computing and brought the cost of high grade cryptographic devices down to where they can be used in such commercial applications as remote cash dispensers and computer terminals. In turn, such applications create a need for new types of cryptographic systems which minimize the necessity of secure key distribution channels and supply the equivalent of a written signature. At the same time, theoretical developments in information theory and computer science show promise of providing provably secure cryptosystems, changing this ancient art into a science.

The development of computer controlled communication networks over an insecure channel order to use cryptography to insure privacy, however, it currently necessary for the communicating parties to share a key which is known to no one else. This is done by sending the key in advance over some secure channel such a private courier or registered mail. A private conversation between two people with no prior acquaintance is a common occurrence in business, however, and it is unrealistic to expect initial business contacts to be postponed long enough for keys to be transmitted by some physical means. The cost and delay imposed by this key distribution problem is a major barrier to the transfer of business communications to large teleprocessing networks.

Section III proposes two approaches to transmitting keying information over public (i.e., insecure) channel without compromising the security of the system. In *public key cryptosystem* enciphering and deciphering are governed by distinct keys, E and D, such that computing D from E is computationally infeasible (e.g., requiring 10^{100} instructions). The enciphering key E can thus be publicly disclosed without compromising the deciphering key D. Each user of the network can, therefore, place his enciphering key in a public directory. This enables any user of the system to send a message to any other user enciphered in such a way that only the intended receiver is able to decipher it. As such, a public key cryptosystem is multiple access cipher. A private conversation can therefore be

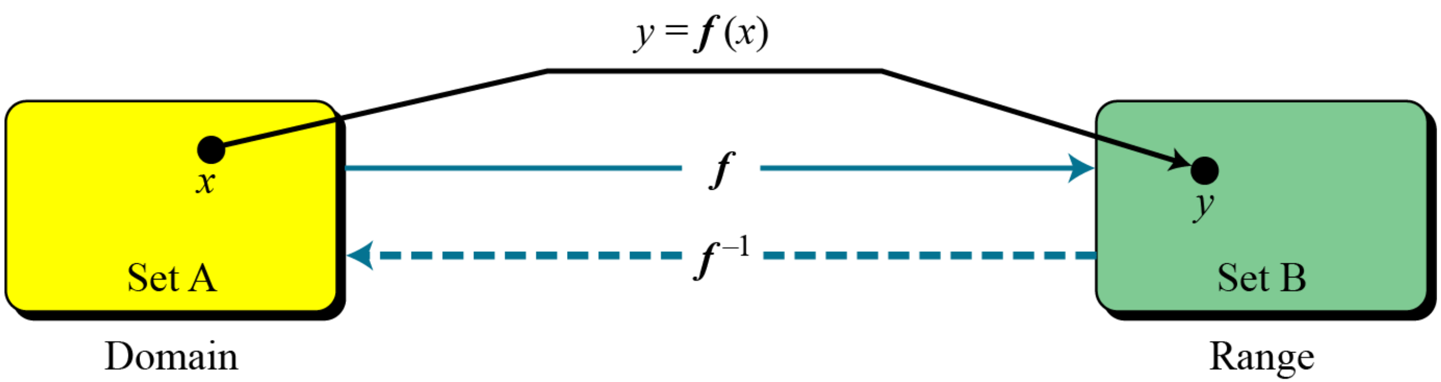
Next …

- ❖ Motivation
- ❖ The Public-Key Revolution
- ❖ **Principles Behind Public-Key Ciphers**
- ❖ Essential Number Theory for Public-Key Algorithms
- ❖ Essential Group Theory for Public-Key Algorithms
- ❖ Public-key Examples

Public-Key Cryptography

- Two keys
  - Sender uses recipient's **public key** to encrypt
  - Recipient uses **private key** to decrypt
- The main idea behind asymmetric-key cryptography is the concept of the trapdoor one-way function
  - "One way" means easy to compute in one direction, but hard to compute in other direction
  - "Trap door" used to create key pairs

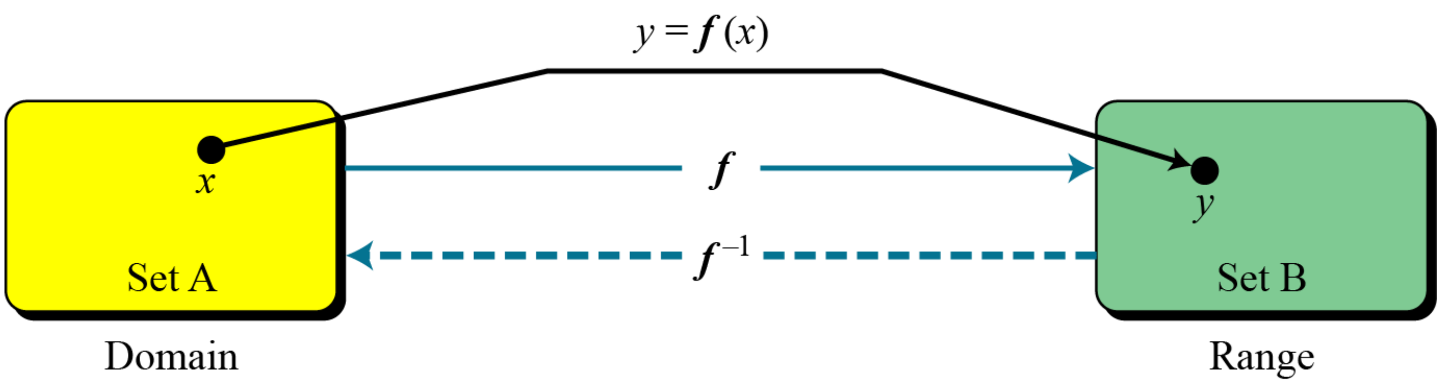
A function is a rule mapping a domain to a range



Trapdoor One-Way Function

- One-Way Function (OWF)
  1. f is easy to compute
  2. f^{-1} is difficult to compute
- Trapdoor One-Way Function (TOWF)
  3. Given y and a trapdoor k', x can be computed easily

A function is a rule mapping a domain to a range

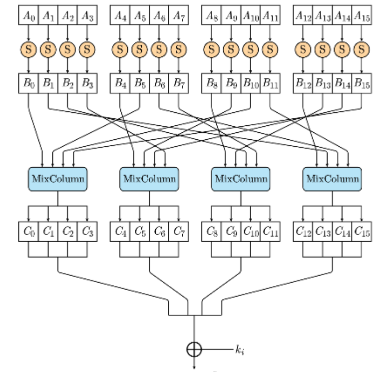


Trapdoor One-Way Function (Continued)

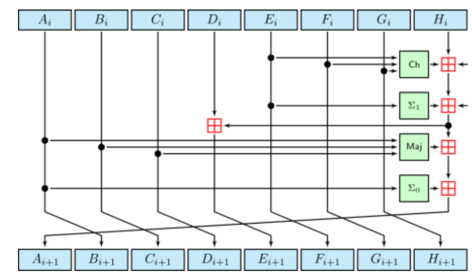
- Example 1:
  - When n is large, $n = p \times q$ is a one-way function
  - Given p and q, it is always easy to calculate n
  - However, given n, it is very difficult to compute p and q
  - This is the factorization problem
- Example 2:
  - When n is large, the function $y = x^k \bmod n$ is a trapdoor one-way function
  - Given x, k, and n, it is easy to calculate y
  - Given y, k, and n, it is very difficult to calculate x
  - This is the discrete logarithm problem
  - However, if we know the trapdoor, k', such that $k \times k' = 1 \bmod \phi(n)$, we can use $x = y^{k'} \bmod n$ to easily find x

Symmetric and Asymmetric Crypto

- Symmetric crypto boils down to a few *primitives*
  - Block ciphers/PRFs, hash functions
  - Why are these considered secure?
    - Lots and lots of cryptanalysis (*well-studied!*)
    - Artificial and man-made
- Want asymmetric crypto to be based on a few well-studied primitives too
- Candidates come from a different place:
  - Hard *mathematical* problems
  - Good candidates: discrete logarithm problem, factoring
    - Much more algebraic structure



AES

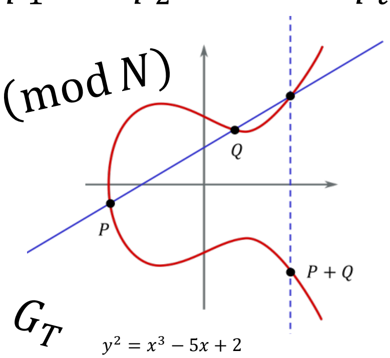


SHA2-256

$$\mathbf{Z}_n^* \simeq \mathbf{Z}_{p_1}^* \times \mathbf{Z}_{p_2}^* \times \cdots \times \mathbf{Z}_{p_t}^*$$

$$C \leftarrow M^e \pmod N$$

$$e : G_1 \times G_2 \to G_T$$



How to build Public-Key Algorithms

- One-way functions are based on **mathematically hard problems**
- Three main families:
  - **Factoring integers** (RSA, ...)
    - Given a composite integer *n*, find its prime factors
    - Multiply two primes: easy
  - **Discrete Logarithm (DL)** (Diffie-Hellman, Elgamal, DSA, …)
    - Given a, y and m, find x such that $y = a^x \bmod m$
    - Exponentiation a^x : easy
  - **Elliptic Curves (EC)** (ECDH, ECDSA)
    - Generalization of discrete logarithm
- **Note:** The problems are considered mathematically hard, but no proof exists (so far)

Key Lengths and Security Levels

Symmetric	ECC	RSA, DL	Remark
64 Bit	128 Bit	≈ 700 Bit	Only short term security (a few hours or days)
80 Bit	160 Bit	≈ 1024 Bit	Medium security (except attacks from big governmental institutions etc.)
128 Bit	256 Bit	≈ 3072 Bit	Long term security (without quantum computers)

- The exact complexity of RSA and DL is difficult to estimate
- The existence of quantum computers would probably be the end for EC, RSA ,& DL (at least 2-3 decades away, and some people doubt that QC will ever exist)

Public-Key Cryptography

- **Encryption**
  - Suppose we **encrypt** M with Bob's public key
  - Bob's private key can **decrypt** to recover M
- **Digital Signature**
  - **Sign** by "encrypting" with your private key
  - Anyone can **verify** signature by "decrypting" with public key
  - But only you could have signed
  - Like a handwritten signature, but way better…

Next …

- ❖ Motivation
- ❖ The Public-Key Revolution
- ❖ Principles Behind Public-Key Ciphers
- ❖ **Essential Number Theory for Public-Key Algorithms**
- ❖ Essential Group Theory for Public-Key Algorithms
- ❖ Public-key Examples

Modular Exponentiation

- A type of exponentiation performed over a modulus
  - e.g., $a^b \bmod m$ or $a^b \pmod m$

- Example: Solve $23^3 \bmod 30$
  - $23 \equiv -7 \bmod 30$
  - $-7^3 = -343 = -13 \equiv 17 \bmod 30$

  ➔ $23^3 \bmod 30 \equiv 17$

- Example: Solve $31^{500} \bmod 30$
  - $31 \equiv 1 \bmod 30$
  - $1^{500} = 1 \bmod 30$

  ➔ $31^{500} \bmod 30 \equiv 1$

Modular Exponentiation

- Example: Solve $242^{329} \bmod 243$
  - $242^{329} \equiv -1^{329} \bmod 30$

  - $-1^{329} = -1 \equiv 242 \bmod 243$

  ➔ $242^{329} \bmod 243 \equiv 242$

- Example: Solve $11^{7} \bmod 13$
  - $11 \equiv -2 \bmod 13$

  - $-2^{7} = -128 \equiv -11 \bmod 13 \equiv 2 \bmod 13$

  ➔ $11^{7} \bmod 13 \equiv 2$

Modular Exponentiation

- Example: Solve 5^{20} *mod* 35
  - $5^{20} = 95367431640625 \equiv 25 \bmod 35$
- A better way: **Square-and-Multiply Algorithm**
  - $20 = (10100)_2$
  - $(1, 10, 101, 1010, 10100) = (1, 2, 5, 10, 20)$
  - Note that $2 = 1 \cdot 2,\ 5 = 2 \cdot 2 + 1,\ 10 = 2 \cdot 5,\ 20 = 2 \cdot 10$
  - $5^1 \equiv 5 \bmod 35$
  - $5^2 = (5^1)^2 = 5^2 \equiv 25 \bmod 35$
  - $5^5 = (5^2)^2 \cdot 5^1 = 25^2 \cdot 5 = 3125 \equiv 10 \bmod 35$
  - $5^{10} = (5^5)^2 = 10^2 = 100 \equiv 30 \bmod 35$
  - $5^{20} = (5^{10})^2 = 30^2 = 900 \equiv 25 \bmod 35$
- No huge numbers and it's efficient!

Modular Exponentiation

- Example: Solve $88^7 \bmod 187$
  - $7 = (111)_2$
  - $(1, 11, 111) = (1, 3, 7)$
  - Note that $3 = 1 \cdot 2 + 1$, $7 = 2 \cdot 3 + 1$
  - $88^1 \equiv 88 \bmod 187$
  - $88^3 = (88^1)^2 \cdot 88^1 = 88^2 \cdot 88 = 7744 \cdot 88 \equiv 77 \cdot 88 = 6776 \equiv 44 \bmod 187$
  - $88^7 = (88^3)^2 \cdot 88^1 = 44^2 \cdot 88 = 1936 . 88 \equiv 66 . 88 = 5808 \equiv 11 \bmod 187$

Euler's Phi (Totient) Function

- New problem, important for public-key systems, e.g., RSA
- Given the set of the *m* integers *{0, 1, 2, ..., m-1}*
- **How many** numbers in the set are **relatively prime to *m***?
- Answer: **Euler's Phi (Totient) Function $\phi(m)$**
  - $\phi(m)$ is "*the number of numbers less than m that are relatively prime to m*". Here, "numbers" are positive integers
- Example: Calculate $\phi(5)$ and $\phi(6)$

$$
\begin{aligned}
\gcd(0,6) &= 6\\
\gcd(1,6) &= 1 \Leftarrow\\
\gcd(2,6) &= 2\\
\gcd(3,6) &= 3\\
\gcd(4,6) &= 2\\
\gcd(5,6) &= 1 \Leftarrow
\end{aligned}
$$

$$
\begin{aligned}
\gcd(0,5) &= 5\\
\gcd(1,5) &= 1 \Leftarrow\\
\gcd(2,5) &= 1 \Leftarrow\\
\gcd(3,5) &= 1 \Leftarrow\\
\gcd(4,5) &= 1 \Leftarrow
\end{aligned}
$$

→ 1 and 5 relatively prime to *m*=6,

hence **$\Phi(6) = 2$**

→ **$\Phi(5) = 4$**

Euler's Phi (Totient) Function

- Testing one *gcd* per number in the set is extremely slow for large *m*
- Fortunately, there exists a relation to calculate $\phi(m)$ much more easily if we know the factorization of *m*

Theorem 6.3.1 *Let m have the following canonical factorization*

$$m = p_1^{e_1} \cdot p_2^{e_2} \cdot \ldots \cdot p_n^{e_n},$$

where the p_i are distinct prime numbers and e_i are positive integers, then

$$\Phi(m) = \prod_{i=1}^{n} (p_i^{e_i} - p_i^{e_i - 1}).$$

Euler's Phi (Totient) Function

- Phi especially easy for $e_i = 1$
  - e.g., $m = p \,.\, q \rightarrow \phi(m) = (p-1) \,.\, (q-1)$
- Example: $m = 899 = 29 \,.\, 31$
  - $\phi(899) = (29 - 1) \,.\, (31 - 1) = 28 \,.\, 30 = 840$
- **Note:** Finding $\phi(m)$ is computationally easy **if factorization of *m* is known**
  - Otherwise, the calculation of becomes computationally infeasible for large numbers
- $\phi(n)$ can be computed recursively by:
  1. $\phi(1) = 1$
  2. if n is a prime power, $n = p^e$, then $\phi(n) = p^e - p^{(e-1)}$
  3. if $\gcd(m, n) = 1$, then $\phi(mn) = \phi(m) \,.\, \phi(n)$

Euler's Phi (Totient) Function

- Examples:
  - $\phi(17) = 16$
  - $\phi(25) = 5^2 = 5^2 - 5 = 20$
  - $\phi(16) = 2^4 = 2^4 - 2^3 = 8$
  - $\phi(105) = \phi(3 \,.\, 5 \,.\, 7) = 2 \,.\, 4 \,.\, 6 = 48$
  - $\phi(200) = \phi(2^3 \,.\, 5^2) = (2^3 - 2^2)\ (5^2 - 5) = 4 \,.\, 20 = 80$
  - $\phi(240) = \phi(2^4 \,.\, 3 \,.\, 5) = (2^4 - 2^3) \,.\, (3 - 1) \,.\, (5 - 1) = 8 \,.\, 2 \,.\, 4 = 64$

Fermat's Little Theorem

- Very useful in **public-key ciphers**
  - e.g., primality testing while generating keys

> **Theorem 6.3.2** Fermat's Little Theorem
> *Let a be an integer and p be a prime, then:*
>
> $$a^p \equiv a \pmod{p}.$$

- Recall that arithmetic in finite fields $GF(p)$ is done modulo p
  - Hence, the theorem holds for all integers $a \in GF(p)$
- **Alternate form:** $a^p \times a^{-1} = a^{p-1} \equiv 1 \pmod{p}$ (why?)
  - $a \times a^{p-2} \equiv 1 \pmod{p} \Rightarrow$ **$a^{-1} \equiv a^{p-2} \pmod{p}$**
  - Quick way to find multiplicative inverse if modulus is a prime!!
  - But slower than EEA unless a hardware accelerator is used for fast exponentiation
- **Example:** Let $p = 7$ and $a = 2 \Rightarrow a^{-1} = a^{p-2} = 2^5 = 32 \equiv 4 \bmod 7$

Euler's Theorem

- Generalization of Fermat's Little Theorem for any modulus
  - i.e., moduli that are not necessarily primes

> **Theorem 6.3.3** Euler's Theorem
> *Let a and m be integers with* $\gcd(a,m) = 1$, *then:*
>
> $$a^{\Phi(m)} \equiv 1 \pmod{m}.$$

- **Example:** Let $m = 12$ and $a = 5 \Rightarrow gcd\ (12, 5) = 1$

  $\Rightarrow \phi(12) = \phi(2^2 \cdot 3) = (2^2 - 2^1)(3^1 - 3^0) = 4$

  Verify: $5^{\phi(12)} = 5^4 = 625 \equiv 1 \bmod 12$

- Theorem is used to prove correctness of RSA (most popular public-key crypto)

Euler's Theorem

- Another important use of Euler's theorem is the following:

$$\boxed{\text{if } k \equiv j \pmod{\phi(n)}\ ,\ \text{then } a^k \equiv a^j \bmod n}$$

  - Helps in exponentiation computation (needed in RSA)

- **Example 1:** $2^{46} = 2^2 \pmod 5$, since $46 \equiv 2 \pmod{\phi(5)}$

- **Example 2:** Compute the following:

  a. $14^{52} \pmod{11}$

  $\Rightarrow 14^{52} \equiv 3^{52} \pmod{11}$. Since $\phi(11) = 10 \Rightarrow 52 \equiv 2 \pmod{10}$

  $\Rightarrow 14^{52} \equiv 3^{52} \equiv 3^2 \equiv 9 \pmod{11}$

  b. $463^{91} \pmod{15}$

  $\Rightarrow 463^{91} \equiv 13^{91} \equiv (-2)^{91} \pmod{15}$. Since $\phi(15) = 2 \times 4 = 8$

  $\Rightarrow 91 \equiv 3 \pmod 8 \Rightarrow (-2)^{91} \equiv (-2)^3 \equiv -8 \equiv 7 \pmod{15}$

Next …

- ❖ Motivation
- ❖ The Public-Key Revolution
- ❖ Principles Behind Public-Key Ciphers
- ❖ Essential Number Theory for Public-Key Algorithms
- ❖ **Essential Group Theory for Public-Key Algorithms**
- ❖ Public-key Examples

Preliminaries

(integers) $\boldsymbol{Z} = \{\dots, -2, -1, 0, 1, 2, 3, \dots\}$

(reals) $\boldsymbol{R} =$ the real numbers $\boldsymbol{R}^* = \boldsymbol{R} \setminus \{0\}$

(integers "mod n") $\boldsymbol{Z}_n = \{0, 1, 2, \dots, n-1\}$

(integers "mod p") $\boldsymbol{Z}_p = \{0, 1, 2, \dots, p-1\}$ p prime $\boldsymbol{Z}_p^* = \boldsymbol{Z}_p \setminus \{0\}$

Examples:

An integer $p > 1$ is **prime** if it's only divisible by 1 and p

$$\boldsymbol{Z}_{11} = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$$

$$\boldsymbol{Z}_{11}^* = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$$

Group - Definition

Definition: A **group** $(G,\circ)$ is a set G together with a binary operation $\circ$ satisfying the following axioms.

$\mathcal{G}1$: $(a \circ b) \circ c = a \circ (b \circ c)$ for all $a, b, c, \in G$ (associativity)

$\mathcal{G}2$: $\exists e \in G$ such that $e \circ a = a \circ e = a$ for all $a \in G$ (identity)

$\mathcal{G}3$: $\forall a \in G$ there exists $a^{-1} \in G$ such that $a \circ a^{-1} = a^{-1} \circ a = e$ (inverse)

A group is **abelian/commutative** if: $a \circ b = b \circ a$ for all $a, b \in G$

The **order** of a group is the number of elements in G, denoted $|G|$

Groups - Examples

Groups

$(\boldsymbol{Z},+)$ $\quad e = 0 \quad$ "3^{-1}" $= -3$

$(\boldsymbol{R},+)$ $\quad e = 0 \quad$ "$(9/7)^{-1}$" $= -9/7$

$(\boldsymbol{R}^*,\cdot)$ $\quad e = 1 \quad (9/7)^{-1} = 7/9$

$(\boldsymbol{Z}_n,+_n)$ $\quad e = 0 \quad$ "3^{-1}" $= x$: $\ 3 + x \equiv 0 \bmod n$

$(\boldsymbol{Z}_p^*,\ \cdot_p)$ $\quad e = 1 \quad$ "3^{-1}" $= x$: $\ 3 \cdot x \equiv 1 \bmod p$

$(\boldsymbol{G},\circ)$

$\circ$	e	a	b
e	e	a	b
a	a	b	e
b	b	e	a

$(\boldsymbol{Z}_3,+_3)$

$+_3$	0	1	2
0	0	1	2
1	1	2	0
2	2	0	1

$(\boldsymbol{G},\circ)$

$\circ$	e	a	b	c
e	e	a	b	c
a	a	b	c	e
b	b	c	e	a
c	c	e	a	b

Not Groups

$(\boldsymbol{Z},\cdot)$ $\quad 2^{-1} = ?$

$(\boldsymbol{Z},-)$ $\quad (1-2)-3 \neq 1-(2-3)$

$(\boldsymbol{R},\cdot)$ $\quad 0 \cdot x = 1?$

$(\boldsymbol{Z}_n,\cdot_n)$ $\quad 2x = 1 \pmod 4$?

$(\boldsymbol{Z}_p,\ \cdot_p)$

$(\boldsymbol{Z}_4,+_4)$

$+_4$	0	1	2	3
0	0	1	2	3
1	1	2	3	0
2	2	3	0	1
3	3	0	1	2

$(\boldsymbol{G},\star)$

$\star$	e	a	b	c
e	e	a	b	c
a	a	e	c	b
b	b	c	e	a
c	c	b	a	e

Group Arithmetic

$$g^0 \stackrel{\text{def}}{=} e$$

$$g^n \stackrel{\text{def}}{=} \overbrace{g \circ g \circ \cdots \circ g}^{n}$$

$$g^{-n} \stackrel{\text{def}}{=} (g^{-1})^n$$

$(\boldsymbol{Z}, +)$: $\text{"}2^{10}\text{"} = \overbrace{2 + 2 + \cdots + 2}^{10}$
$= 10 \cdot 2 = 20$

Fact: $g^n \circ g^m = g^n g^m = \underbrace{\overbrace{g \circ \cdots \circ g}^{n} \circ \overbrace{g \circ \cdots \circ g}^{m}}_{n+m} = g^{n+m}$

Fact: $(g^n)^m = g^{nm} = (g^m)^n$

Cyclic Groups

Definition: A **group** $(G,\circ)$ is **cyclic** if there exists $g \in G$ such that

$$G = \{ g^i \mid i \in \boldsymbol{Z} \} = \{\dots, g^{-2}, g^{-1}, g^0, g^1, g^2, \dots\}$$

Element g is called a **generator** for G and we write $(G,\circ) = \langle g \rangle$

Examples:

$(\boldsymbol{Z}, +) = \langle 1 \rangle$

$(\boldsymbol{Z}_n, +_n) = \langle 1 \rangle$

$(\boldsymbol{Z}_p^*, \cdot) = \langle a \rangle$

$$\begin{aligned}(\boldsymbol{Z}_7^*, \cdot) &= \langle 3 \rangle = \{3^0, 3^1, 3^2, 3^3, 3^4, 3^5\} = \{1, 3, 2, 6, 4, 5\}\\ &= \langle 5 \rangle = \{5^0, 5^1, 5^2, 5^3, 5^4, 5^5\} = \{1, 5, 4, 6, 2, 3\}\\ &\neq \langle 2 \rangle = \{2^0, 2^1, 2^2, 2^3, 2^4, 2^5\} = \{1, 2, 4, 1, 2, 4\} = \{1, 2, 4\}\end{aligned}$$

Not cyclic groups:

$(\boldsymbol{R}, +)$ $(\boldsymbol{R}^*, \cdot)$

Cyclic Groups

Definition: A **group** $(G,\circ)$ is **cyclic** if there exists $g \in G$ such that

$$G = \{g^i \mid i \in \boldsymbol{Z}\} = \{\ldots, g^{-2}, g^{-1}, g^0, g^1, g^2, \ldots\}$$

Element g is called a **generator** for G and we write $(G,\circ) = \langle g \rangle$

❖ Cyclic groups are the basis of discrete logarithm cryptosystems

- Consider $(\boldsymbol{Z}_{47}^*,\cdot)$ with the generator $g = 5$.
  Find x such that $5^x \equiv 41 \; mod \; 47$

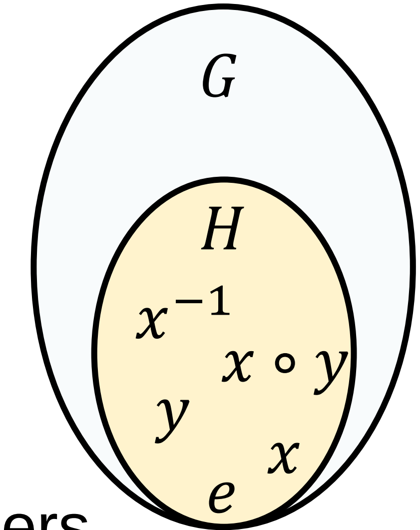
Theorem:

For every prime p, $(\boldsymbol{Z}_p^*,\cdot)$ is an abelian finite cyclic group.

Subgroups

Definition: A set $H \subseteq G$ is a **subgroup,** written $H < G$, if

$$\forall \boldsymbol{a}, \boldsymbol{b} \in \boldsymbol{H}: \quad \boldsymbol{a} \circ \boldsymbol{b} \in \boldsymbol{H}$$



Fact: a subgroup H is a group

Examples:

positive real numbers

$\boldsymbol{R}_+ < (\boldsymbol{R}^*, \cdot)$

$\{1, -1\} < (\boldsymbol{R}^*, \cdot)$

$\{e\} < G$ (for all groups)

$G < G$ (for all groups)

$\langle 20 \rangle < \langle 10 \rangle < \langle 5 \rangle < (\boldsymbol{Z}_{40}, +)$

$2\boldsymbol{Z} = \{\dots, -2, 0, 2, 4, 6, \dots\} < (\boldsymbol{Z}, +)$

$3\boldsymbol{Z} = \{\dots, -3, 0, 3, 6, 9, \dots\} < (\boldsymbol{Z}, +)$

$\langle 5 \rangle = \{0, 5, 10, \dots, 35\}$

$\langle 10 \rangle = \{0, 10, 20, 30\}$

$\langle 20 \rangle = \{0, 20\}$

Finite Cyclic Groups

Theorem: if $(G,\circ)$ is a finite group, then for all $g \in G$:

$$\boldsymbol{g^{|G|} = e}$$

Proof (finite cyclic groups):

$|G| = |\langle g \rangle| = n$

$$\overbrace{e \quad g^1 \quad g^2 \quad g^3 \quad \cdots \quad g^{n-1}}^{G} \quad g^n \quad g^{n+1} \quad g^{n+2} \quad \cdots$$

$g^n = g^3 \implies g^{n-3} = e \implies g^j = e \quad j < n$ contradiction!

Corollary I: $g^i = g^{i \bmod n} = g^{i \bmod |G|}$

Corollary II (Lagrange's theorem): if $H < G$, then the order of H divides the order of G (i.e., $|G| \,/\, |H|$)

Groups of Prime Order

Corollary II (Lagrange's theorem): if $H < G$, then the order of H divides the order of G

- **Fact:** any prime-order group is cyclic
- **Fact:** any non-trivial element ($\neq e$) in a prime-order group is a generator
- **Warning:** $(\boldsymbol{Z}_p^*, \cdot)$ is *not* a prime-order group! $|\boldsymbol{Z}_p^*| = p - 1$
- Suppose $p = 2q + 1$, with q being prime; what are the possible sub-groups of $(\boldsymbol{Z}_p^*, \cdot)$?

$$|\boldsymbol{Z}_p^*| = p - 1 = 2q$$

$$\boldsymbol{Z}_p^* = \begin{cases} \{1\}, \\ \{1, -1\}, \\ H, \quad |H| = q \\ \boldsymbol{Z}_p^* \end{cases}$$

Example: $\boldsymbol{Z}_{11}^* = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$

$11 = 2 \cdot 5 + 1$

$\{1\} < \boldsymbol{Z}_{11}^*$

$\{1, -1\} = \{1,10\} < \boldsymbol{Z}_{11}^*$

$H = \langle 3 \rangle = \langle 4 \rangle = \langle 5 \rangle = \langle 9 \rangle = \{1, 3, 4, 5, 9\} < \boldsymbol{Z}_{11}^*$

$\boldsymbol{Z}_{11}^* < \boldsymbol{Z}_{11}^*$

Next …

- Motivation
- The Public-Key Revolution
- Principles Behind Public-Key Ciphers
- Essential Number Theory for Public-Key Algorithms
- Essential Group Theory for Public-Key Algorithms
- **Public-key Examples**
  - **Diffie-Hellman (DH)**
  - Elgamal
  - RSA

Diffie-Hellman

- Based on **discrete log** problem:
  - **Given:** g, p, and $g^k \bmod p$
  - **Find:** exponent k
- Let p be prime, let g be a **generator**
  - For any $x \in \{2, \ldots, p-2\}$ there is n s.t. $x = g^n \bmod p$
- Alice selects her private value a
- Bob selects his private value b
- Alice sends $g^a \bmod p$ to Bob
- Bob sends $g^b \bmod p$ to Alice
- Both compute shared secret, $g^{ab} \bmod p$
- Shared secret can be used as symmetric key

Diffie-Hellman

$$G = \langle g \rangle \quad \leftarrow \text{public}$$

$$A = g^a \longrightarrow$$

$$B = g^b \longleftarrow$$

$$a \xleftarrow{\$} \{2, \ldots, |G| - 1\} \qquad\qquad b \xleftarrow{\$} \{2, \ldots, |G| - 1\}$$

$$Z \leftarrow B^a = (g^b)^a = g^{ab} \qquad\qquad Z' \leftarrow A^b = (g^a)^b = g^{ab}$$

Claim: $Z = Z'$

- **Public:** *g* and p
- **Private:** Alice's exponent **a**, Bob's exponent **b**
  - Alice computes $(g^b)^a = g^{ba} = g^{ab}$ mod p
  - Bob computes $(g^a)^b = g^{ab}$ mod p
  - Use $k = g^{ab}$ mod p as symmetric key

Diffie-Hellman - Example

$$\boldsymbol{Z}^*_{1019} = \langle 2 \rangle$$

$$\xrightarrow{570}$$

$$\xleftarrow{72}$$

$$493 \overset{\$}{\leftarrow} \{2, \ldots, 1017\}$$

$$570 \leftarrow 2^{493} \bmod 1019$$

$$Z \leftarrow 72^{493} \bmod 1019 \equiv \mathbf{531}$$

$$901 \overset{\$}{\leftarrow} \{2, \ldots, 1017\}$$

$$72 \leftarrow 2^{901} \bmod 1019$$

$$Z' \leftarrow 570^{901} \bmod 1019 \equiv \mathbf{531}$$

- **Public:** g (2) and p (1019)
- **Private:** Alice's exponent **a (493)**, Bob's exponent **b (901)**
- Alice computes $(g^b)^a = g^{ba} = g^{ab} \bmod p$
- Bob computes $(g^a)^b = g^{ab} \bmod p$
- Use $k = g^{ab} \bmod p$ as symmetric key

Diffie-Hellman - Security

$G = \langle g \rangle$ ← public

$A = g^a$ →

$B = g^b$ ←

$a \xleftarrow{\$} \{2, \dots, |G| - 1\}$

$b \xleftarrow{\$} \{2, \dots, |G| - 1\}$

$Z \leftarrow B^a = (g^b)^a = g^{ab}$

$Z' \leftarrow A^b = (g^a)^b = g^{ab}$

Doesn't work: $A \circ B = g^a \circ g^b = g^{a+b} \neq g^{ab}$

❖ **Security:**

✧ Must be hard to compute $Z \leftarrow g^{ab}$ given g, A, B (DH assumption)

✧ Must be hard to find a (or b) given g, A, B (DL assumption)

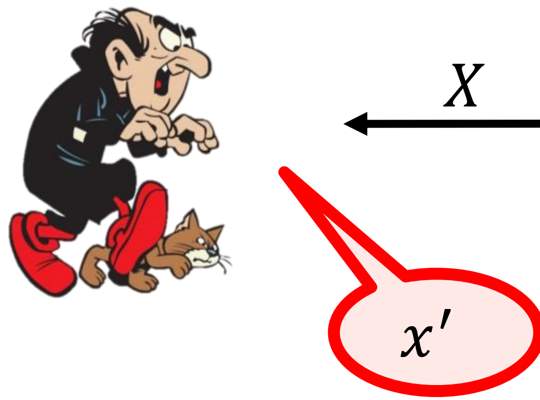
✧ If Trudy can solve discrete log problem, she can find a or b

Discrete Logarithm Problem (DLP)

$\mathbf{Exp}_{G,g}^{DLP}(A)$
1. $x \xleftarrow{\$} \{2, \dots, \lvert G\rvert - 1\}$
2. $X \leftarrow g^x$
3. $x' \leftarrow A(X)$
4. **return** $x \stackrel{?}{=} x$

Public: $G = \langle g \rangle$

<u>Challenger</u>



$\xleftarrow{X}$

$x \xleftarrow{\$} \{2, \dots, \lvert G\rvert - 1\}$

$X \leftarrow g^x$

Adversary wins if $x' = x$

In other words: $x' = \mathrm{DL}_g(X)$

Definition: The **DLP-advantage** of an adversary A is

$$\mathbf{Adv}_{G,g}^{DLP}(A) = \Pr\left[\mathbf{Exp}_{G,g}^{DLP}(A) \Rightarrow \text{true}\right]$$

Diffie-Hellman Problem (DHP)

$\mathbf{Exp}_{G,g}^{DHP}(A)$

1. $x, y \xleftarrow{\$} \{2, \ldots, |G| - 1\}$
2. $X \leftarrow g^x$
3. $Y \leftarrow g^y$
4. $z \leftarrow A(X, Y)$
5. **return** $g^z \stackrel{?}{=} g^{xy}$

Public: $G = \langle g \rangle$

$\xleftarrow{X, Y}$

z

Challenger

$x, y \xleftarrow{\$} \{2, \ldots, |G| - 1\}$

$X \leftarrow g^x$

$Y \leftarrow g^y$

Adversary wins if $g^z = g^{xy}$

Definition: The **DHP-advantage** of an adversary A is

$$\mathbf{Adv}_{G,g}^{DHP}(A) = \Pr\left[\mathbf{Exp}_{G,g}^{DHP}(A) \Rightarrow \text{true}\right]$$

DLP vs. DHP

$\mathbf{Exp}_{G,g}^{DLP}(A)$
1. $x \overset{\$}{\leftarrow} \{2, \dots, \lvert G \rvert - 1\}$
2. $X \leftarrow g^x$
3. $x' \leftarrow A(X)$
4. **return** $x \overset{?}{=} x$

$\mathbf{Exp}_{G,g}^{DHP}(A)$
1. $x, y \overset{\$}{\leftarrow} \{2, \dots, \lvert G \rvert - 1\}$
2. $X \leftarrow g^x$
3. $Y \leftarrow g^y$
4. $z \leftarrow A(X, Y)$
5. **return** $g^z \overset{?}{=} g^{xy}$

DL security $\Longleftarrow$ DH security

DL security $\overset{?}{\Longrightarrow}$ DH security

If the only way of solving the DHP requires the DLP, one would say that *"the DHP is equivalent to the DLP"*. However, this is not proven (yet).

Attacks Against the DLP

- Generic algorithms; works for *all* (cyclic) groups
  - *Brute-Force Search*

    1. Given $\boldsymbol{g}$ and $\boldsymbol{X} \in \boldsymbol{G}$
    2. for $\boldsymbol{i} = \mathbf{2}, \mathbf{3}, \ldots, |\boldsymbol{G}| - \mathbf{1}$ check if $\boldsymbol{g}^{\boldsymbol{i}} = \boldsymbol{X}$

    running time: $\mathcal{O}(|G|) \approx \mathcal{O}(2^n)$,
    given $|G| \approx 2^n$

  - **Are there better algorithms?**

- Non-generic algorithms; efficiently exploits algebraic features, i.e., the inherent structure, of given group

Generic Algorithms for Solving DL

- **Nechaev '94 & Shoup '97:** Solving DL requires time $\Omega\left(\sqrt{|G|}\right)$ in *generic* groups

- **Square-Root Attacks**:
  - *Shanks' Baby-Step Giant-Step:* Time $\mathcal{O}\left(\sqrt{|G|}\right)$ Memory $\mathcal{O}\left(\sqrt{|G|}\right)$
  - *Pollard's rho:* Time $\mathcal{O}\left(\sqrt{|G|}\right)$ Memory $\mathcal{O}(1)$

- **Consequence:** $\sqrt{|G|}$ must be large enough
  - $|G| \approx 2^{128}$ only gives $\sqrt{2^{128}} = 2^{64}$ security
  - $|G| \approx \mathbf{2^{160}}$ only gives $\sqrt{2^{160}} = \mathbf{2^{80}}$ security
  - $|G| \approx \mathbf{2^{256}}$ only gives $\sqrt{2^{256}} = \mathbf{2^{128}}$ security
  - $|G| \approx 2^{512}$ only gives $\sqrt{2^{512}} = 2^{256}$ security
  - etc...

Shanks' Baby-Step Giant-Step Method

Given: $X \leftarrow g^x$ $\quad Y \leftarrow g^m \quad m \leftarrow \left\lceil \sqrt{|G|} \right\rceil$

Find: x

$$\mathcal{O}\left(\sqrt{|G|}\right) \left\{ \begin{array}{l} X_0 \leftarrow X \cdot g^{-0} \\ X_1 \leftarrow X \cdot g^{-1} \\ X_2 \leftarrow X \cdot g^{-2} \\ X_3 \leftarrow X \cdot g^{-3} \\ \vdots \\ X_i \leftarrow X \cdot g^{-i} \\ \vdots \\ X_m \leftarrow X \cdot g^{-m} \end{array} \right.$$

$$\mathcal{O}\left(\sqrt{|G|} \cdot \log \sqrt{|G|}\right) \approx \mathcal{O}\left(\sqrt{|G|}\right)$$

Sort the values and Find "collision"

$$X_i = Y^j$$

$$X \cdot g^{-i} \cdot g^i = g^{mj} \cdot g^i$$

$$X = g^{mj+i}$$

$$\mathrm{DL}(X) = \mathrm{DL}\left(g^{mj+i}\right)$$

$$\boxed{x = mj + i}$$

Time + Memory: $\mathcal{O}\left(\sqrt{|G|}\right)$

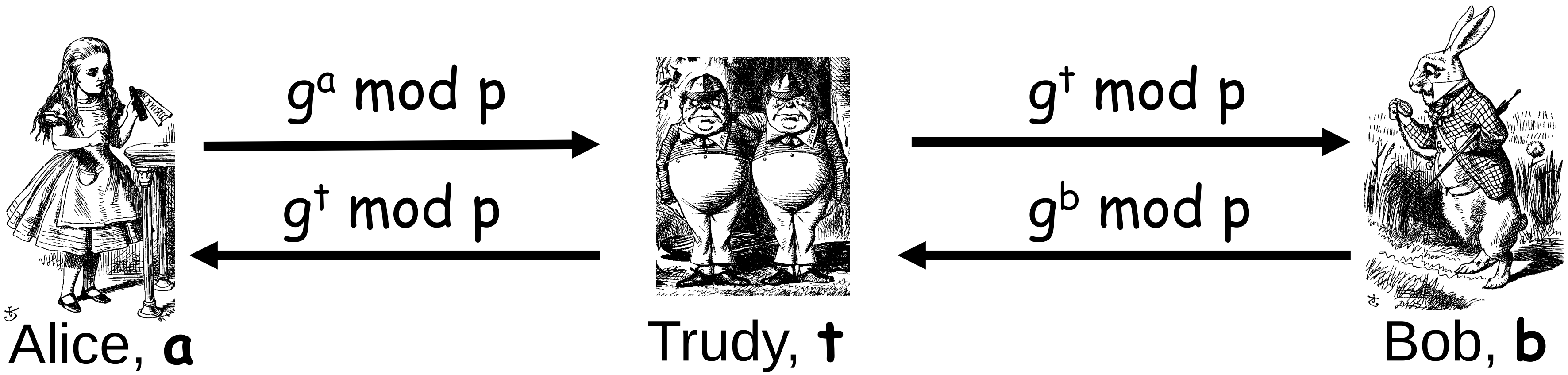
$$\left. \begin{array}{l} Y^0 \\ Y^1 \\ Y^2 \\ Y^3 \\ \vdots \\ Y^j \\ \vdots \\ Y^m \end{array} \right\} \mathcal{O}\left(\sqrt{|G|}\right)$$

Non-Generic Algorithms for DL

- Unfortunately, $(\boldsymbol{Z}_p^*,\cdot)$ is *not* a generic group!
- *Much* faster specific algorithms exist for solving DL in $(\boldsymbol{Z}_p^*,\cdot)$
  - Index-calculus method
  - Elliptic-curve method
  - Special number-field sieve (SNFS)
  - General number-field sieve (GNFS)
- Current DL-solving record: $|\boldsymbol{Z}_p^*| \approx 2^{795}$ (240-digit) using GNFS (Heninger et al. 2019)
  - Previous records: https://en.wikipedia.org/wiki/Discrete_logarithm_records
- $|\boldsymbol{Z}_p^*| \geq 2^{1024}$ typically required as a minimum today
- Better alternatives to $(\boldsymbol{Z}_p^*,\cdot)$?
  - Elliptic Curves

Diffie-Hellman - Attack

- Subject to **man-in-the-middle (MiM)** attack:

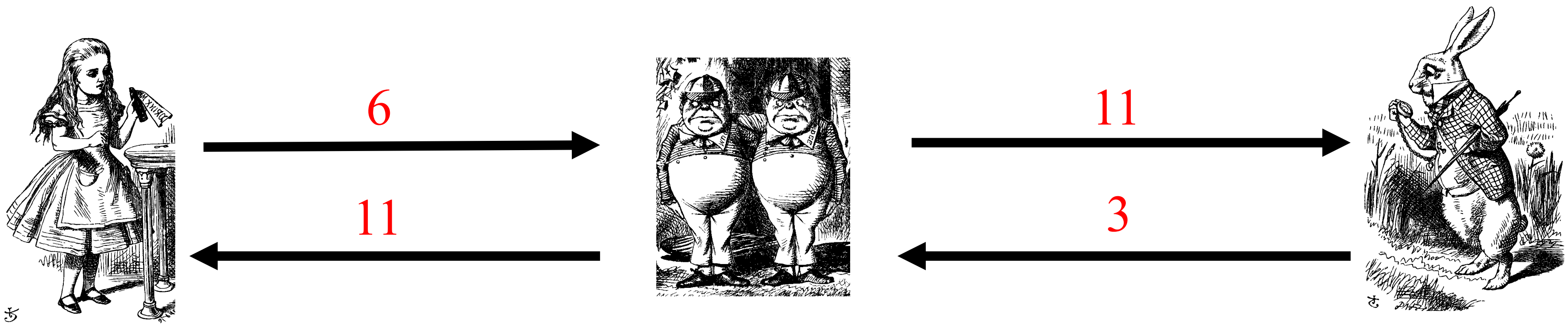


- Trudy shares secret $\mathbf{g^{at} \bmod p}$ with Alice
- Trudy shares secret $\mathbf{g^{bt} \bmod p}$ with Bob
- Alice and Bob don’t know Trudy exists!
- In any case, you MUST be aware of MiM attack on Diffie-Hellman

Diffie-Hellman - Attack

❖ Subject to **man-in-the-middle (MiM)** attack

✧ **Example:** Assume using $(\boldsymbol{Z}_{13}^{*}, \cdot)$ with $\boldsymbol{Z}_{13}^{*} = \langle 2 \rangle$



Alice, 5

$6 \leftarrow 2^{5} \bmod 13$

$Z \leftarrow 11^{5} \bmod 13 \equiv 7$

Trudy, 7

$11 \leftarrow 2^{7} \bmod 13$

$Z_A \leftarrow 6^{7} \bmod 13 \equiv 7$

$Z_B \leftarrow 3^{7} \bmod 13 \equiv 3$

Bob, 4

$3 \leftarrow 2^{4} \bmod 13$

$Z \leftarrow 11^{4} \bmod 13 \equiv 3$

Next …

- Motivation
- The Public-Key Revolution
- Principles Behind Public-Key Ciphers
- Essential Number Theory for Public-Key Algorithms
- Essential Group Theory for Public-Key Algorithms
- **Public-key Examples**
  - Diffie-Hellman (DH)
  - **Elgamal**
  - RSA

Public-Key Cipher

❖ What can we do with public-key algorithms?

Service	Algorithm Family		
	Integer Factorization	Discrete Logarithm	Elliptic Curves
Key Exchange	RSA	DH	ECDH
Digital Signature	RSA	DSA, Elgamal	ECDSA
Encryption	RSA	Elgamal	ECxxx

❖ Goal: Develop an encryption scheme from DH key exchange

$$\boldsymbol{G = \langle g \rangle, p}$$

Alice, a

Message x

$$A = g^a \longrightarrow$$

$$\longleftarrow B = g^b$$

Bob, b

$$K_{AB} \equiv B^a = \left(g^b\right)^a = g^{ab} \bmod p \qquad\qquad K_{AB} \equiv A^b = (g^a)^b = g^{ab} \bmod p$$

$$Y \equiv x \cdot K_{AB} \bmod p \quad \xrightarrow{\;Y\;} \quad x \equiv Y \cdot (K_{AB})^{-1} \bmod p$$

Public-Key Encryption - Procedure

- Scenario:
  - Alice wants to send an encrypted message to Bob
- Procedure:
  - Bob computes a public and a private key, the keypair
  - Bob publishes his public key
  - Alice Encrypts the message using Bob’s public key
  - Alice sends the message to Bob.
  - Bob encrypts the message using his private key
- Effect:
  - Nobody intercepting the message can read
  - nor alter it unrecognized

Elgamal Public-key Cryptosystem

- Elgamal is a public-key cryptosystem that was developed by Dr. Taher Elgamal in 1985
  - Based on the Diffie–Hellman key exchange, but with reordering of steps
- Key aspects:
  - Based on the Discrete Logarithm problem
  - Randomized encryption
- Application:
  - Establishing a secure channel for key sharing
  - Encrypting messages

Elgamal - Key Generation

- Bob (**receiver**) must do the following:
  1. Generate a large random prime number p
  2. Choose a generator number α
  3. Choose secret number d, $d \in \{2, 3, \ldots, p-2\}$
  4. Compute $\beta = \alpha^{d} \, mod \, p$

- Public key: (β, p, α)
- Private key: d

- **Notes:**
  - The public key β is fixed, p and α are chosen by Bob
  - p must be > 300 digits (i.e., > 1024 bits)

Elgamal - Encryption

- Alice (**sender**) must do the following:
  1. Obtain public key (β, p, α) from Bob (receiver)
  2. Choose an integer i, $i \in \{2, 3, \ldots, p-2\}$
  3. Compute $K_E = \alpha^i \; mod \; p$, K_E: Ephemeral Key
  4. Compute $K_M = \beta^i \; mod \; p$, K_M: Masking Key
  5. Represent the plaintext as an integer x
  6. Compute ciphertext $Y = x \times K_M \; (mod \; p)$
  7. Send (Y, K_E) to Bob

- **Notes:**
  - i must be new for each encryption, i.e., K_E must be different for every plaintext
  - Because of that, Elgamal is a “probabilistic encryption scheme”

Elgamal - Decryption

- Bob (**receiver**) does the following:
  1. Obtain ciphertext and ephemeral key (Y, K_E) from Alice (sender)
  2. Compute $K_M = {K_E}^{d} \ mod\ p$, K_M: Masking Key
  3. Recover plaintext $x = Y \times {K_M}^{-1}\ (mod\ p)$

- **Notes:**
  - To compute ${K_M}^{-1}$, we need to apply the ***square-and-multiply*** algorithm to calculate K_M first, and then apply the ***extended Euclidean algorithm*** to calculate the inverse
  - However, using the ***Fermat's little theorem***, the computation of ${K_M}^{-1}$ can be simplified as follows:

$$
\begin{aligned}
{K_M}^{-1} &\equiv \left({K_E}^{d}\right)^{-1} (mod\ p) \equiv {K_E}^{-d} \times 1\ (mod\ p) \\
&\equiv {K_E}^{-d} \times {K_E}^{p-1} (mod\ p) \equiv {K_E}^{p-1-d} (mod\ p)
\end{aligned}
$$

  - Thus, steps (2) and (3) are merged as: $x = Y \times {K_E}^{p-1-d}\ (mod\ p)$

Generator Number

❖ Testing if α is generator number:

1. α must $\in \{1, 2, 3, \dots, p-1\}$
2. Find $\emptyset(p) = p - 1$
3. Find the all factors of $\emptyset(p)$, $\{f_1, f_2, f_3, \dots, f_n\} - \{1\}$
4. Find $\{q_1, q_2, q_3, \dots, q_n\}$, where $q_i = f_i$
   - For redundant factors $q_i = f_i^h$, where $h = 1, 2, \dots, \mathrm{freq}(f_i)$
5. α is generator *iff* $w_i = (\alpha)^{\emptyset(p)/q_i} \neq 1 \; mod \; p$, for all q_i

Generator Number - Example 1

- Let $p = 11$, $\alpha = 2$, test if α is generator number

$\phi(p) = 10$, factors of 10 = {2, 5} $\Rightarrow q_1 = 2, q_2 = 5$

$w_1 = 2^{10/2} \bmod 11 = 10 \neq 1$

$w_2 = 2^{10/5} \bmod 11 = 4 \neq 1$

$\Rightarrow \alpha$ is a generator number

Generator Number - Example 2

- Let p = 11, α = 3, test if α is generator number

  $\phi(p)$ = 10, factors of 10 = {2, 5} $\Rightarrow q_1 = 2,\ q_2 = 5$

  $w_1 = 3^{10/2} \bmod 11 = 1$

  $w_2 = 3^{10/5} \bmod 11 = 9 \neq 1$

  $\Rightarrow \alpha$ is NOT a generator number

Generator Number – Example 3

❖ Let p = 37, α = 2, test if α is generator number

$\phi(p)$ = 36, factors of 36 = {2, 2, 3, 3}

$\Rightarrow q_1 = 2^1 = 2,\ q_2 = 2^2 = 4,\ q_3 = 3^1 = 3,\ q_4 = 3^2 = 9$

$w_1 = 2^{36/2} \bmod 37 = 36 \neq 1$

$w_2 = 2^{36/4} \bmod 37 = 31 \neq 1$

$w_3 = 2^{36/3} \bmod 37 = 26 \neq 1$

$w_4 = 2^{36/9} \bmod 37 = 16 \neq 1$

$\Rightarrow \alpha$ is a generator number

Key Generation - Example

❖ Let p = 11, α = 2, and d = 5

⇒ Calculate $\beta = \alpha^{d} \, mod \, p = 2^{5} \, mod \, 11 = 10$

Public key: $(\beta, p, \alpha) = (10, 11, 2)$

Private key: $d = (5)$

Encryption/Decryption - Example

- Let public key $(\beta, p, \alpha) = (10, 11, 2)$ and plaintext $x = (1, 7, 5)$

- Encryption:

  $K_E = \alpha^i \bmod p$, $K_M = \beta^i \bmod p$, and $Y = x \times K_M \pmod p$

  $\boldsymbol{x = 1}$**, choose a random integer** $\boldsymbol{i = 6}$

  $\Rightarrow K_E = 2^6 \bmod 11 = 9$, $K_M = 10^6 \bmod 11 = 1$, and $Y = 1 \times 1 \pmod{11} = 1$

  $\boldsymbol{x = 7}$**, choose a random integer** $\boldsymbol{i = 4}$

  $\Rightarrow K_E = 2^4 \bmod 11 = 5$, $K_M = 10^4 \bmod 11 = 1$, and $Y = 7 \times 1 \pmod{11} = 7$

  $\boldsymbol{x = 5}$**, choose a random integer** $\boldsymbol{i = 7}$

  $\Rightarrow K_E = 2^7 \bmod 11 = 7$, $K_M = 10^7 \bmod 11 = 10$, and $Y = 5 \times 10 \pmod{11} = 6$

➔ Send: $(\mathbf{1}, \mathbf{9})(\mathbf{7}, \mathbf{5})(\mathbf{6}, \mathbf{7})$

Encryption/Decryption - Example

- Let public key $(\beta, p, \alpha) = (10, 11, 2)$, private key $d = (5)$, and received ciphertext $(\mathbf{1}, \mathbf{9})(\mathbf{7}, \mathbf{5})(\mathbf{6}, \mathbf{7})$

- Decryption:

$$K_M{}^{-1} = K_E{}^{p-1-d} \,(mod\; p) \quad \text{and} \quad x = Y \times K_M{}^{-1} \,(mod\; p)$$

$\boldsymbol{Y = 1, K_E = 9}$

$$\Rightarrow K_M{}^{-1} = 9^{11-1-5} \,(mod\; 11) = 1 \quad \text{and} \quad x = 1 \times 1 \,(mod\; 11) = 1$$

$\boldsymbol{Y = 7, K_E = 5}$

$$\Rightarrow K_M{}^{-1} = 5^{11-1-5} \,(mod\; 11) = 1 \quad \text{and} \quad x = 7 \times 1 \,(mod\; 11) = 7$$

$\boldsymbol{Y = 6, K_E = 7}$

$$\Rightarrow K_M{}^{-1} = 7^{11-1-5} \,(mod\; 11) = 10 \quad \text{and} \quad x = 6 \times 10 \,(mod\; 11) = 5$$

Plaintext $x = (1, 7, 5)$

Elgamal Public-key Cryptosystem

- **Encryption** requires *two modular exponentiations* that are independent of the plaintext
  - Can be computed ahead of time if need be.

On the other hand,

- **Decryption** only requires *one modular exponentiation*
- **Ciphertext** is **twice** as long as the corresponding plaintext (disadvantage)

Elgamal Attacks

❖ Attack computes DLP

✧ $d = log_{\alpha}\beta$ ➔ $K_M{}^{-1} = K_E{}^{p-1-d}$, and $x = Y \cdot K_M{}^{-1}$

OR

✧ $i = log_{\alpha}K_E$ ➔ $K_M = \beta^i$, and $x = Y \cdot K_M{}^{-1}$

Thus, the DLP needs to be a computational hard problem ➔ p must be large $p \geq 2^{1024}$

❖ Attack Re-use of secret exponent ***i***

$K_E = \alpha^i$, $K_M = \beta^i$

$$Y_1 \equiv x_1 \cdot K_M \xrightarrow{(Y_1, K_E)}$$

$$Y_2 \equiv x_2 \cdot K_M \xrightarrow{(Y_2, K_E)}$$

Assume Trudy knows x_1 (known-plaintext attack).

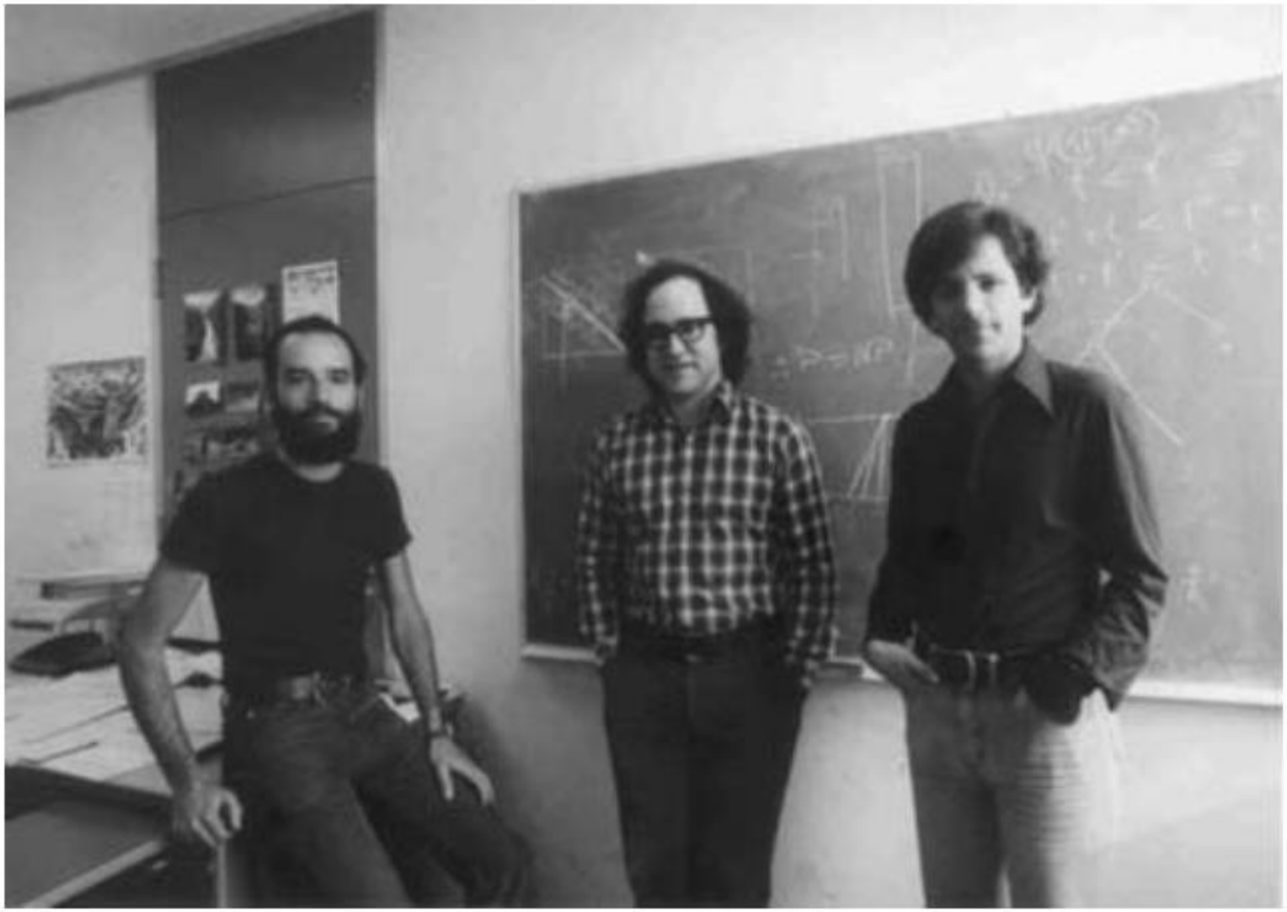
$K_M = Y_1 \cdot x_1{}^{-1} = Y_2 \cdot x_2{}^{-1}$ ➔ $x_2 = Y_2 \cdot Y_1{}^{-1} \cdot x_1 \; mod \; p$

Next …

- Motivation
- The Public-Key Revolution
- Principles Behind Public-Key Ciphers
- Essential Number Theory for Public-Key Algorithms
- Essential Group Theory for Public-Key Algorithms
- **Public-key Examples**
  - Diffie-Hellman (DH)
  - Elgamal
  - **RSA**

RSA

- Invented in 1977 by **R**ivest, **S**hamir, and **A**dleman
  - RSA is the ***gold standard*** in public key crypto



RSA – Key Generation

- Let p and q be two ***large prime*** numbers
- Let $N = pq$ be the **modulus**
- Calculate $\phi(N) = (p-1)(q-1)$
- Choose e, $e \in \{1, 2, \dots, \phi(N) - 1\}$, relatively prime to $\phi(N)$
  - i.e., $\gcd(e, \phi(N)) = 1$ (why?)
- Find d such that $ed \equiv 1 \bmod \phi(N)$
  - i.e., $d \equiv e^{-1} \bmod (p-1)(q-1)$
- **Public key** is (N, e)
- **Private key** is d
- In practice, p & q should be large (≥ 1024 bits)
- Thus, N & d should be large (≥ 2048 bits)

RSA – Enc/Dec

- Message M (i.e., plaintext) is treated as a number
- To encrypt plaintext $M \Rightarrow C = M^e \bmod N$
- To decrypt ciphertext $C \Rightarrow M = C^d \bmod N$
- Recall that e and N are public
- If Trudy can factor $N = pq$, she can use e to easily find d since $ed = 1 \bmod (p-1)(q-1)$
- **Factoring the modulus breaks RSA**
  - Is factoring the only way to break RSA?

Does RSA Really Work?

- Given $C \equiv M^e \bmod N$, we must show

  $M \equiv C^d \equiv M^{ed} \bmod N$

- Facts:

  1) $\phi(N) = (p-1)(q-1)$
  2) $ed \equiv 1 \bmod (p-1)(q-1) \equiv 1 \bmod \phi(N)$
  3) By definition of "mod":

     $ed = t\phi(N) + 1$, where t is an integer

$\Rightarrow$ Must show $M \equiv C^d \equiv M^{ed} \equiv M^{t\phi(N)+1} \equiv M^{t\phi(N)} M^1$

$$\equiv (M^{\phi(N)})^t M \bmod N$$

Does RSA Really Work?

- Must show $(M^{\phi(N)})^t\, M \equiv M \bmod N$
- Case 1: gcd(M, N) = 1
  - Use Euler's Theorem ☺ $\Rightarrow$ if $\gcd(M, N) = 1$, then $1 \equiv M^{\phi(N)} \bmod N$

  $\Rightarrow C^d \equiv M^{ed} \equiv (M^{\phi(N)})^t\, M \equiv (1)^t\, M \equiv \mathbf{M \bmod N}$
- Case 2: gcd(M, N) = gcd(M, p · q) ≠ 1
  - Can't use Euler's Theorem directly ☹
  - p and q are primes $\Rightarrow M = (r \cdot p)$ or $M = (s \cdot q)$, where $r < q$ and $s < p$
    - Note that $M \neq (x \cdot p \cdot q)$ (i.e., isn't factor of both p & q)
  - Assume $M = (r \cdot p)$ — will also work if $M = (s \cdot q) \Rightarrow \gcd(M, q) = 1$
  - Using Euler's Theorem $\Rightarrow 1^t \equiv (M^{\phi(q)})^t \bmod q$
  - Consider again $(M^{\phi(N)})^t \equiv (M^{(p-1)(q-1)})^t \equiv ((M^{\phi(q)})^t)^{(p-1)} \equiv 1^{(p-1)} \equiv 1 \bmod q$
  - But by definition of "mod" $\Rightarrow (M^{\phi(N)})^t = u\,q + 1$, where u is an integer

  $\Rightarrow M(M^{\phi(N)})^t = M(u\,q + 1) = M\,u\,q + M = (r\,p)\,u\,q + M = (r\,u)\,N + M \equiv M \bmod N$

  $\Rightarrow C^d \equiv M^{ed} \equiv (M^{\phi(N)})^t\, M \equiv \mathbf{M \bmod N}$

Simple RSA Example

- Example of RSA
  - Select "large" primes $p = 11,\ q = 3$
  - Then $N = pq = 33$ and $(p-1)(q-1) = 20$
  - Choose $e = 3$ (relatively prime to 20)
  - Find d such that $ed \equiv 1 \bmod 20$
    - We find that $d = 7$ works
- **Public key:** $(N, e) = (33, 3)$
- **Private key:** $d = 7$

Simple RSA Example

- **Public key:** $(N, e) = (33, 3)$
- **Private key:** $d = 7$
- Suppose message $M = 4$
- Ciphertext C is computed as

$$C = M^e \bmod N = 4^3 = 64 \equiv \mathbf{31} \bmod 33$$

- Decrypt C to recover the message M by

$$M \equiv C^d \bmod N = 31^7 = 27{,}512{,}614{,}111$$

$$= 833{,}715{,}579 * 33 + 4$$

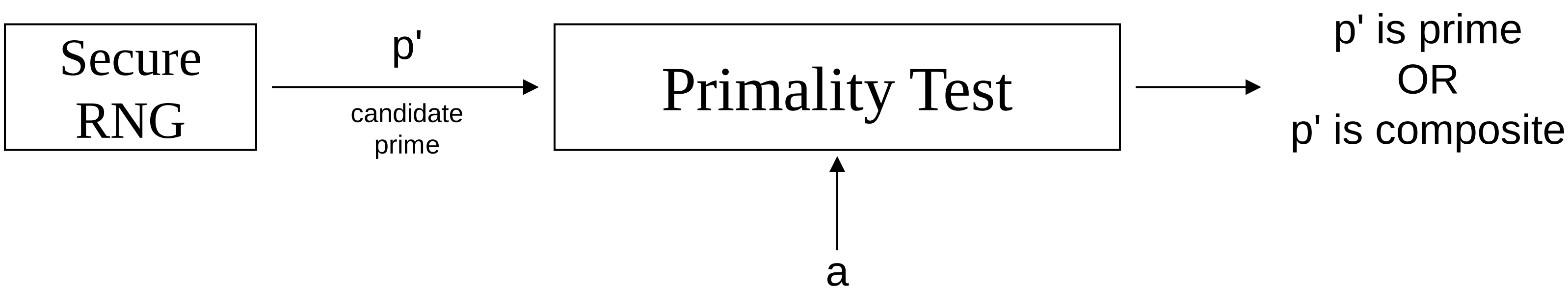
$$\equiv \mathbf{4} \bmod 33$$

Key Generation

- Like all asymmetric schemes, RSA has set-up phase during which the private and public keys are computed
- Key generation: choose two large, distinct primes p and q
  - Not-trivial!
- So, how to find p and q?

Finding Large Primes 1/2

- Generating keys for RSA requires finding two large primes *p* and *q* such that $n = p * q$ is sufficiently large
- The size of *p* and *q* is typically half the size of the desired size of *n*
- To find primes, random integers are generated and tested for primality:



- For this approach to work, we have to answer two questions:
  1. How many random integers do we have to test before we have a prime?
     - If the likelihood of a prime is too small, it might take too long
  2. How fast can we check whether a random integer is prime?
     - Again, if the test is too slow, the approach is impractical
- It turns out that both steps are reasonably fast!!!

How many primes are there?

- By looking at the first few positive integers that primes become **less dense** as the value increases:
  2,3,5,7,11,13,17,19,23,29,31,37, . . .
- What is the chance that a random number (e.g., 512 bits) is a prime?
  - The chance that a randomly picked integer $\tilde{p}$ is a prime is approximately $1/\ln(\tilde{p})$ (based on “prime number theorem”)
  - In practice, test only **odd** numbers so that the likelihood doubles
    $\Rightarrow$ probability for a random **odd** number $\tilde{p}$ to be prime is

$$\boxed{P(\tilde{p} \text{ is prime}) \approx \frac{2}{\ln(\tilde{p})}}$$

How many primes are there?

- **Example:** RSA with a 2048-bit N, each of p and q ≈ 2^{1024}

  $\Rightarrow P(\tilde{p} \text{ is prime}) \approx 2/\ln(2^{1024}) = 2/(1024 \ln(2)) \approx 1/355$

  $\Rightarrow$ Expect to test 355 random numbers before finding a prime

- Likelihood of integers being primes decreases slowly, proportional to integer bit length

  $\Rightarrow$ For very long RSA parameters (e.g., 4096 bit), the density of primes is still sufficiently high

How long to check if integer is prime?

- Factoring *p* and *q* to test for primality is typically not feasible
- However, we are not interested in the factorization, we only want to know whether *p* and *q* are composite or prime
- Typical **primality tests** are probabilistic, i.e., they are not 100% accurate but their output is correct with very high probability
- A probabilistic test has two outputs:
  - p' is composite – always true
  - p' is a prime – only true with a certain probability
- Among the well-known primality tests are the following:
  - Fermat Primality-Test
  - Miller-Rabin Primality-Test

Fermat Primality-Test

- Basic idea: Fermat's Little Theorem holds for all primes, i.e., if a number p' is found for which $a^{p'-1} \not\equiv 1 \ mod \ p'$, then p' is **not** a prime

<u>Algorithm: Fermat Primality-Test</u>

Input: Prime candidate p', security parameter s

Output: (p' is composite) or (p' is likely a prime)

1. **FOR** $i = 1$ **TO** s

2. choose random $a \in \{2,3, ..., p'-2\}$

3. **IF** $a^{p'-1} \not\equiv 1 \bmod p'$ **THEN**

4. **RETURN** (p' is composite)

5. **RETURN** (p' is likely a prime)

- For certain numbers ("Carmichael numbers", such as 561 = 3x11x17) this test returns (p' is likely a prime) often even though these numbers are composite!!!
- Therefore, the Miller-Rabin Test is preferred

Theorem for Miller-Rabin's test

- The more powerful Miller-Rabin Test is based on the following theorem

Theorem

Given the decomposition of an odd prime candidate *p'*

$$p' - 1 = 2^u \cdot r$$

where *r* is odd. If we can find an integer *a* such that

$$a^r \not\equiv 1 \bmod p' \qquad \text{and} \qquad a^{2^j r} \not\equiv p' - 1 \bmod p'$$

For all $j = \{0, 1, \ldots, u-1\}$, then *p'* is composite.

Otherwise it is probably a prime.

- This theorem can be turned into an algorithm

Miller-Rabin Primality-Test 1/3

Algorithm: Miller-Rabin Primality-Test

Input: Prime candidate p' with p'-1 = (2^u . r), security parameter s

Output: (p' is composite) or (p' is likely a prime)

1. **FOR** i = 1 **TO** s

2. choose random $a \in \{2,3, ..., p'-2\}$

3. $z \equiv a^r \bmod p'$

4. **IF** $z \neq 1$ **AND** $z \neq p'-1$ **THEN**

5. **FOR** j = 1 **TO** u-1

6. $z \equiv z^2 \bmod p'$

7. **IF** $z = 1$ **THEN**

8. **RETURN** (p' is composite)

9. **IF** $z \neq p'-1$ **THEN**

10. **RETURN** (p' is composite)

11. **RETURN** (p' is likely a prime)

Miller-Rabin Primality-Test 2/3

- Possible that a composite number $\tilde{p}$ gives the incorrect statement "prime"
- However, the likelihood of this rapidly decreases as we run the test with several different random base elements ***a***
- Number of runs is given by security parameter ***s*** in the Miller–Rabin test
- Following table shows how many different values ***a*** must be chosen in order to have a probability $\leq 2^{-80}$ that a composite is ***incorrectly*** detected as a prime

Bit lengths of $\tilde{p}$	Security parameter s
250	11
300	9
400	6
500	5
600	3

Miller-Rabin Primality-Test 3/3

- **Example:** Let $\tilde{p} = 91 \Rightarrow \tilde{p}-1 = 2^1 \cdot 45$. Select a security parameter of $\boldsymbol{s} = 4 \Rightarrow$ Choose $\boldsymbol{s}$ times a random value $\boldsymbol{a}$:
  1. Let $\boldsymbol{a} = 12 \Rightarrow \boldsymbol{z} = 12^{45} \equiv 90 \bmod 91$, hence, $\tilde{p}$ is likely prime.
  2. Let $\boldsymbol{a} = 17 \Rightarrow \boldsymbol{z} = 17^{45} \equiv 90 \bmod 91$, hence, $\tilde{p}$ is likely prime.
  3. Let $\boldsymbol{a} = 38 \Rightarrow \boldsymbol{z} = 38^{45} \equiv 90 \bmod 91$, hence, $\tilde{p}$ is likely prime.
  4. Let $\boldsymbol{a} = 39 \Rightarrow \boldsymbol{z} = 39^{45} \equiv 78 \bmod 91$, hence, $\tilde{p}$ is composite.

  Since the numbers 12, 17 and 38 give incorrect statements for the prime candidate $\tilde{p} = 91$, they are called “liars for 91”

Attacks and Countermeasures

- There are two distinct types of attacks on cryptosystems
  1) Analytical attacks
     - Try to break the mathematical structure of the underlying problem of RSA
     - RSA is typically exposed to these **analytical** attack vectors

     a) **Mathematical attacks**
        - The best-known attack is factoring of N in order to obtain $\phi(N)$
        - Can be prevented using a sufficiently large modulus N
        - Currently, it is recommended that N should have a bit length between 2048 and 4096 bits

     b) **Protocol attacks**
        - Exploit the malleability of RSA, i.e., the property that a ciphertext can be transformed into another ciphertext which decrypts to a related plaintext – without knowing the private key
        - Can be prevented by proper padding

Attacks and Countermeasures

- There are two distinct types of attacks on cryptosystems
  2) Implementation attacks
     - Attack a real-world implementation by exploiting inherent weaknesses in the way RSA is realized in software or hardware
     - **Implementation** attacks can be one of the following
       a) **Side-channel analysis**
          - Exploit physical leakage of RSA implementation (e.g., power consumption, electromagnetic emanation, etc.)
       b) **Fault-injection attacks**
          - Inducing faults in the device while Chinese Remainder Theorem (CRT) is executed can lead to a complete leakage of the private key

Slides Original Source

- Jonathan Katz and Yehuda Lindell, "Introduction to Modern Cryptography," Third Edition, 2021
- M. Stamp, "Information Security: Principles and Practice," John Wiley
- B. Forouzan, "Cryptography and Network Security," McGraw-Hill
- C. Paar and J. Pelzl, "Understanding Cryptography – A Textbook for Students and Practitioners," Springer (www.crypto-textbook.com)