

CSE 3400/CSE 5850 - Introduction to Computer & Network  
Security  
/ Introduction to Cybersecurity

Lecture 10  
Public Key Cryptography – Part I

Ghada Almashaqbeh  
UConn

\*Adapted from the textbook slides

# Outline

- Introduction to public key cryptography and motivation.
- Number theory review.
- The discrete log assumption.
- The Diffie-Hellman key exchange protocol.

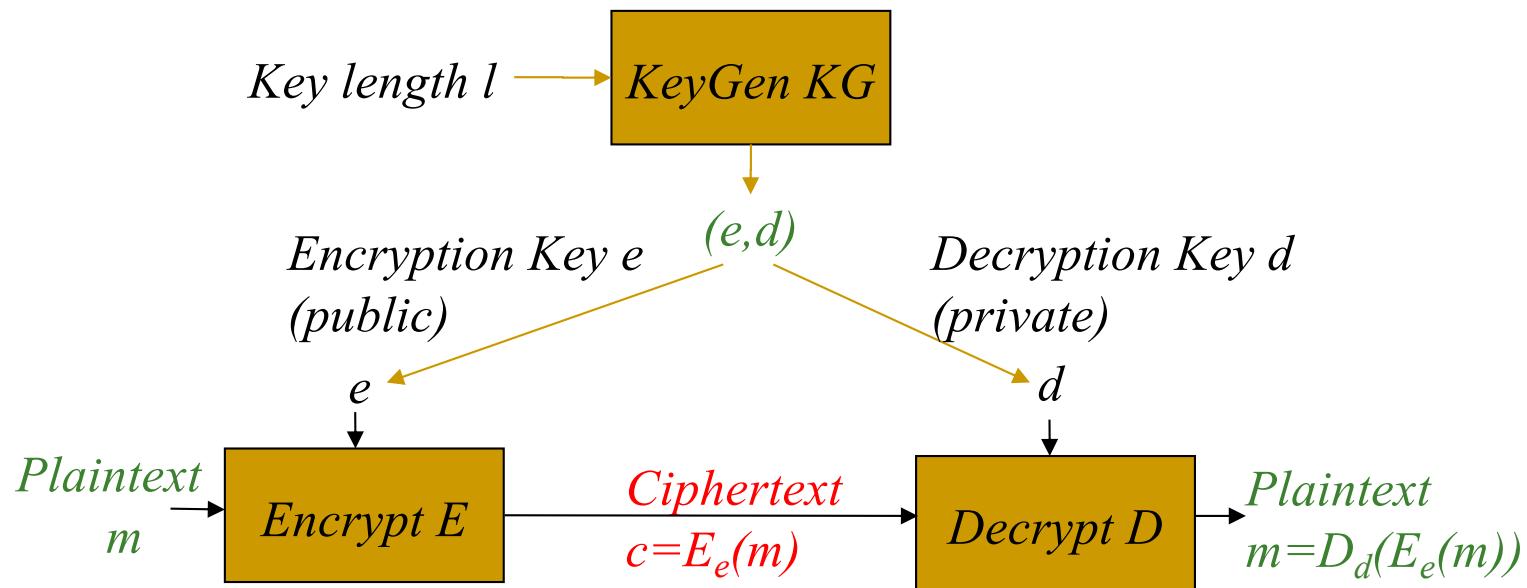
# Intro to Public Key Cryptography

# Public Key Cryptography

- Kerckhoff's principle: the cryptosystem (algorithm) is public
- What we learned until now: ***symmetric or shared key*** setting
  - Only the key is secret (unknown to attacker)
  - Same key for encryption and decryption → if you can encrypt, you can also decrypt!
  - Shared keys for MACs and PRFs, etc.
- But can we give ***asymmetric*** cryptographic capability, e.g., encryption capability without a decryption capability?
  - Yes, using public key cryptography!

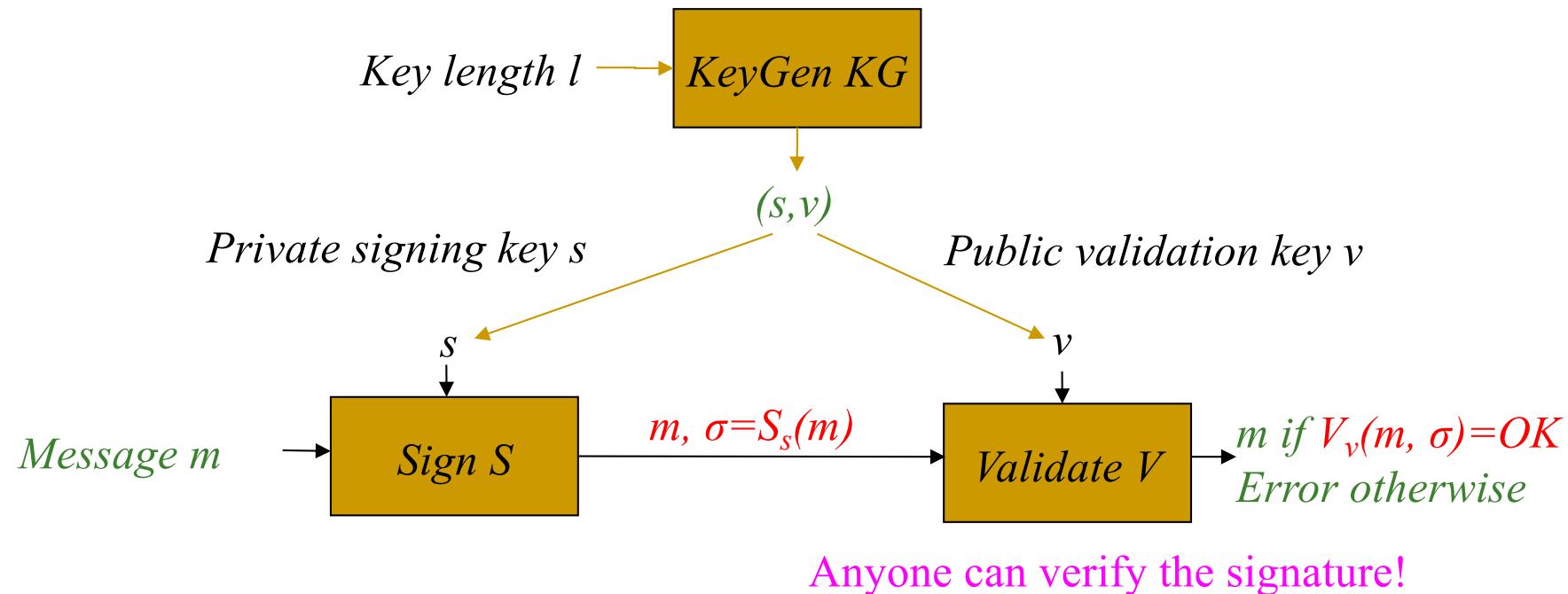
# Public Key Cryptosystem (PKC)

- Kerckhoff: cryptosystem (algorithm) is public.
- [DH76]: can *encryption key be public*, too??
  - Decryption key will be different (and private).
  - Everybody can send me emails, only I can read them.



# Is it Only About Encryption?

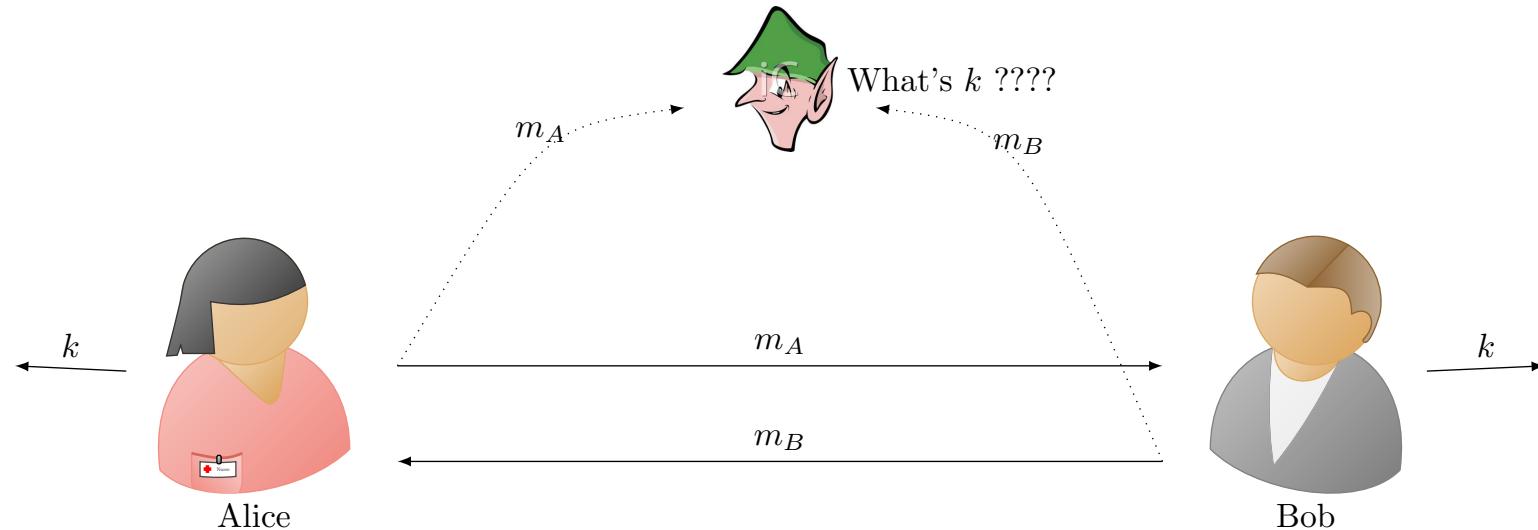
- Also: Digital signatures for integrity and non-repudiation.
  - Sign with private key  $s$ , verify with public key  $v$
  - (Recall MACs; a shared key cryptosystem for message authentication).



# More: Key-Exchange Protocol

## ■ Key Exchange Protocols.

- ❑ Establish shared key between Alice and Bob **without** assuming an existing shared ('master') key !!
- ❑ Use public information from Alice and Bob to setup shared secret key  $k$ .
- ❑ Eavesdroppers cannot learn the key  $k$ .



# Public keys solve more problems ...

- Signatures provide **evidence**
  - Everyone can validate, only ‘owner’ can sign
- Establish shared secret keys
  - Use authenticated public keys
    - Signed by trusted certificate authority (CA)
  - Or: use DH (Diffie Hellman) key exchange
- Stronger resiliency to key exposure
  - Perfect forward secrecy and recover security
  - Threshold security
    - Resilient to key exposure of  $t$  out of  $n$  parties

# Public keys are easier...

- To distribute:
  - From directory or from incoming message (still need to be authenticated)
  - Less keys to distribute (same public key to all)
- To maintain:
  - Can keep in non-secure storage as long as being validated (e.g. using MAC) before using
  - Less keys:  $O(|parties|)$ , not  $O(|parties|^2)$
- So: why not **always** use public key crypto?

# The Price of PKC

- Assumptions
  - Applied PKC algorithms are based on a small number of specific computational assumptions
    - Mainly: hardness of factoring and discrete-log
    - Both may fail against quantum computers
- Overhead
  - Computational
  - Key length
  - Output length (e.g., ciphertext or signature)

# Public key crypto is harder...

- Requires related public, private keys
  - Usually we say a keypair (pk, sk)
  - Public key does not expose private key
- Substantial overhead
  - Successful cryptanalytic shortcuts → need long keys
  - Elliptic Curves (EC) may allow shorter key (almost no shortcuts found)
  - Complex computations, e.g., complex (slow) key generation

Commercial-grade security from [LV02]

[LV02]	Required key size		
Year	AES	RSA, DH	ECIES
2010	78	1369	160
2020	86	1881	161
2030	93	2493	176
2040	101	3214	191

For the table:

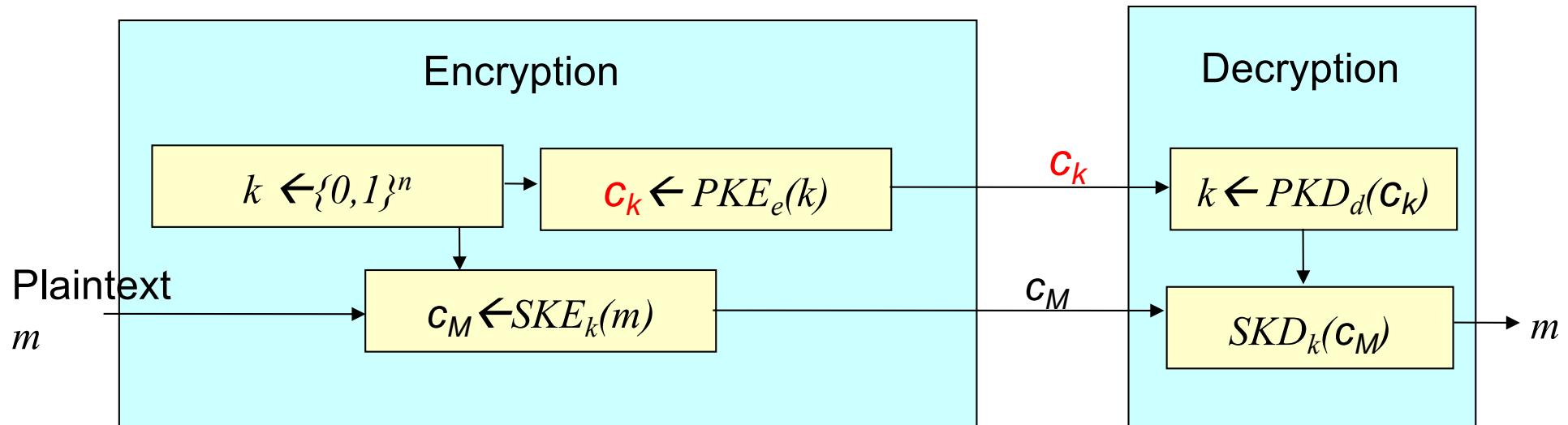
- The year indicates until when confidentiality to be preserved.
- AES: A symmetric encryption scheme
- RSA and DH: encryption schemes based on factoring and discrete log hardness problems
- ECIES: Elliptic Curve Integrated Encryption Scheme

# In Sum

- Minimize the use of PKC
- In particular: as possible, apply PKC only to ***short inputs***
- How??
  - For signatures:
    - **Hash-then-sign**
  - For public-key encryption:
    - **Hybrid encryption**

# Hybrid Encryption

- Challenge: public key cryptosystems are slow
- Hybrid encryption:
  - Use a shared key encryption scheme to encrypt all messages.
  - But use a public key encryption system to exchange the shared key.
    - Alice generates  $k$ , encrypts it under Bob's public key and sends the ciphertext  $c_k$  to Bob.
    - Bob can decrypt and recover  $k$ , and then use  $k$  to decrypt  $c_M$ .



# Going Forward

- First, introduce the mathematical concepts (mainly number theory) that we need for a particular primitive/protocol.
  - This would involve hardness problems/assumptions.
- Then, study the primitive/protocol itself.
- Lastly, and as before, show correctness and reason about security.
  - In general, security will be based on mathematical hardness problems.



# Number Theory Review

## --Modular Arithmetic--

# Notation

- $\mathbb{Z}$  : The set of all integers  $\{\dots, -3, -2, -1, 0, 1, 2, 3, \dots\}$ .
- $\mathbb{Z}_n$ : The set of integers modulo  $n$ , i.e.,  $\{0, 1, \dots, n - 1\}$
- $\mathbb{N}$  : The set of natural numbers  $\{1, 2, 3, \dots\}$ .
- **Prime number:**  $p$  is prime if its only factors are 1 and  $p$ .
- **Composite number:** not prime.
- **Co-prime numbers:**  $m$  and  $n$  are co-prime if their greatest common divisor is 1.
- $\mathbb{Z}_p^*$ : For a prime  $p$ , it is the set of integers modulo  $p$  *excluding zero*, i.e.,  $\{1, \dots, p - 1\}$
- $\mathbb{Z}_n^*$ : For a composite  $n$ , it is the set of positive integers that are less than  $n$  (excluding zero) and co-prime to  $n$ .

# The Modulo Operation

**Definition 1.2** (The modulo operation). *Let  $a, m \in \mathbb{Z}$  be integers such that  $m > 0$ . We say that an integer  $r$  is a residue of  $a$  modulo  $m$  if  $0 \leq r < m$  and  $(\exists i \in \mathbb{Z})(a = r + i \cdot m)$ . For any given  $a, m \in \mathbb{Z}$ , there is exactly one such residue of  $a$  modulo  $m$ ; we denote it by  $a \pmod{m}$ .*

Properties (make it easier to compute complex modular arithmetic expressions):

$$(a + b) \pmod{m} = [(a \pmod{m}) + (b \pmod{m})] \pmod{m} \quad (1.2)$$

$$(a - b) \pmod{m} = [(a \pmod{m}) - (b \pmod{m})] \pmod{m} \quad (1.3)$$

$$a \cdot b \pmod{m} = [(a \pmod{m}) \cdot (b \pmod{m})] \pmod{m} \quad (1.4)$$

$$a^b \pmod{m} = (a \pmod{m})^b \pmod{m} \quad (1.5)$$

# Examples

- $7 \bmod 9 = ?$
- $13 \bmod 8 = ?$
- $0 \bmod 11 = ?$
- $4 \bmod 4 = ?$
- $(30 + 66) \bmod 11 = ?$
- How about:  $445 \cdot (81 \cdot 34^{13} + 83 \cdot 33^{345}) \bmod 4$

Denote  $445 \cdot (81 \cdot 34^{13} + 83 \cdot 33^{345}) \bmod 4$  by  $x$ . Then we find  $x$  as follows:

$$\begin{aligned}x &= 445 \cdot (81 \cdot 34^{13} + 83 \cdot 33^{345}) \bmod 4 \\&= (445 \bmod 4) \cdot ((81 \bmod 4) \cdot (34 \bmod 4)^{13} + \\&\quad + (83 \bmod 4) \cdot (33 \bmod 4)^{345}) \bmod 4 \\&= 1 \cdot (1 \cdot 2^{13} + 3 \cdot 1^{345}) \bmod 4 \\&= (2 \cdot 4^6 + 3) \bmod 4 \\&= 3 \bmod 4 = 3\end{aligned}$$

# Multiplicative Inverse

- Needed to support division in modular arithmetic.
  - Division does not always produce integers.
  - Modular arithmetic requires integers to work with!!
- To compute  $a/c \text{ mod } m$ , multiply  $a$  by the multiplicative inverse of  $c$ .
  - That is compute  $a/c \text{ mod } m = ac^{-1} \text{ mod } m$ .
  - Where  $c^{-1}$  is the multiplicative inverse such that  $cc^{-1} \text{ mod } m = 1$
- Not all integers have multiplicative inverses with respect to a specific modulus  $m$ .

# Multiplicative Inverse

**Fact A.2.** Let  $a \in \mathbb{Z}$  be an integer. We say that integer  $b$  is the multiplicative inverse modulo  $m$  of  $a$ , if  $a \cdot b \equiv 1 \pmod{m}$ ; if it exists, we denote the multiplicative inverse by  $b = a^{-1} \pmod{m}$  (or, when  $m$  is clear from context, simply  $a^{-1}$ ).

An integer  $a$  has multiplicative inverse modulo integer  $m > 0$ , if and only if  $a$  and  $m$  are coprime, namely, they do not have a common divisor (except 1).

## □ Examples:

- $3/5 \pmod{4} = 3 \cdot 5^{-1} \pmod{4} = ?$
- $3/5 \pmod{6} = 3 \cdot 5^{-1} \pmod{6} = ?$

- The algorithm used to compute the inverse is called the Extended Euclidean algorithm (out of scope for this course).

# Modular Exponentiation

- Will be encountered a lot; discrete log-based scheme, RSA, etc.
- We have seen a property to reduce the base, but how about the exponent?
  - Its reduction will be with respect to a different modulus than the one in the original operation.
- Fermat's Little Theorem:

**Theorem 1.1.** *For any integers  $a, b, p \in \mathbb{Z}$ , if  $p$  is a prime and  $p > 0$ , then*

$$\begin{aligned} a^b \mod p &= a^{b \mod (p-1)} \mod p \\ &= (a \mod p)^{b \mod (p-1)} \mod p \end{aligned} \tag{1.9}$$

# Modular Exponentiation

- Examples; Use Fermat's Little theorem (if applicable) to solve the following:
- $13^{32} \bmod 31 = ?$
- $19^{930} \bmod 4 = ?$
- $19^{60} \bmod 7 = ?$
  
- Can we reduce the exponent for non-prime (composite) modulus?
  - We can use Euler's Theorem.

# Euler's Function

- Called also Euler's Totient function. For every integer  $n \geq 1$ , this function computes the number of positive integers that are less than  $n$  and co-prime to  $n$ .
  - gcd below is the greatest common divisor.

$$\phi(n) = |\{i \in \mathbb{N} : i < n \wedge \gcd(i, n) = 1\}|$$

Examples:

$n$	1	2	3	4	5	6	7	8	9	10
$\phi(n)$	1	1	2	2	4	2	6	4	6	4
factors?	none	none	none	$2 \cdot 2$	none	$2 \cdot 3$	none	$2^3$	$3 \cdot 3$	$2 \cdot 5$

# Euler's Function Properties

**Lemma 1.1.** *For any prime  $p > 1$  holds  $\phi(p) = p - 1$ . For prime  $q > 1$  s.t.  $q \neq p$  holds  $\phi(p \cdot q) = (p - 1)(q - 1)$ .*

**Lemma 1.2** (Euler function multiplicative property). *If  $a$  and  $b$  are co-prime positive integers, then  $\phi(a \cdot b) = \phi(a) \cdot \phi(b)$ .*

**Lemma 1.3.** *For any prime  $p$  and integer  $l > 0$  holds  $\phi(p^l) = p^l - p^{l-1}$ .*

**Theorem 1.3** (The fundamental theorem of arithmetic). *Every number  $n > 1$  has a unique representation as a product of powers of distinct primes.*

**Lemma 1.4.** *Let  $n = \prod_{i=1}^n (p_i^{l_i})$ , where  $\{p_i\}$  is a set of distinct primes (all different), and  $l_i$  is a set of positive integers (exponents of the different primes). Then:*

$$\phi(n) = \phi\left(\prod_{i=1}^n (p_i^{l_i})\right) = \prod_{i=1}^n \left(p_i^{l_i} - p_i^{l_i-1}\right) \quad (1.12)$$

# Euler's Theorem

**Theorem 1.2** (Euler's theorem). *For any co-prime integers  $m, n$  holds  $m^{\phi(n)} = 1 \pmod{n}$ . Furthermore, for any integer  $l$  holds:*

$$m^l \pmod{n} = m^{l \pmod{\phi(n)}} \pmod{n} \quad (1.19)$$

## ■ Examples:

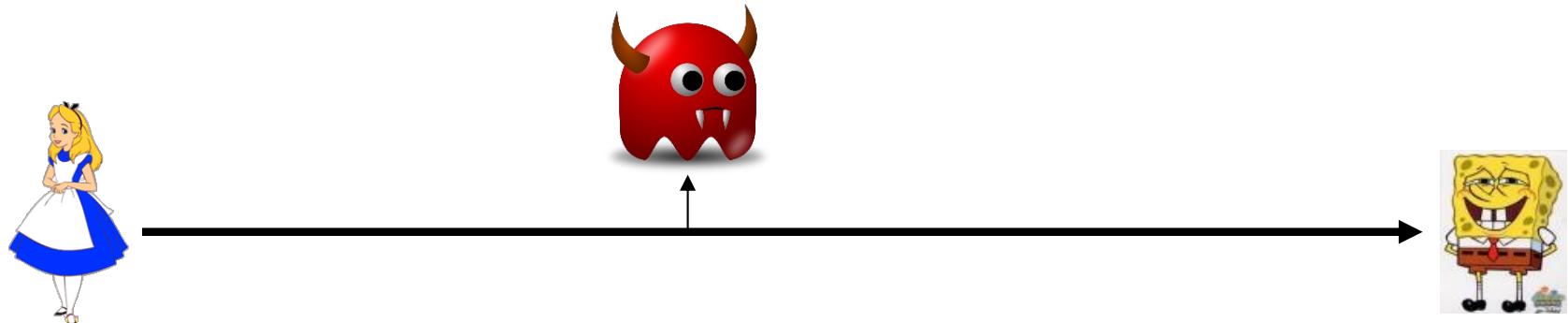
- $13^{31} \pmod{31} = ?$
- $27^{26} \pmod{10} = ?$

# Key Exchange

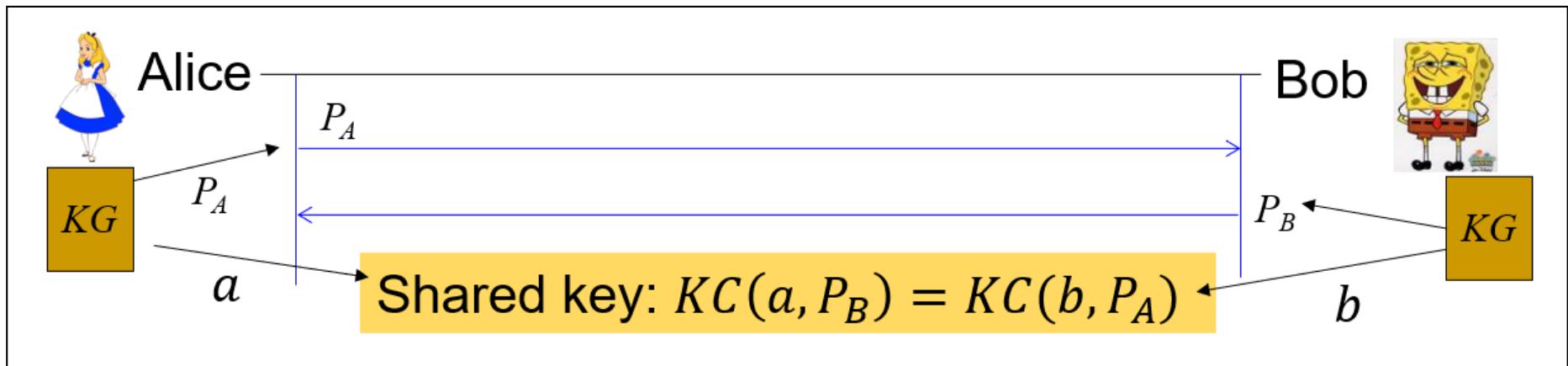
# The Key Exchange Problem

Aka key agreement

- Alice and Bob want to agree on secret (key)
  - Secure against **eavesdropper** adversary
  - Assume no prior shared secrets (key)



# Defining a Key Exchange Protocol



\*KG: Key Generate, KC: Key Compute,  $a$  and  $b$  are secret, while  $P_A$  and  $P_B$  are public

Must satisfy:

- **Correctness**; both parties compute the same shared key,
- and **key indistinguishability**; the key that the two parties establish is indistinguishable from random.

# Discrete Log (DL) Assumption

# --Group Theory Review--

- A group is a pair of  $(G, op)$  is composed of a set of elements  $G$  and an operation  $op$  such that  $G$  is closed under the operation  $op$ , i.e., for any two elements  $a, b \in G$  we have  $a \text{ op } b = c \in G$  , and it satisfies the following requirements:

**Associativity:** for every  $a, b, c \in G$  holds  $(a \cdot b) \cdot c = a \cdot (b \cdot c)$ .

**Identity element:** there exists a (unique) element in  $G$ , which we call the identity element and usually denote by  $1 \in G$ , such that for every element  $a \in G$  holds:  $a = a \cdot 1 = 1 \cdot a$ .

**Inverse:** For each  $a \in G$ , there is an element  $a^{-1} \in G$  such that  $a \cdot a^{-1} = a^{-1} \cdot a = 1$ , where  $1$  is the identity element. For each  $a$ , there is only one such element, which we call the inverse of  $a$  and denote  $a^{-1}$ . (From the identity element property, it follows that the identity element is always its own inverse.)

A commutative group | is a group that also satisfies:

**Commutativity:** for every  $a, b \in G$  holds  $a \cdot b = b \cdot a$ .

# --Group Theory Review--

- We focus on finite commutative groups.
- We will consider Finite Additive Groups:
  - Example:  $(\mathbb{Z}_n, +)$  where  $\mathbb{Z}_n = \{0, 1, 2, \dots, (n - 1)\}$  and the operation is addition modulo  $n$
  - Exercise: show the group above satisfies all properties listed in the previous slide.
- We will consider Finite Multiplicative Groups, mostly, modulo a prime  $p$ :
  - Example:  $(\mathbb{Z}_p^*, \cdot)$  where  $\mathbb{Z}_p^* = \{1, 2, \dots, (p - 1)\}$  and the operation is multiplication modulo  $p$
  - Exercise: show the group above satisfies all properties listed in the previous slide.
- We use the exponentiation notation to denote the repeated application of the group operation.
  - That is,  $a^1 = a$  and  $a^i = a^{i-1} \text{ op } a$  and so on.

# --Cyclic Groups--

**Definition A.4** (Cyclic group, generator and order). *A group  $G$  is cyclic, if there is an element  $g \in G$  such that for every element  $a \in G$ , there is an integer  $i$  such that  $a = g^i$ . Such an element  $g$  is called a generator of  $G$ . The order of  $G$  is the integer  $q > 0$  such that  $g^q = 1$ , where  $g$  is a generator of  $G$  and  $1$  is the unit element of  $G$ .*

Note that  $G = \{g^1, \dots, g^q\} = \{1, g, g^2, \dots, g^{q-1}\}$ , hence, the order  $q$  of a cyclic group  $G$ , is also the number of elements in  $G$ . We also define the order of an element  $a \in G$ ; this is the smallest possible integer  $q > 0$  such that  $a^q = 1$ . In particular, the order of  $a$  is the same as the order of  $G$  if, and only if,  $a$  is a generator of  $g$ .

Examples:

- For prime  $p$ , the additive group  $\mathbb{Z}_p = \{0, 1, \dots, p - 1\}$  is a cyclic group of order  $p$  and every element in this group (except 0) is a generator (because its order is prime).
- For prime  $p$ , the group  $\mathbb{Z}_p^* = \{1, \dots, p-1\}$  is a cyclic multiplicative group. E.g.,  $\mathbb{Z}_7^* = \{1, 2, \dots, 6\}$  is a cyclic group of order 6, a generator for this group is 3 (2, for example, is not a generator. Exercise: verify that!).

# The Discrete Log Problem

- Computing logarithm is quite efficient over the reals
- Discrete log problem: given generator  $g$  and element  $a \in G$ , find  $i$  such that  $a = g^i$ 
  - Verification: exponentiation (efficient algorithm)
- Is discrete-log hard?
  - Some ‘weak’ groups, i.e., where discrete log is **not** hard:
    - $\mathbb{Z}_p^*$  for prime  $p$ , where  $(p - 1)$  has only ‘small’ prime factors
      - Using the Pohlig-Hellman algorithm
    - Check!! Mistakes/trapdoors found, e.g., in OpenSSL'16
  - Other groups studied, considered Ok (‘hard’)
  - **Safe-prime** groups:  $\mathbb{Z}_p^*$  for **safe prime**:  $p = 2q + 1$  for prime  $q$

# Discrete Log Assumption [for safe prime group]

Given PPT adversary A, and  $n$ -bit safe prime  $p$ :

$$\Pr \left[ \begin{array}{l} g \leftarrow \text{Generator}(Z_p^*); \\ x \stackrel{\$}{\leftarrow} Z_p^* \\ A(x) = a | x = g^a \text{ mod } p \end{array} \right] \approx \text{negl}(n)$$

Comments:

1. Similar assumptions for (some) other groups
2. Knowing  $q$ , it is easy to find a generator  $g$
3. Any generator (primitive element) will do



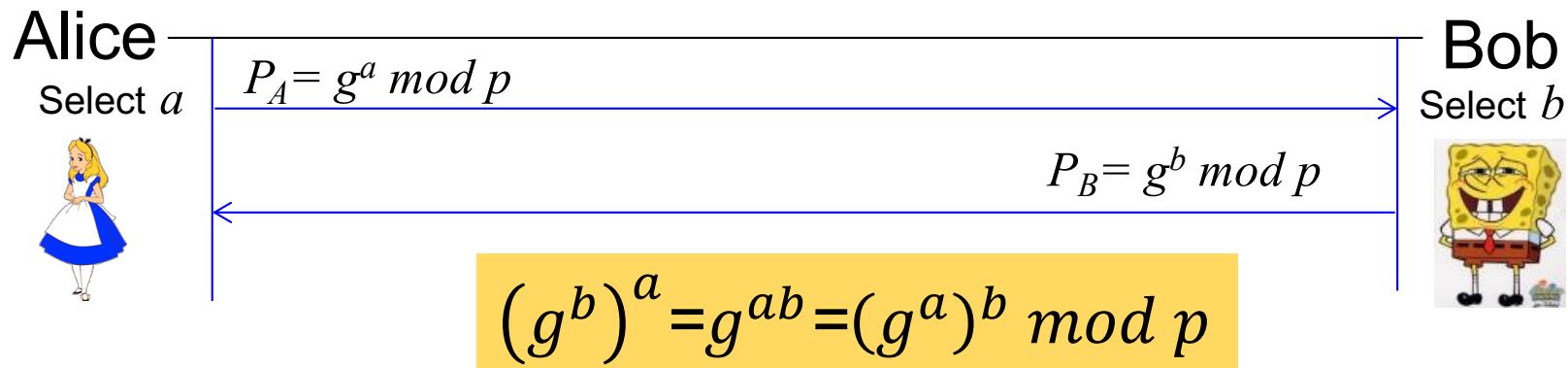
# The Diffie-Hellman (DH) Key Exchange Protocol and The Computational/Decisional Diffie-Hellman Assumptions (CDH/DDH)

---

# Diffie-Hellman [DH] Key Exchange

Using cyclic group  $\mathbb{Z}_p^*$

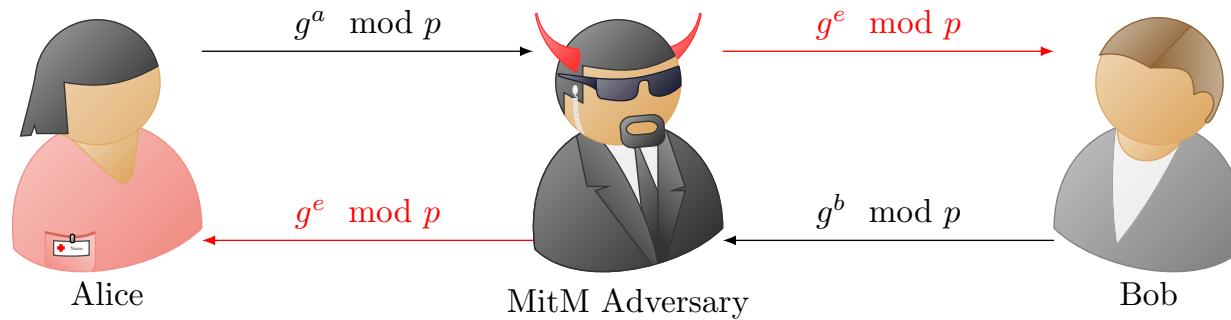
- Simplified Discrete Exponentiation Key Exchange
- Agree on a random safe prime  $p$ 
  - And generator  $g$  for the cyclic group  $\mathbb{Z}_p^*$
- Alice: secret key  $a$ , public key  $P_A = g^a \text{ mod } p$
- Bob: secret key  $b$ , public key  $P_B = g^b \text{ mod } p$
- To set up a shared key :



# Caution: Authenticate Public Keys!

- Diffie-Hellman key exchange is only secure using the authentic public keys
  - Or (equivalently): against eavesdropper
- If Bob simply receives Alice's public key, [DH] is vulnerable to 'Man in the Middle' attack

$$a \xleftarrow{\$} \{1, \dots, p\} \quad e \xleftarrow{\$} \{1, \dots, p\} \quad b \xleftarrow{\$} \{1, \dots, p\}$$



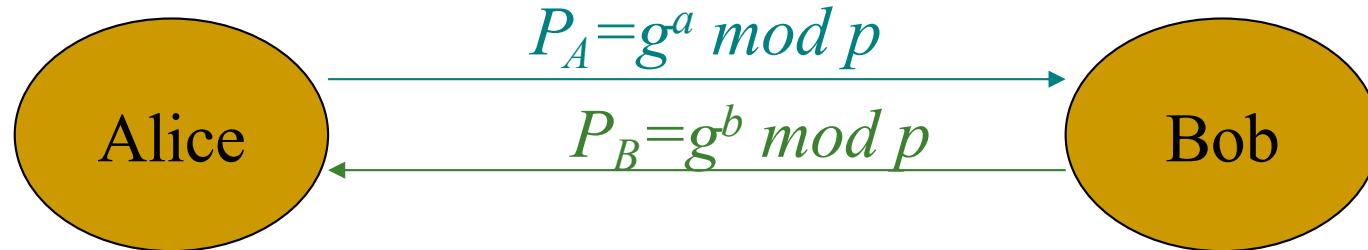
$$(g^e)^a = g^{a \cdot e} \text{ mod } p$$

$$\begin{aligned} (g^a)^e &= g^{a \cdot e} \text{ mod } p, \\ (g^b)^e &= g^{b \cdot e} \text{ mod } p \end{aligned}$$

$$(g^e)^b = g^{b \cdot e} \text{ mod } p$$

# Security of [DH] Key Exchange

- Assume authenticated communication
- Based on Computational Discrete Log Assumption
- But DH requires stronger assumption than Discrete Log:
  - Maybe from  $g^b \text{ mod } p$  and  $g^a \text{ mod } p$ , adversary can compute  $g^{ab} \text{ mod } p$  (without knowing/learning  $a, b$  or  $ab$ )?



# Computational DH (CDH) Assumption [for safe prime group]

Given PPT adversary A:

$$\Pr \left[ \begin{array}{l} (p, q) \leftarrow \text{primes s.t. } p = 2q + 1; \\ g \leftarrow \text{Generator}(\mathbb{Z}_p^*); \\ a, b \leftarrow \{1 \dots p - 1\}; \\ A(g^a, g^b \bmod p) = g^{ab} \bmod p \end{array} \right] \approx \text{negl}(n)$$

Assume CDH holds. Can we use  $g^{ab}$  as key?

Not necessarily; maybe finding some bits of  $g^{ab}$  is easy?

# Using DH securely?

- Consider  $\mathbb{Z}_p^*$  (multiplicative group for (safe) prime  $p$ )
- Can  $g^a, g^b$  expose *something* about  $g^{ab} \text{ mod } p$  ?
- Bad news:
  - Finding (at least) one bit about  $g^{ab} \text{ mod } p$  is easy!
  - (details in textbook if interested)
- So...how to use DH ‘securely’?

# Using DH securely?

- Two options!
  - Option 1: Use DH but with a `stronger' group, e.g., Schnorr's - **not**  $\mathbb{Z}_p^*$  (mod safe-prime  $p$ )
    - The (stronger) **Decisional DH (DDH) Assumption:** adversary can't **distinguish** between  $[g^a, g^b, g^{ab}]$  and  $[g^a, g^b, g^c]$ , for random  $a, b, c$ .
  - Option 2: use DH with safe prime  $p$ ... (*where only CDH holds*) but use a **key derivation function (KDF)** to derive a secure shared key
  - Applied crypto mostly uses KDF... and we too ☺

# Using DH ‘securely’: CDH+KDF

- **Key Derivation Function (KDF)**
  - Two variants: random-keyed and unkeyed (deterministic)
- Randomized - KDF:  $k = KDF_s(g^{ab} \text{ mod } p)$  where  $KDF$  is a key derivation function and  $s$  is public random ('salt')
- Deterministic - crypto-hash:  $k = h(g^{ab} \text{ mod } p)$  where  $h$  is randomness-extracting crypto-hash
  - No need in salt, but **not** provably-secure

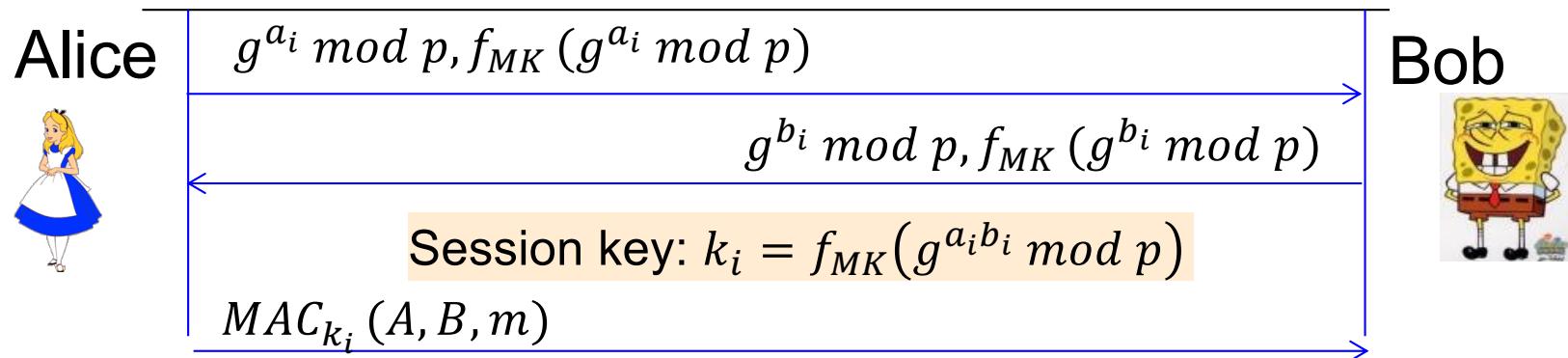
# Resilience to Key Exposure

# Authenticated DH

- Recall: DH is not secure against MitM attacker
- Use DH for **resiliency to key exposure**
  - Do authenticated DH periodically
  - Use derived key for confidentiality, authentication
    - Some protocols use key to authenticate next exchange
  - → Perfect Forward Secrecy (PFS):
    - Confidentiality of session  $i$  is resilient to exposure of all keys, except  $i$ -th session key, after session  $i$  ended

# Authenticated DH: using KDF/PRF<sub>[TLS]</sub>

- Assume  $f$  which is both a PRF and a KDF
- $MK$  is secret +  $f$  is PRF (& MAC) → authentication
  - And, assuming  $MK$  is secret, session keys are secure – even if discrete-log would be easy (quantum computers or math breakthrough)
- Assuming CDH and that  $f$  is **KDF**: secure if MK exposed
  - Since most bits of  $g^{a_i b_i}$  are secret
  - Against eavesdropping or if MK is exposed only after session ends.
  - Perfect forward secrecy (PFS) !



# Resilience to Key Exposure: Recover Security

- ❑ The previous DH protocol does not achieve recover security, why?
  - ❑ Exposing ML makes all future session vulnerable to MitM (this adversary can authenticate any public key he wants to the other party).
- There is another version, called Ratchet DH, that achieves perfect recover security.
  - ❑ Will not be covered in this class.

# Covered Material From the Textbook

- Appendix A.2
- Chapter 6:
  - Sections 6.1 (except 6.1.8.3),
  - Section 6.2 (6.2.1 and 6.2.3 are optional reading),
  - and Section 6.3 (except 6.3.2)

# Thank You!

