CS 442
Introduction to Cryptography

Lecture 21: Public-Key Encryption

Instructor: Aarushi Goel

Spring 2026

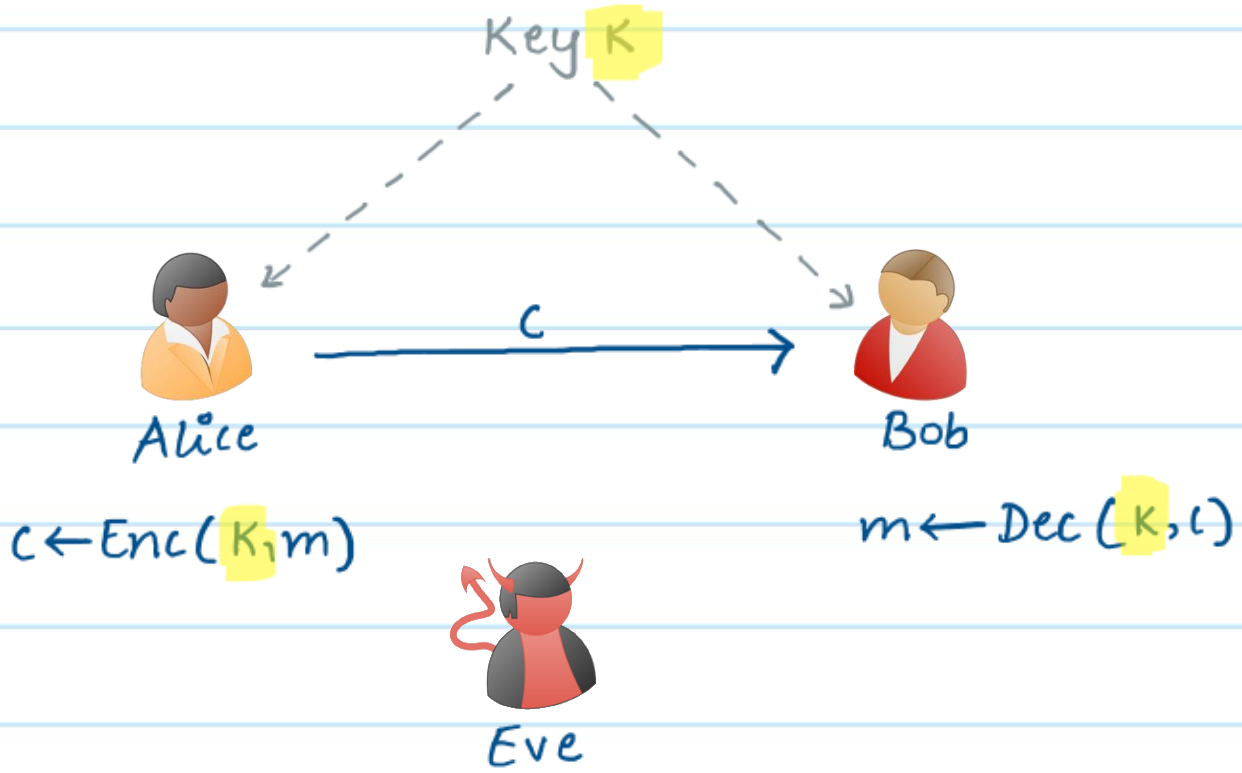
Agenda

* Public-Key Encryption (PKE)
* One-message secure PKE
* Applications of PKE
* ElGammal Encryption
* Multi-message secure PKE

* HW4 is due on April 19.

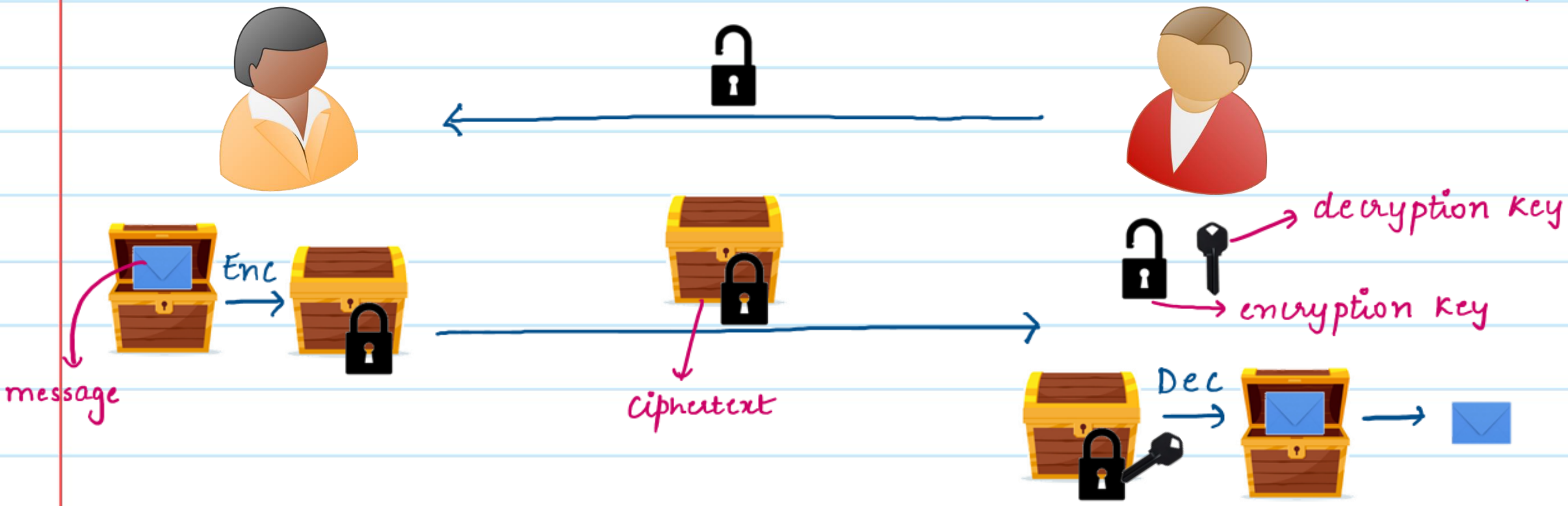
Secret-Key Encryption (SKE)

* So far, when discussing encryption schemes, we assumed that Alice and Bob use the same *secret-key* for both encryption and decryption.
* Such schemes are called secret-key encryption or symmetric-key encryption.



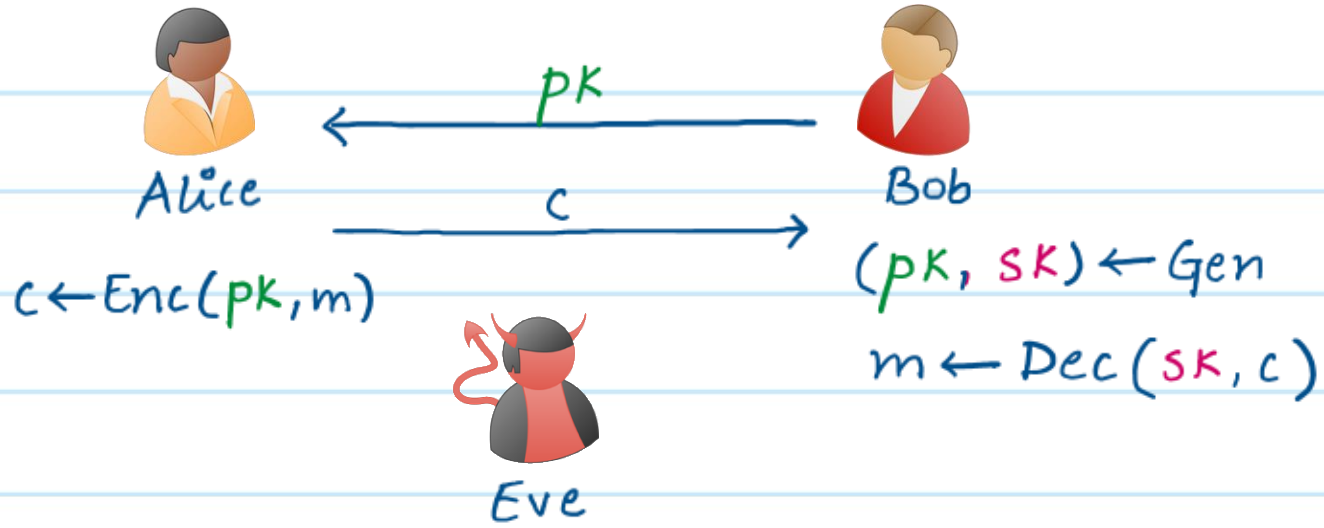
Public-Key Encryption (PKE)

* Public-key encryption is a different type of encryption, where the key used for encryption is different from the key used for decryption.



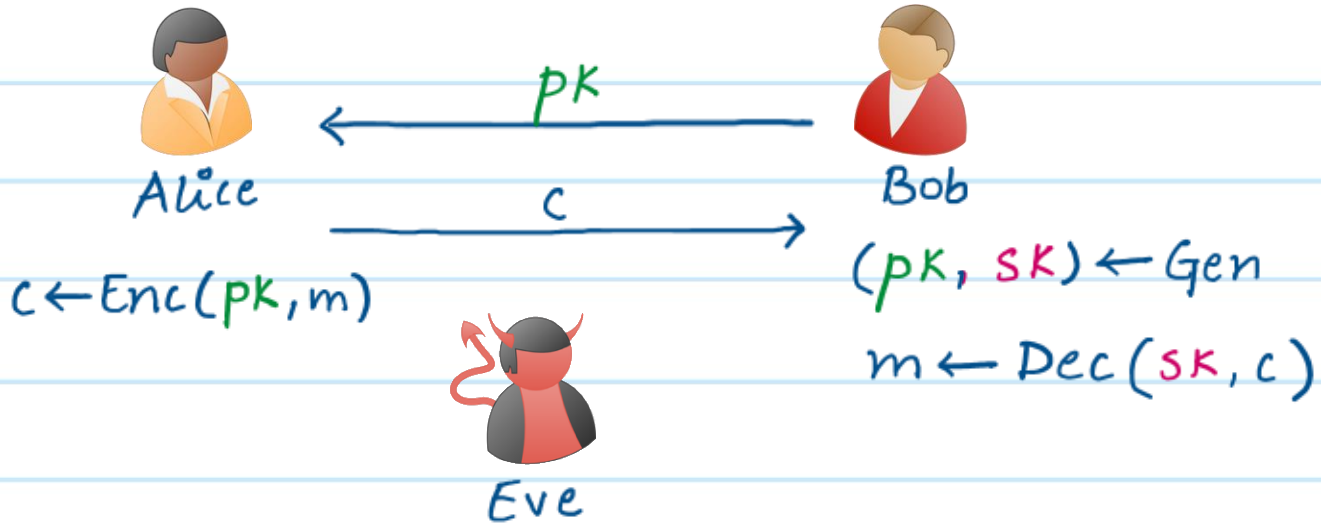
Public-Key Encryption

* Encryption Key = public Key (pk)
* Decryption Key = secret Key (sk)
* If Bob wants to receive a private message from Alice, he can generate a (pk, sk) pair and send pk to Alice.
* Alice uses pk to encrypt the message
* Bob uses sk to decrypt the message



Public-Key Encryption

* We don't need to hide the public key. It can be sent over a public communication channel. We only care about hiding the secret-key.
* Public-key does not reveal any information about the secret-key.
* No need for a key-exchange protocol.



Public-Key Encryption in the Real-World

* 

End-to-end encrypted messaging

* 

HTTPS (hypertext transfer protocol secure)

* 

VPN (Virtual Private Network)

One-Message Secure Public-Key Encryption

Definition: A one-message secure PKE scheme with message space $\mathcal{M}$, key space $\mathcal{K}$, ciphertext space $\mathcal{C}$, comprises of the following algorithms:

* $\text{Gen} \rightarrow pk, sk$: This algorithm samples a pair $(pk, sk) \in \mathcal{K}$.
* $\text{Enc}(pk, m) \rightarrow c$: On input a public key pk & a message $m \in \mathcal{M}$, it outputs ciphertext $c \in \mathcal{C}$.
* $\text{Dec}(sk, c) \rightarrow m$: On input a secret key sk & a ciphertext $c \in \mathcal{C}$, it outputs message $m \in \mathcal{M}$.

These algorithms must satisfy the following:

→ **Correctness:** For $(pk, sk) \leftarrow \text{Gen}$, $\forall m \in \mathcal{M}$, the following holds:

$$Pr[\text{Dec}(sk, \text{Enc}(pk, m)) = m] = 1$$

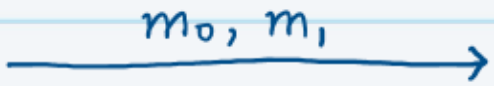
→ **Security:** For all PPT adversaries A, $\exists$ a negligible function $\nu(.)$, such that:

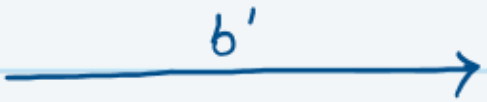
$$Pr\left[\begin{array}{l|l} (pk, sk) \leftarrow \text{Gen}(1^n), & \\ (m_0, m_1) \leftarrow A(pk), & A(pk, \text{Enc}(sk, m_b)) = b \\ b \xleftarrow{\$} \{0,1\} & \end{array}\right] \leq \frac{1}{2} + \nu(n).$$

Another Interpretation of One-message Security

Adversary

Challenger





$$Pr[b = b'] \leq \frac{1}{2} + \nu(n).$$

OR equivalently: $\{pk, Enc(pk, m_0)\} \approx_c \{pk, Enc(pk, m_1)\}$

$\forall m_0, m_1 \in \mathcal{M}.$

Diffie - Hellman Assumptions

Let's sample a cyclic group $(G, \cdot)$ of order q with generator g.

* Decisional Diffie-Hellman (DDH) Assumption: Sample $x, y, z \xleftarrow{\$} \mathbb{Z}_q$, compute $h_1 = g^x$, $h_2 = g^y$. Given (G, q, g, h_1, h_2), it is computationally hard to distinguish between g^{xy} and g^z.

In other words,

$$(G, g, q, h_1, h_2, g^{xy}) \overset{c}{\approx} (G, g, q, h_1, h_2, g^z)$$

$\overset{c}{\approx}$: computationally indistinguishable.

ElGamal Encryption

Public-key encryption for encrypting $m \in \mathbb{G}$.

* Gen: Sample a cyclic group $(\mathbb{G}, \cdot)$ of order q with generator g.
  Sample $x \xleftarrow{\$} \mathbb{Z}_q$. $pK = (\mathbb{G}, g, q, X = g^x)$, $sK = (\mathbb{G}, g, q, x)$

* Enc(pK, m): Sample $y \xleftarrow{\$} \mathbb{Z}_q$. $c = (Y = g^y, D = m \cdot X^y)$

* Dec(sK, c): Parse $c = (Y, D)$. Compute $m = D \cdot Y^{-x}$.

* **Correctness:** We need to show $Pr[Dec(sk, Enc(pk, m)) = m] = 1$.

$Enc(pk, m) = (Y = g^y, D = m \cdot X^y)$

$Dec(sk, c)$: $D \cdot Y^{-x} = m \cdot X^y \cdot Y^{-x} = m \cdot g^{xy} \cdot g^{-xy} = m$

* **One-message Security:** We need to show:

$\forall m_0, m_1 \in \mathcal{M}. : \{pk, Enc(pk, m_0)\} \approx_c \{pk, Enc(pk, m_1)\}$

$\{\mathbb{G}, g, q, g^x, g^y, m_0 \cdot g^{xy}\}$ $\{\mathbb{G}, g, q, g^x, g^y, m_1 \cdot g^{xy}\}$

Lets use hybrid arguments.

H_0: $\{\mathbb{G}, g, q, g^x, g^y, m_0 \cdot g^{xy}\}$

H_1: $\{\mathbb{G}, g, q, g^x, g^y, m_0 \cdot g^z\}$ $z \xleftarrow{\$} \mathbb{Z}_q$ ← random

H_2: $\{\mathbb{G}, g, q, g^x, g^y, m_1 \cdot g^z\}$

H_3: $\{\mathbb{G}, g, q, g^x, g^y, m_1 \cdot g^{xy}\}$

* $H_0 \approx_c H_1$: follows from the DDH assumption
* $H_1 \equiv H_2$: perfectly indistinguishable (like one-time pad)
* $H_2 \approx_c H_3$: follows from the DDH assumption

From hybrid lemma, it follows that $H_0 \approx_c H_3$.

PKE Cannot be Deterministic

* The encryption algorithm in a PKE scheme must be a randomized algorithm, in other words, it must take an additional random value as input.

WHY?

* If the encryption algorithm is deterministic, then given a pk, there will be a unique ciphertext corresponding to each message.
* A PPT adversary in this case has pk and can encrypt both m_0 and m_1 on his own, compare these to the ciphertext sent by the challenger to deduce whether its an encryption of m_0 or m_1 in the one-message security game.

Perfectly Secret PKE is Impossible.

* No PKE scheme can be secure against a computationally unbounded adversary.

WHY?

* We have already established that the encryption algorithm in a PKE scheme must be randomized.
* An unbounded adversary can easily win the one-message security game by computing all possible encryptions of m_0 and m_1, using all possible random values. The adversary can then check whether the ciphertext received from the challenger corresponds to one of the encryptions of m_0 or m_1.

Multi-Message Security

Adversary

Challenger

pk ←

$(pk, sk) \leftarrow Gen(1^n)$

m_0^1, m_1^1 →

$b \xleftarrow{\$} \{0,1\}$

← c^1

$c^1 = Enc(pk, m_b^1)$

m_0^2, m_1^2 →

$c^2 = Enc(pk, m_b^2)$

← c^2

⋮ ⋮

m_0^k, m_1^k →

$c^k = Enc(pk, m_b^k)$

← c^k

b' →

$$Pr[b = b'] \leq \frac{1}{2} + \nu(n)$$

Multi-Message Security

* Every one-message secure PKE is also multi-message secure.

WHY?