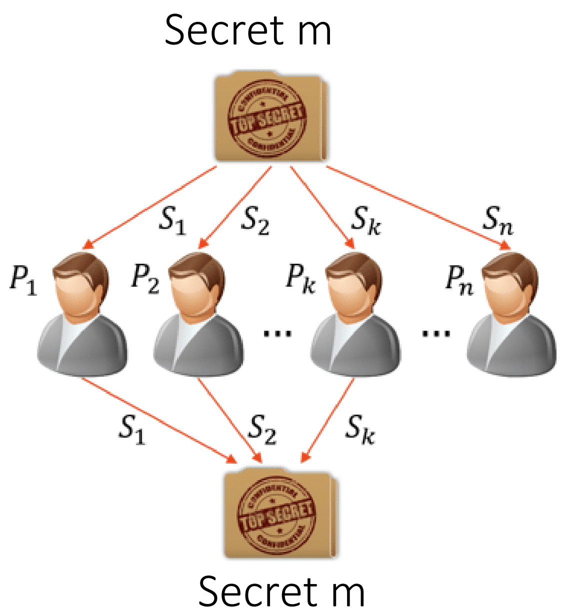
CSE 539: Applied Cryptography
Lec 4: Security Definition

Ni Trieu (ASU)

Recap: Secret Sharing

- m – Secret to be shared
- P – Set of participants

=> A qualified subsets of can reconstruct m



- Formally, secret sharing scheme allows share a secret m among n parties such that for a fixed number t < n, the following conditions are satisfied.
  - If < t parties get together, then they get no additional information about the secret.
  - If > t parties get together, then they can correctly reconstruct the secret

Project: Secret Sharing

- Verifiable Secret Sharing
  - https://docs.google.com/spreadsheets/d/1iYJ0UNXLk5_1EMaMClZdwPr4hjuJbdlu5uPYJuL9uBo/edit?usp=sharing
  - https://link.springer.com/chapter/10.1007/3-540-68339-9_17
  - https://ieeexplore.ieee.org/abstract/document/4568297?casa_token=ORLzB8c9LPsAAAAA:vsQtCX4nBzLU9d51nc-WWEUxwvOJp2jyBEqEXZ9fArV5D5iUS2toJByMvGY53gEmPVOPrjgV
  - https://scholar.google.com/scholar?hl=en&as_sdt=0%2C3&q=Verifiable+Secret+Sharing&btnG=

One-time Pad

$\underline{\text{KeyGen:}}$
$k \leftarrow \{0,1\}^\lambda$
return k

$\underline{\text{Enc}(k, m \in \{0,1\}^\lambda):}$
return $k \oplus m$

$\underline{\text{Dec}(k, c \in \{0,1\}^\lambda):}$
return $k \oplus c$

Provable Security

- Consider the attacker as a calling program to the following subroutine

$\underline{\textsc{ctxt}(m \in \Sigma.\mathcal{M}):}$

$\quad k \leftarrow \Sigma.\text{KeyGen}$

$\quad c := \Sigma.\text{Enc}(k, m)$

$\quad \text{return } c$

Provable Security

- Consider the attacker as a calling program to the following subroutine

$$\begin{array}{l}\underline{\textsc{ctxt}(m \in \Sigma.\mathcal{M}):}\\ \quad k \leftarrow \Sigma.\mathsf{KeyGen}\\ \quad c := \Sigma.\mathsf{Enc}(k, m)\\ \quad \text{return } c\end{array}$$

- "Real-vs-Random" Style of Security Definition

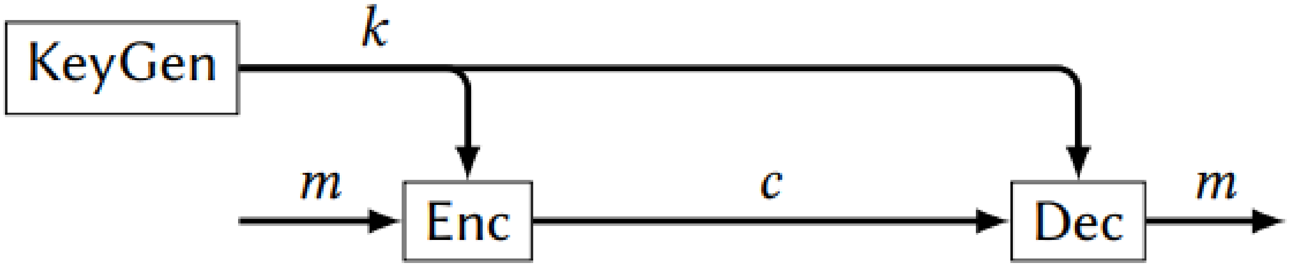
$$\begin{array}{l}\underline{\textsc{ctxt}(m \in \Sigma.\mathcal{M}):}\\ \quad k \leftarrow \Sigma.\mathsf{KeyGen}\\ \quad c := \Sigma.\mathsf{Enc}(k, m)\\ \quad \text{return } c\end{array} \quad \text{vs.} \quad \begin{array}{l}\underline{\textsc{ctxt}(m \in \Sigma.\mathcal{M}):}\\ \quad c \leftarrow \Sigma.\mathcal{C}\\ \quad \text{return } c\end{array}$$

  - An encryption scheme is a good one if the two implementations of ctxt induce identical behavior in every calling program (Uniform ctxs)

Provable Security: One-time Pad (Example)

KeyGen:	Enc($k, m \in \{0,1\}^\lambda$):	Dec($k, c \in \{0,1\}^\lambda$):
$k \leftarrow \{0,1\}^\lambda$	return $k \oplus m$	return $k \oplus c$
return k		

Encryption Basics & Terminology



- “Symmetric” Encryption scheme:
  - KeyGen() -> k
  - Enc(k,m) -> c
  - Dec(k,c) -> m
- Requirement: $Dec(k, Enc(k, m)) = m$

Provable Security

- "Real-vs-Random" Style of Security Definition

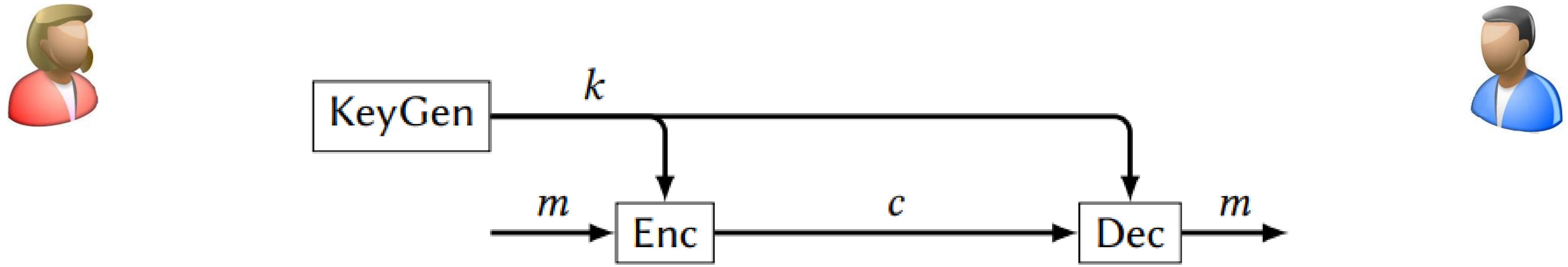
$$\begin{array}{|l|}\hline \textsc{ctxt}(m \in \Sigma.\mathcal{M}): \\ \hline \quad k \leftarrow \Sigma.\text{KeyGen} \\ \quad c := \Sigma.\text{Enc}(k, m) \\ \quad \text{return } c \\ \hline \end{array} \quad \text{vs.} \quad \begin{array}{|l|}\hline \textsc{ctxt}(m \in \Sigma.\mathcal{M}): \\ \hline \quad c \leftarrow \Sigma.\mathcal{C} \\ \quad \text{return } c \\ \hline \end{array}$$

- An encryption scheme is a good one if the two implementations of ctxt induce identical behavior in every calling program (Uniform ctxs)

$\Rightarrow$Security definitions for encryption capture the case where a key is used to encrypt only one plaintext.

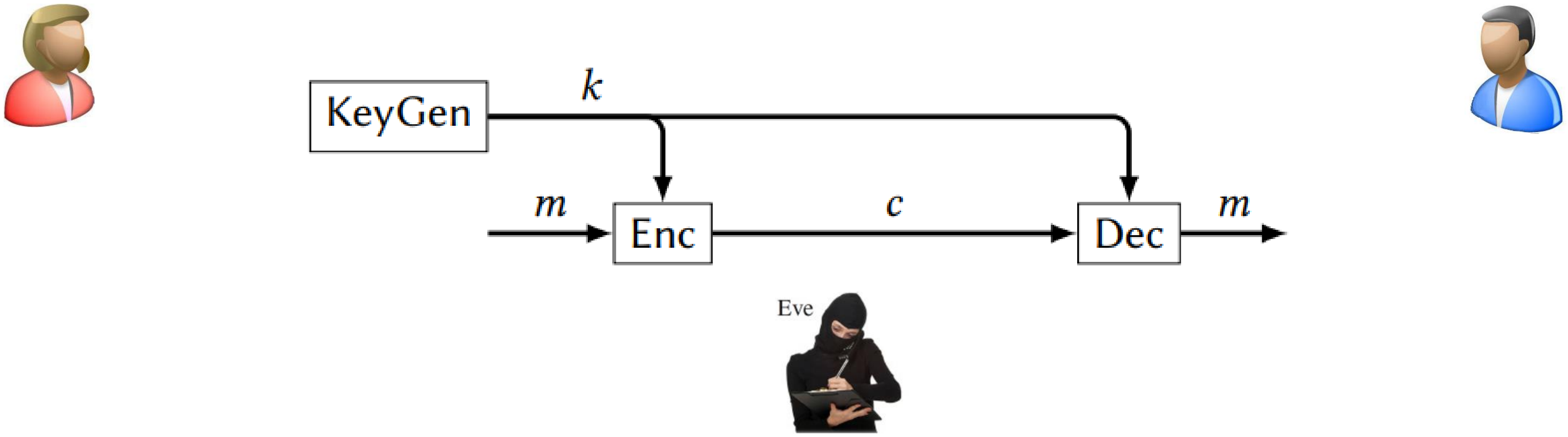
$\Rightarrow$It would be more useful to have an encryption scheme that allows many plaintexts to be encrypted under the same key

Encryption Basics & Terminology



- What if Alice sends Enc(k,m) twice?

Encryption Basics & Terminology



- What if Alice sends c=Enc(k,m) twice?
  - Eve can observe it

Security Against Chosen Plaintext Attacks (CPA)

- CPA Security Definition:

Let Σ be an encryption scheme. We say that Σ has ***security against chosen-plaintext attacks (CPA security)*** *if* $\mathcal{L}^{\Sigma}_{\text{cpa-L}} \approx \mathcal{L}^{\Sigma}_{\text{cpa-R}}$, *where:*

$\mathcal{L}^{\Sigma}_{\text{cpa-L}}$	$\mathcal{L}^{\Sigma}_{\text{cpa-R}}$
$k \leftarrow \Sigma.\text{KeyGen}$	$k \leftarrow \Sigma.\text{KeyGen}$
EAVESDROP($m_L, m_R \in \Sigma.\mathcal{M}$):	EAVESDROP($m_L, m_R \in \Sigma.\mathcal{M}$):
$c := \Sigma.\text{Enc}(k, m_L)$	$c := \Sigma.\text{Enc}(k, m_R)$
return c	return c

- CPA security is often called “IND-CPA” security, meaning “indistinguishability of ciphertexts under chosen-plaintext attack.”
  - “CPA is a type of cryptanalysis where an attacker can choose some or all of the plaintext messages that are encrypted with a secret key. By analyzing the resulting ciphertexts, the attacker may be able to recover the key or some information about the plaintext.”

https://www.linkedin.com/advice/0/what-common-types-methods-chosen-plaintext-attacks-how#:~:text=Chosen%2Dplaintext%20attacks%20are%20a,some%20information%20about%20the%20plaintext.

CPA-Security

- Deterministic encryption can never be CPA-secure
- Why?

Quiz Sample:

- Is 2OTP CPA-Secure?

2OTP($m \in \{0,1\}^\lambda$):

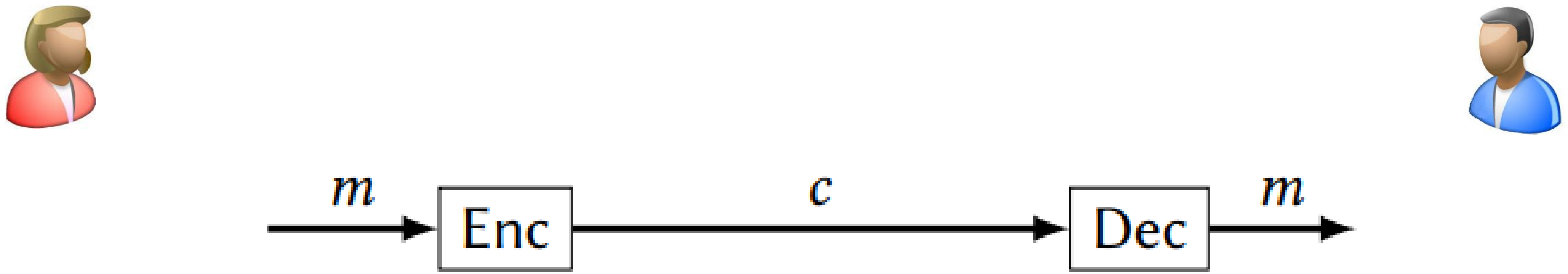
$k_1 \leftarrow \{0,1\}^\lambda$ // Choose a random key k_1 from $\{0,1\}^\lambda$

$k_2 \leftarrow \{0,1\}^\lambda$ // Choose a random key k_2 from $\{0,1\}^\lambda$

$c := k_2 \oplus (k_1 \oplus m)$

return c

Security Against Chosen Plaintext Attacks (CPA)



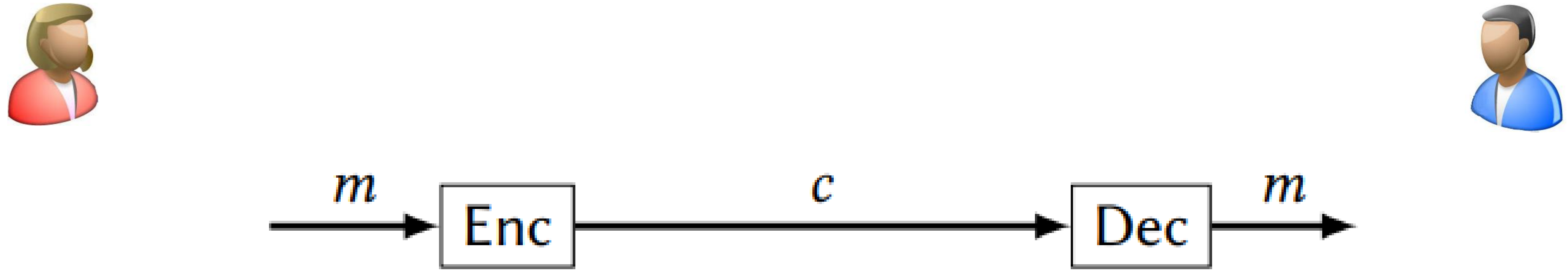
Security Discussion

CPA: secure if Adversary chooses plaintext

- Cares about m ---> c direction

Security Against Chosen Ciphertext Attacks (CCA)

What if the adversary changes c?



Security Against Chosen Ciphertext Attacks (CCA)

- CCA Security Definition:
  - Goal: Can't learn what inside ciphertext c, even if you can decrypt anything other than c

Security Against Chosen Ciphertext Attacks (CCA)

Let Σ be an encryption scheme. We say that Σ has ***security against chosen-ciphertext attacks (CCA security)*** *if $\mathcal{L}^{\Sigma}_{\text{cca-L}} \approx \mathcal{L}^{\Sigma}_{\text{cca-R}}$, where:*

$\mathcal{L}^{\Sigma}_{\text{cca-L}}$	$\mathcal{L}^{\Sigma}_{\text{cca-R}}$
$k \leftarrow \Sigma.\text{KeyGen}$	$k \leftarrow \Sigma.\text{KeyGen}$
$\mathcal{S} := \emptyset$	$\mathcal{S} := \emptyset$
$\underline{\text{EAVESDROP}(m_L, m_R \in \Sigma.\mathcal{M}):}$	$\underline{\text{EAVESDROP}(m_L, m_R \in \Sigma.\mathcal{M}):}$
if $\lvert m_L \rvert \neq \lvert m_R \rvert$ return err	if $\lvert m_L \rvert \neq \lvert m_R \rvert$ return err
$c := \Sigma.\text{Enc}(k, m_L)$	$c := \Sigma.\text{Enc}(k, m_R)$
$\mathcal{S} := \mathcal{S} \cup \{c\}$	$\mathcal{S} := \mathcal{S} \cup \{c\}$
return c	return c
$\underline{\text{DECRYPT}(c \in \Sigma.\mathcal{C}):}$	$\underline{\text{DECRYPT}(c \in \Sigma.\mathcal{C}):}$
if $c \in \mathcal{S}$ return err	if $c \in \mathcal{S}$ return err
return $\Sigma.\text{Dec}(k, c)$	return $\Sigma.\text{Dec}(k, c)$

Security Discussion

CPA: secure if Adversary chooses plaintext

- Cares about m ---> c direction

CCA: secure if Adversary gets all of Dec(ctxt)

- Cares about c ---> m direction

Security Discussion

CPA: secure if Adversary chooses plaintext

- Cares about m ---> c direction

CCA: secure if Adversary gets all of Dec(ctxt)

- Cares about c ---> m direction
- In 1998, Daniel Bleichenbacher demonstrated a devastating attack against early versions of the SSL protocol. By presenting millions of carefully crafted ciphertexts to a webserver, an attacker could eventually recover arbitrary SSL session keys.

https://blog.cryptographyengineering.com/2016/03/01/attack-of-week-drown/