CSE 3400 - Introduction to Computer & Network Security
(aka: Introduction to Cybersecurity)

Lecture 4
Encryption – Part III
(and Pseudo-randomness)

Ghada Almashaqbeh
UConn

From Textbook Slides by Prof. Amir Herzberg
UConn

Outline

- Block ciphers.
- Pseudorandom permutations (PRPs).
- Defining security of encryption.
- Encryption modes.
- Concluding remarks.

Block Ciphers

- A pair of algorithms E_k and D_k (encrypt and decrypt with key k) with domain and range of $\{0,1\}^n$
  - Encrypt and decrypt data in blocks each of which is of size n bits.
- Conventional correctness requirement: $m = D_k(E_k(m))$
- Several schemes used in practice including DES and AES.
  - No security proofs, just resistance to cryptanalysis.
  - DES is insecure for short keys, replaced by AES.
- Security requirement of block ciphers is to be a pair of Pseudorandom Permutations (PRP).

So what is a Random Permutation?

And what is a PRP?

What is a random **permutation** ρ ?

- Random permutation ρ over finite domain D, usually: $\{0,1\}^m$
- How can we select a random permutation ρ ?
- Let $D = \{x_1, x_2, \dots, x_n\}$
- For $i = 1, \dots, n$:
  - $\rho(x_i) \overset{\$}{\leftarrow} D - \{\rho(x_1), \rho(x_2), \dots, \rho(x_{i-1})\}$
- Examples:

Domain D $\{0,1\}^2$

	$\rho(\ )$
00	10
01	11
10	00
11	01

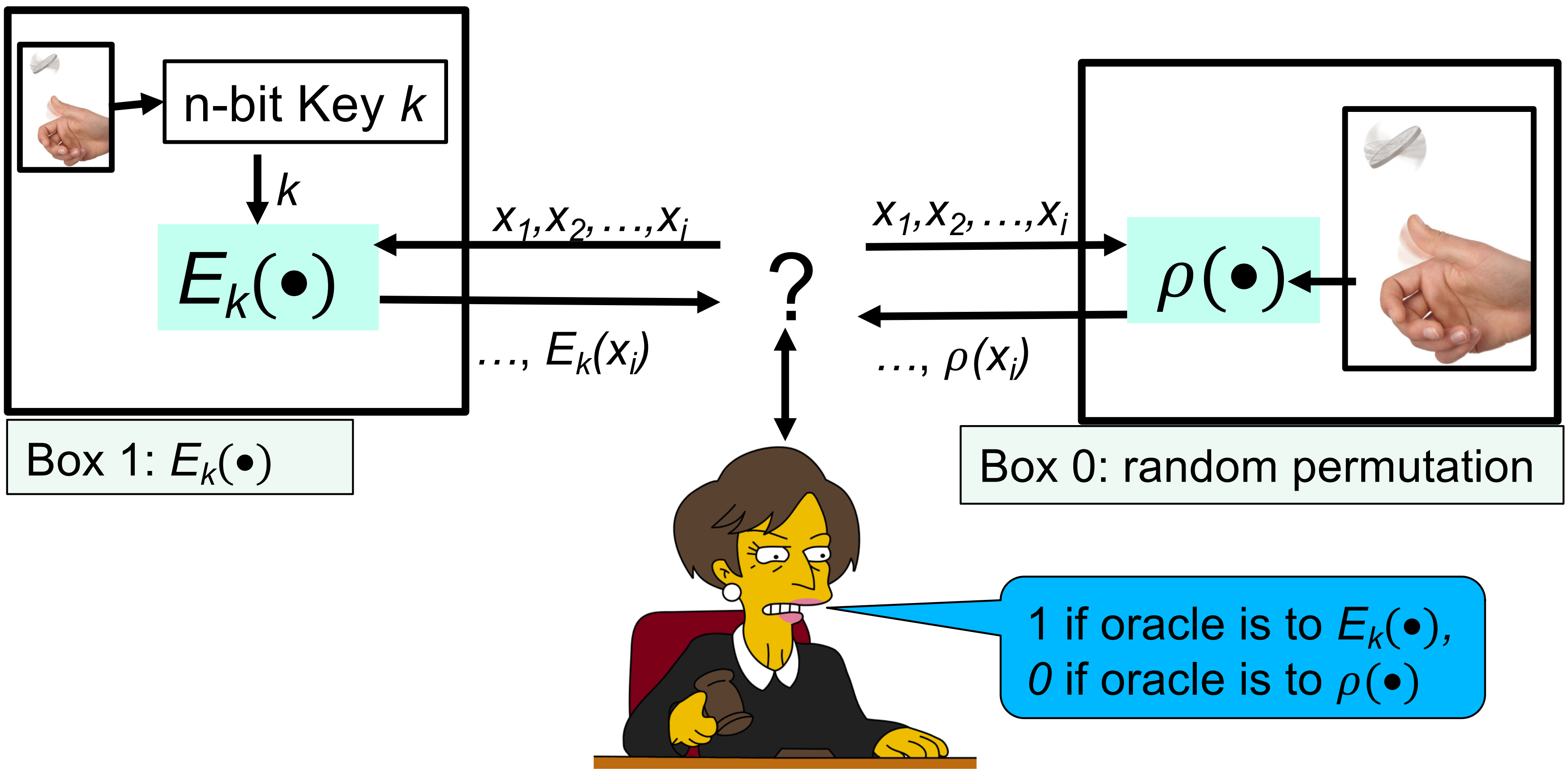
Domain D $\{0,1\}^2$

	$\rho(\ )$
00	00
01	01
10	10
11	11

Pseudo-Random Permutation (PRP)

and their Indistinguishabity Test

- E is a PRP over domain D, if no distinguisher D:
  - Outputs 1 (signaling PRP) given oracle to $E_k(\bullet)$, for random (n-bits) key k, and
  - Outputs 0 (signaling random) given oracle to $\rho(\bullet)$, a <u>random</u> permutation (over D)



Pseudo-Random Permutation (PRP)

- Pseudo-Random Permutation (PRP) $E_k(\cdot)$
  - Cannot be distinguished from truly random permutation over same domain
  - Against efficient adversaries (PPT), allowing negligible advantage
  - Yet practical, even efficient

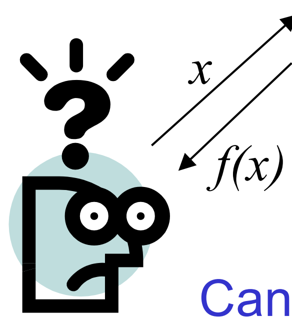
Definition 2.9. *A* pseudo-random Permutation *(*PRP*) is a polynomial-time computable function $E_k(x) : \{0,1\}^* \times D \to D \in PPT$ s.t. for all PPT algorithms $\mathcal{A}$, $\varepsilon^{PRP}_{\mathcal{A},E}(n) \in NEGL(n)$, i.e., is negligible, where the* advantage *$\varepsilon^{PRP}_{\mathcal{A},E}(n)$ of the PRP E against adversary $\mathcal{A}$ is defined as:*

$$\varepsilon^{PRP}_{\mathcal{A},E}(n) \equiv \Pr_{k \stackrel{\$}{\leftarrow} \{0,1\}^n} \left[\mathcal{A}^{E_k}(1^n)\right] - \Pr_{\rho \stackrel{\$}{\leftarrow} Perm(D)} \left[\mathcal{A}^{\rho}(1^n)\right] \tag{2.16}$$

The probabilities are taken over random coin tosses of $\mathcal{A}$, and random choices of the key $k \stackrel{\$}{\leftarrow} \{0,1\}^n$ and of the function $\rho \stackrel{\$}{\leftarrow} Perm(D)$.

Block Cipher: Invertible PRP (E, D)

- Common definition for **block cipher**
- Invertible Pseudo-Random Permutation (PRP):
  - A pair of PRPs (E,D), s.t.: $m=D_k(E_k(m))$
  - And (E,D) is indistinguishable from (π, π^{-1})
    - where π is a random permutation
  - Note: it is deterministic, stateless → not secure encryption!
    - But used to construct encryption (soon)



$f \leftarrow E_k(.)$ or $D_k(.)$ [for random k]

or

$f \leftarrow$ *random n-bit permutation or its inverse*

Can't tell if (f, f^{-1}) is a random permutation+inverse, or *(E, D)* with a random key!

Example of a Block Cipher Security and Correctness

- On the whiteboard.

Constructing block-cipher, PRP

- Focus: constructions from a PRF $f_k(\cdot)$
  - PRFs seem easier to design (less restrictions)
- First: 'plain' PRP $E_k(\cdot)$ (not a block cipher)
- What is the simplest construction to try? $E_k(x)$=$\underline{f_k(x)\quad}$

Lemma 2.4 (The PRP/PRF Switching Lemma). *Let E be a polynomial-time computable function $E_k(x) : \{0,1\}^* \times D \to D \in PPT$, and let $\mathcal{A}$ be a PPT adversary, which is limited to at most q oracle queries. Then:*

$$\left|\varepsilon^{PRF}_{\mathcal{A},E}(n) - \varepsilon^{PRP}_{\mathcal{A},E}(n)\right| < \frac{q^2}{2 \cdot |D|} \tag{2.17}$$

Where the advantage functions are as defined in Equation 2.16 and Equation 2.13.

In particular, if the size of the domain D is exponential in the security parameter n (the length of key and of the input to $\mathcal{A}$), e.g., $D = \{0,1\}^n$, then $\varepsilon^{PRF}_{\mathcal{A},E}(n) - \varepsilon^{PRP}_{\mathcal{A},E}(n) \in NEGL(n)$. In this case, E is a PRP over D, if and only if it is a PRF over D.

Constructing block-cipher, PRP

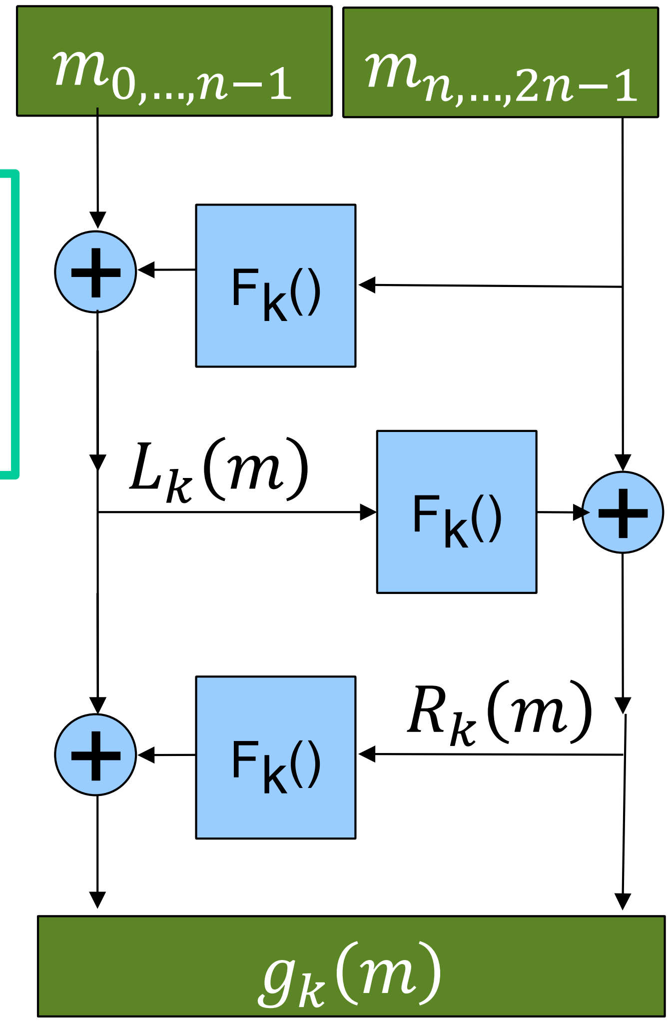
- Focus: constructions from a PRF $f_k(\cdot)$
  - PRFs seem easier to design (less restrictions)
- Before: ‘plain’ PRP $E_k(\cdot)$ (not a block cipher)
- Now: construct block cipher (invertible PRP) E_k, D_k
- Challenge: making it invertible…
- Solution: The Feistel Construction

The Feistel Block-cipher Construction

- Turn PRF F_k into a block cipher
  - Three 'rounds' suffice [LR88]

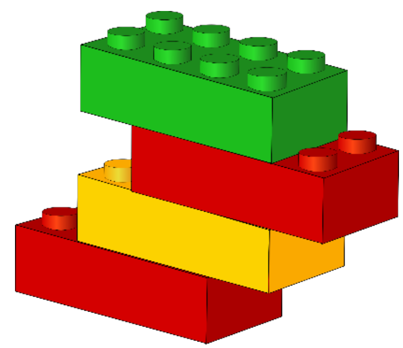
$$
\begin{aligned}
L_k(m) &= m_{0,\ldots,n-1} \oplus F_k(m_{n,\ldots,2n-1}) \\
R_k(m) &= F_k(L_k(m)) \oplus m_{n,\ldots,2n-1} \\
g_k(m) &= L_k(m) \oplus F_k(R_k(m)) \mathbin{+\!\!+} R_k(m)
\end{aligned}
$$

- Used in DES (but not in AES)
  - With 16 'rounds'



Crypto Building-Blocks Principle

- Design and focus cryptanalysis efforts on few basic functions: simple, easy to test, replaceable
- Construct schemes from basic functions
  - Provably secure constructions: attack on scheme ➔ attack on function
  - Allows replacing broken/suspect functions
  - Allows upgrading to more secure/efficient function



- E.g., encryption from block cipher (or PRG/PRF/PRP)
  - Block-cipher, PRG,PRF,PRP: deterministic, stateless, FIL (Fixed-Input-Length)
  - Encryption: randomized/stateful, VIL (Variable-Input-Length)

Why standardize block ciphers, and not encryption?

- Crypto building blocks principle, rephrased:
  design, cryptanalyze simple function,
  use function to construct more complex scheme
- Design, cryptanalyze PRF; use it to build block cipher; and block cipher to construct cryptosystem
  - Attack on cryptosystem ➔ attack on block cipher, PRF
  - Design (FIL, deterministic, stateless) PRF, construct (VIL, randomized/stateful) cryptosystem
  - Easier to design and to combine:
    - Given two PRFs F, F', let $F''_{k,k'}(x)=F_k(x)\oplus F'_{k'}(x)$
      - If <u>either</u> F <u>or</u> F' is a secure PRF ➔ F'' is secure PRF
      - This is a <u>robust combiner</u> for PRFs (block ciphers: also not hard)
  - Next: Feistel construction of Block-cipher from PRF!

We defined security for PRG, PRF and PRP. Block cipher too (informally).

But…

what about security of encryption??

A bit tricky, in fact.

Defining Secure Encryption

- Attacker capabilities:
  - Computational limitations? ➔ PPT
  - Ciphertext only (CTO), Known / chosen plaintext attack (KPA/CPA), Chosen ciphertext (CCA)?
- What's a successful attack?
  - Key recovery ?
    - May be impossible yet weak cipher…
  - (Full) Message recovery?
    - What of partial exposure, e.g., $m \in$ {"Advance", "Retreat"}
  - <u>Prudent</u>: attacker 'wins' for <u>any</u> info on plaintext

Conservative Design Principle

- When designing, evaluating a cryptosystem…
  - Consider most powerful attacker (CTO< KPA< CPA)
  - Be as general as possible – cover many applications
  - And `easiest' attacker-success criteria
    - Not message/key recovery!
  - Make it easy to use securely, hard to use insecurely!
- When designing, evaluating a system
  - Which use some cryptosystem
  - Restrict attacker's capabilities (e.g., avoid known/chosen plaintext)

Cryptanalysis Success Criteria

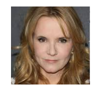
- Key recovery ? -- meaningless
- (Full) Message recovery? – may be an overkill. E.g., when $m \in$ {“Advance”, “Retreat”}
- Can’t learn anything at all about plaintext – how to define? Can we achieve it ?
  - Well-defined notion: ‘semantic security’ [crypto course]
- **Indistinguishability**: Eve ‘wins’ if she <u>distinguishes</u> between encryptions of (any) two messages
  - We focus on indistinguishability:
  - In crypto course: equivalent to semantic security

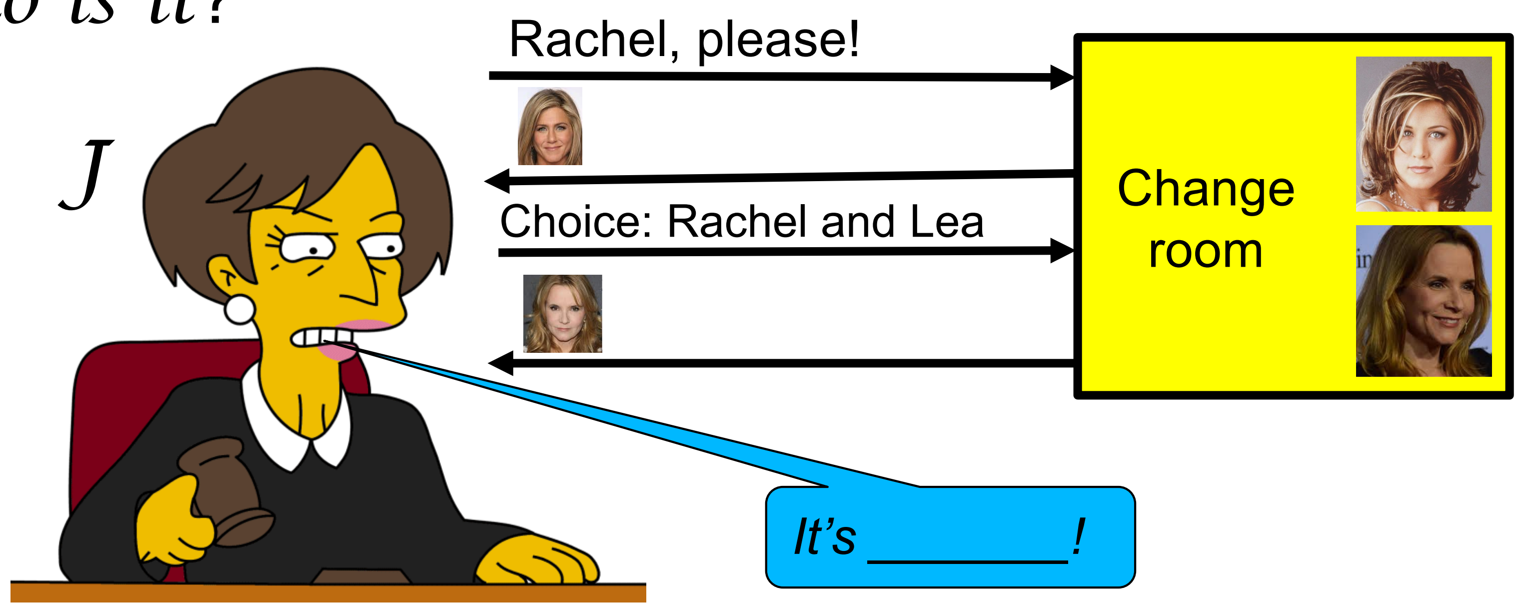
Defining Secure Encryption

- Attacker’s capabilities:
  - Computational limitations? ➔ PPT
  - Ciphertext only (CTO), Known / chosen plaintext attack (KPA/CPA), Chosen ciphertext (CCA)?
- Attacker’s goal: **distinguish** btw encryptions of two messages
  - Which messages? Let adversary choose!
  - Intuition: encryption is like ‘perfect disguise’

The Disguise Indistinguishability Test/Party

- *J* (Judge/Jacob): choses actress, see disguised
  - Many times, actresses……………... : Rachel, Lea, Natalie, …

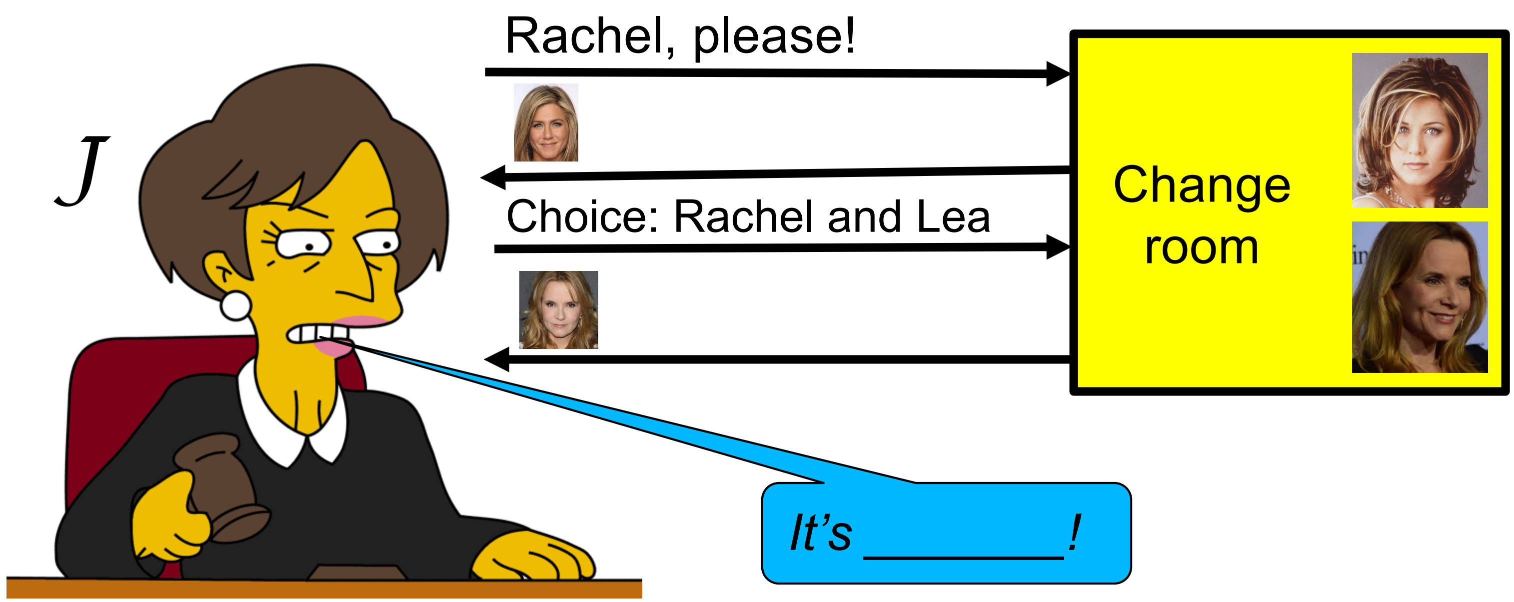
- *J* picks **two** of them… say: Rachel, Lea
- *J* sees one of them (disguised) 
- *Who is it?*



 19

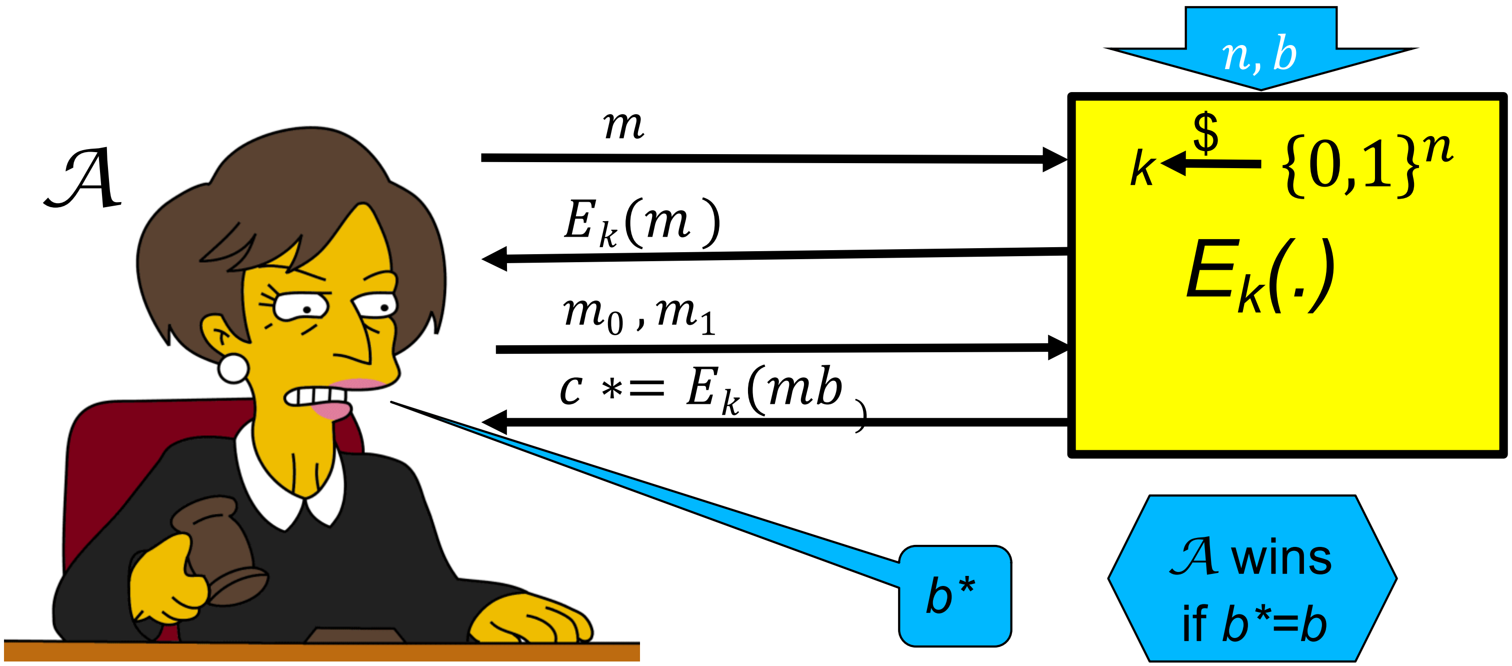
The Disguise Indistinguishability Test/Party

- Basic rules:
  - Actresses change custom *each time*
  - All are roughly same size
    - Can't ask a giant to disguise as a dwarf !



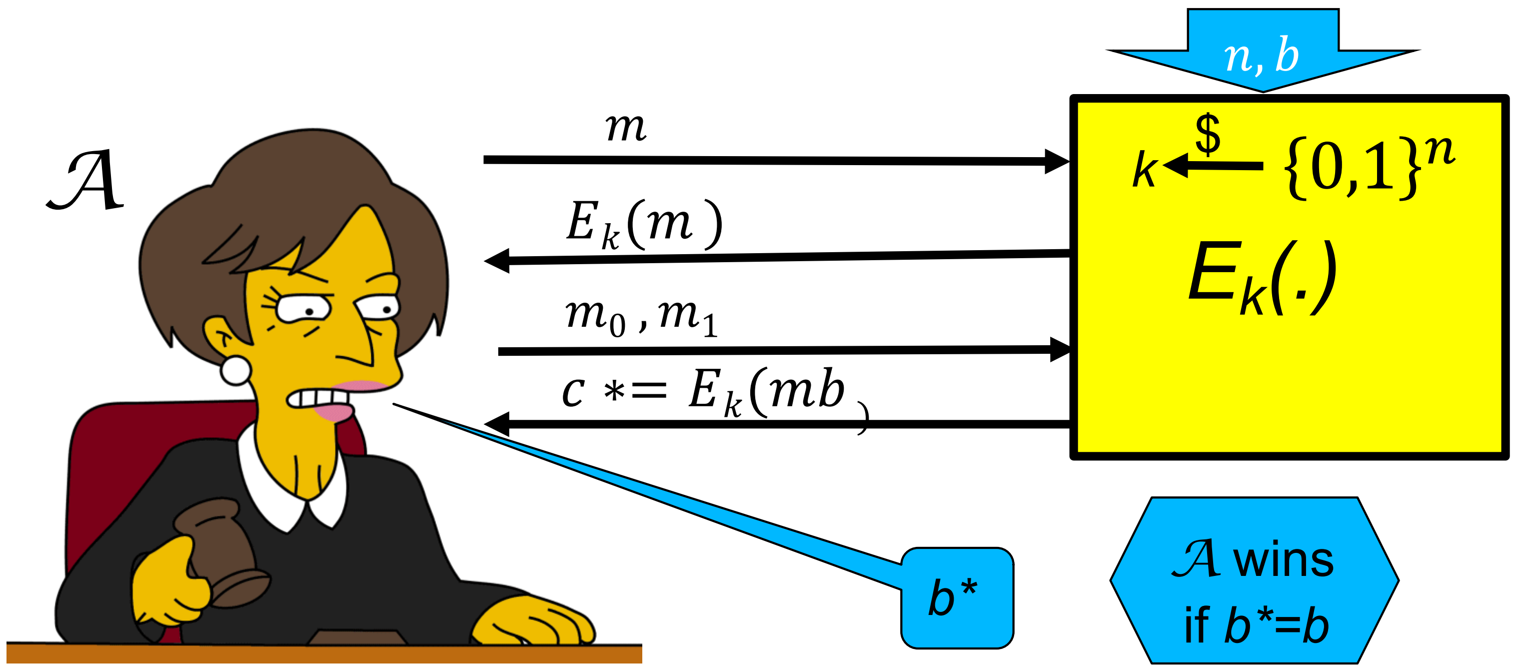
IND-CPA-Encryption Test (1st try)

- Flip coins to select random bit *b* and key *k*
- $\mathcal{A}$ (adversary) gives message *m*, receives $E_k(m)$
  - Repeat if desired (with different messages *m*)
  - Chosen Plaintext Attack
- $\mathcal{A}$ gives two messages (m_0,m_1), receives $c^*=E_k(m_b)$
- $\mathcal{A}$ output b^* , and 'wins' if $b^*=b$



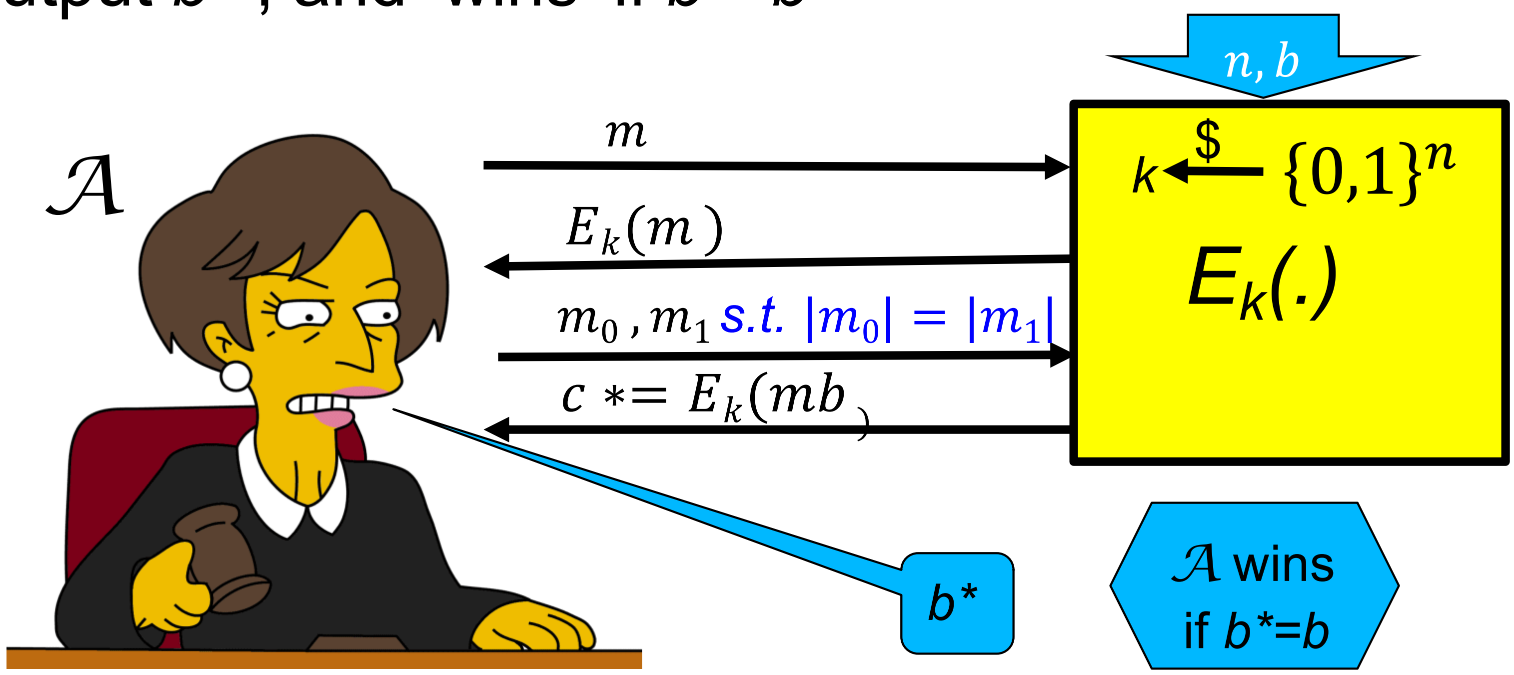
IND-CPA-Encryption Test (1^{st} try): **too easy!!**

- This test is too easy!! The adversary can easily win!!
- How?????????
- Hint: messages can be arbitrary binary strings
  - Namely, $m, m_0, m_1 \in \{0,1\}$^*
  - **Solution:** let $m_0=0$, $m_1=11111111111111111111111111111$
  - If $c^*=E_k(m_b)$ is `short', output $b^*=0$; if 'long', output $b^*=1$



IND-CPA-Encryption Test (fixed)

- Flip coins to select random bit *b* and key *k*
- $\mathcal{A}$ (adversary) gives message *m*, receives $E_k(m)$
  - Repeat if desired (with another message)
  - Chosen Plaintext Attack
- $\mathcal{A}$ gives messages (m_0,m_1) s.t. $|m_0|=|m_1|$, receives $E_k(m_b)$
- $\mathcal{A}$ output b^* , and 'wins' if $b^*=b$

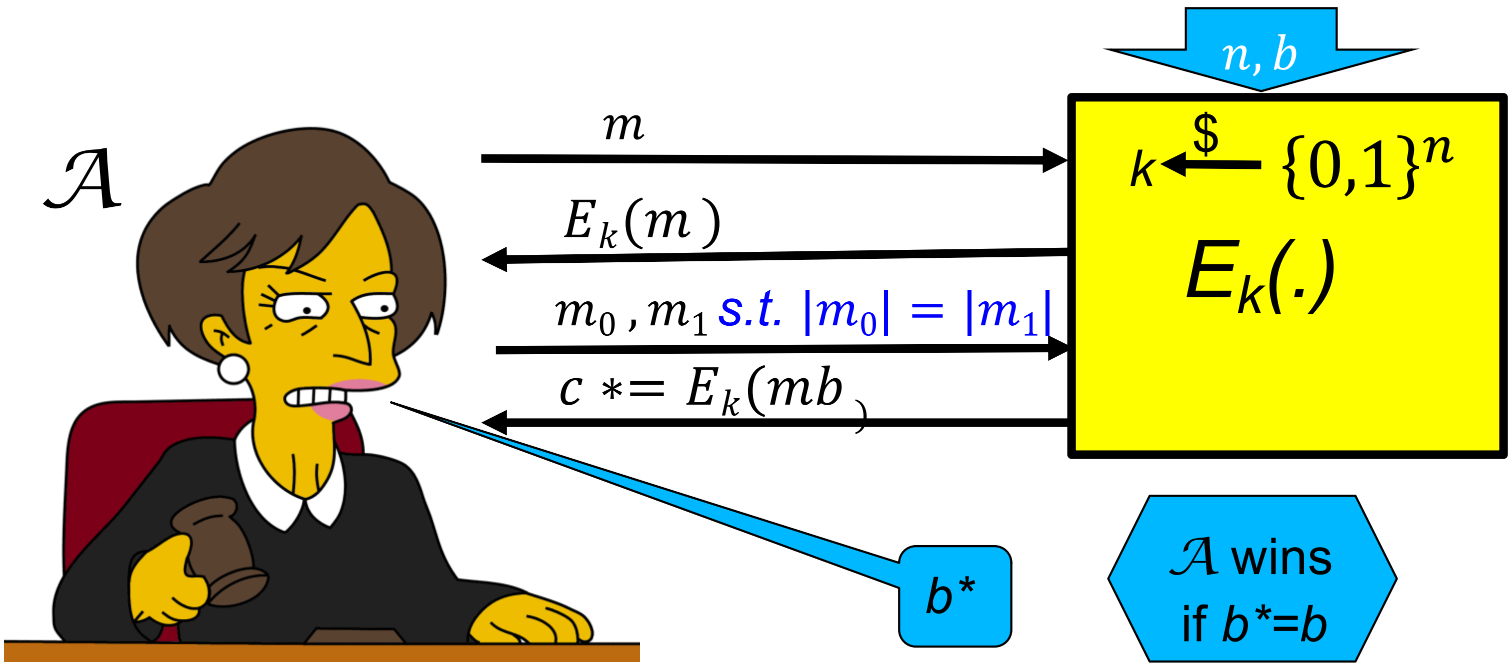


IND-CPA-Encryption Test (fixed)

- Or, as pseudo-code:

$$
\begin{aligned}
& T^{IND-CPA}_{\mathcal{A},\langle E,D\rangle}(b,n)\ \{\\
& \quad k \xleftarrow{\$} \{0,1\}^n\\
& \quad (m_0,m_1) \leftarrow \mathcal{A}^{E_k(\cdot)}(\text{'Choose'},1^n)\ \text{s.t.}\ |m_0|=|m_1|\\
& \quad c^* \leftarrow E_k(m_b)\\
& \quad b^* = \mathcal{A}^{E_k(\cdot)}(\text{'Guess'},c^*)\\
& \quad \text{Return } b^*\\
& \}
\end{aligned}
$$

Oracle notation



Definition: IND-CPA Encryption

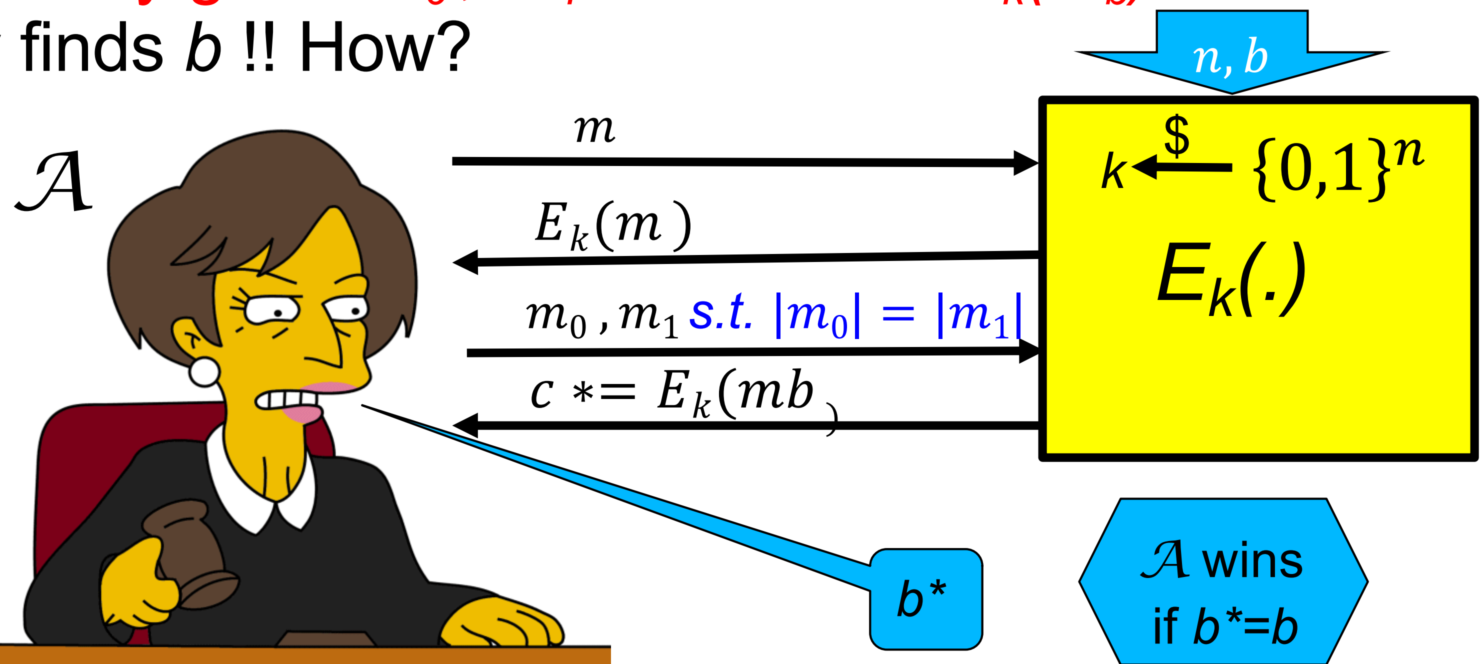
- Shared key cryptosystem (E, D) is **IND-CPA**, if every efficient adversary A has negligible advantage:

$$\varepsilon_{\langle E,D\rangle,\mathcal{A}}^{IND-CPA}(n) \equiv \Pr\left[T_{\mathcal{A},\langle E,D\rangle}^{IND-CPA}(1,n)=1\right] - \Pr\left[T_{\mathcal{A},\langle E,D\rangle}^{IND-CPA}(0,n)=1\right]$$

$$
\begin{aligned}
& T_{\mathcal{A},\langle E,D\rangle}^{IND-CPA}(b,n)\ \{\\
&\quad k \xleftarrow{\$} \{0,1\}^n\\
&\quad (m_0, m_1) \leftarrow \mathcal{A}^{E_k(\cdot)}(\text{`Choose'}, 1^n) \text{ s.t. } |m_0| = |m_1|\\
&\quad c^* \leftarrow E_k(m_b)\\
&\quad b^* = \mathcal{A}^{E_k(\cdot)}(\text{`Guess'}, c^*)\\
&\quad \text{Return } b^*\\
& \}
\end{aligned}
$$

IND-CPA : distinguish monoalph. sub.!

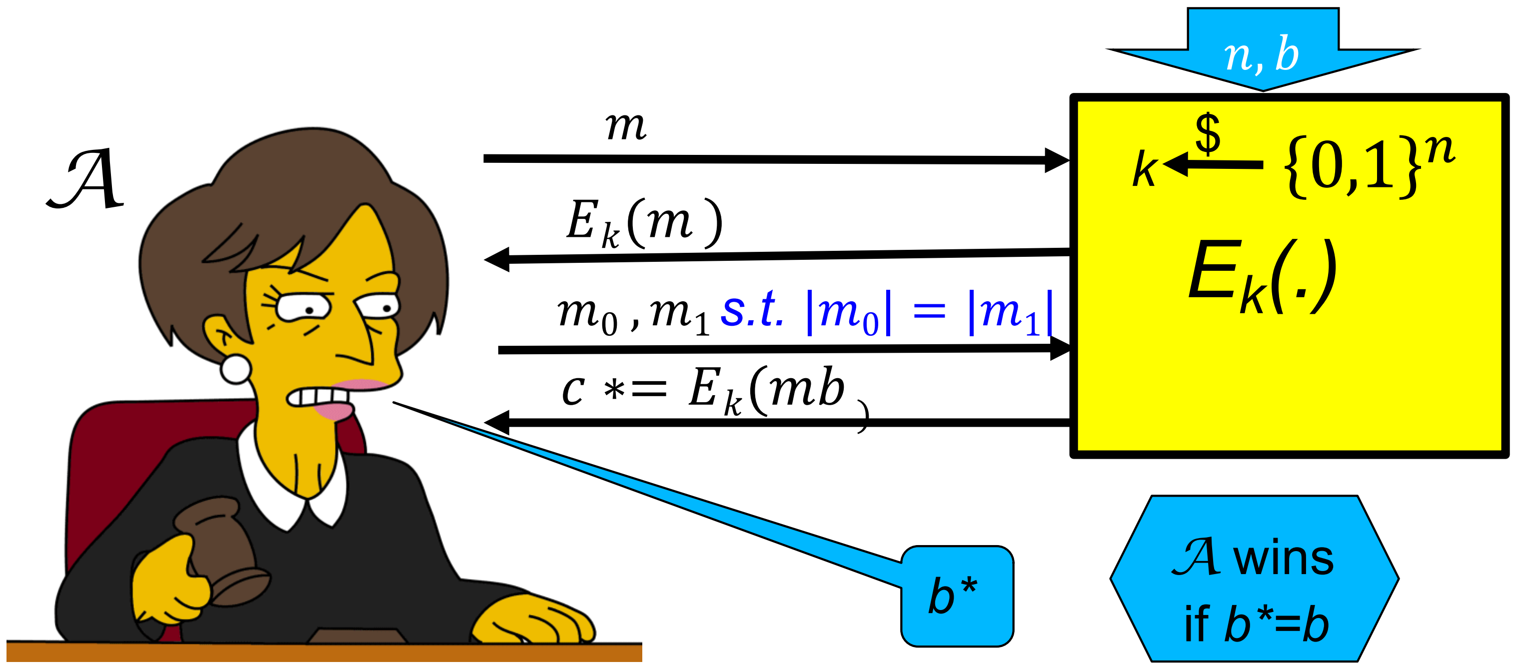
- Students split to pairs: adversary and \`tester'
- Tester selects (or receives) \`random' *(k, b)*
  - *k* is monoalphabetic substitution table: $E_k(abc)=k(a)||k(b)||k(c)$
- Adversary gives message(s) *m*, receives $E_k(m)$
- Then adversary gives m_0, m_1 … receives $E_k(m_b)$
- Adversary finds *b* !! How?



Monoalphabetic substitution is not IND-CPAistinguishable!

Can IND-CPA encryption be **deterministic**?

- No!! But why? Suppose $E_k(x)$ is deterministic…
- Assume messages are words (arbitrary length).
- $\mathcal{A}$ gives m=__________ , receives $c=E_k(m)$
- $\mathcal{A}$ gives m_0=__________ , m_1=_________, receives $c^*=E_k(m_b)$
- $\mathcal{A}$ outputs 1 if __________ , 0 otherwise - and **wins**!!
- Conclusion: IND-CPA Encryption **must be randomized**



What's next?

Present a secure cryptosystem?

… provably secure w/o assumptions ?

Unlikely: Proof of security ➔ $P \neq NP$

(similar argument to PRF)

Instead, let's build secure encryption from PRF !

(I.e.: PRF is secure ➔ encryption is IND-CPA)

Actually, we'll use **block cipher** (and build it)

PRP→Encryption: Modes of Operation

- `Modes of operation': use block cipher (PRP), to...
  - Encrypt long (Variable Input Length, VIL) messages
  - Randomize/add state for security
    - Often: use random/stateful *Initialization Vector (IV)*
  - Use longer or shorter keys
    - Longer key (e.g., Triple-DES): better security (at least against exhaustive search)
    - Shorter key: intentionally-weakened version, e.g. to meet export regulations
  - Other tasks (e.g., message authentication)

Encryption Modes of Operation

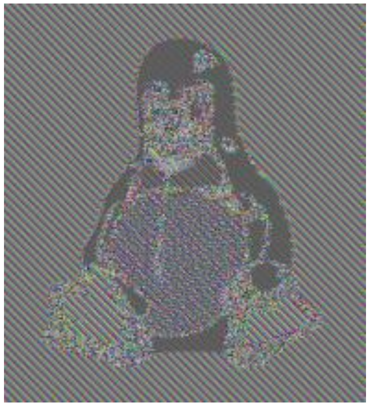
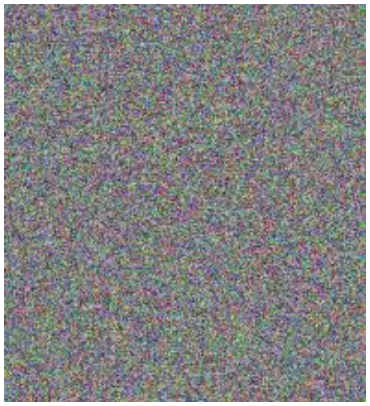
Mode	Encryption	Properties
Electronic code book (ECB)	$c_i = E_k(m_i)$	Insecure
Per-Block Random (PBR)	$r_i \xleftarrow{\$} \{0,1\}^n$, $c_i = (r_i, m_i \oplus E_k(r_i))$	Nonstandard, long ciphertext
Output Feedback (OFB)	$r_0 \xleftarrow{\$} \{0,1\}^n, r_i = E_k(r_{i-1})$, $c_0 \leftarrow r_0,\ c_i \leftarrow r_i \oplus m_i$	Parallel, fast online, PRF, 1-localization
Cipher Feedback (CFB)	$c_0 \xleftarrow{\$} \{0,1\}^n$, $c_i \leftarrow m_i \oplus E_k(c_{i-1})$	Parallel decrypt PRF, $n+1$-localization
Cipher-Block Chaining (CBC)	$c_0 \xleftarrow{\$} \{0,1\}^n$, $c_i \leftarrow E_k(m_i \oplus c_{i-1})$	parallel decrypt $n+1$-localization
Counter (CTR)	$T_1 \leftarrow nonce \# 0^{n/2}$, $T_i \leftarrow T_{i-1}+1$, $c_i = m_i \oplus E_k(T_i)$	Parallel, fast online, PRF, 1-localization, **stateful** (*nonce*)

Block Cipher Modes of Operation

- For encryption
  - Later: modes for message authentication
  - Assume plaintext is in blocks: $m_0||m_1||...$
- Electronic Code Book mode (ECB): encryption $c_i=E_k(m_i)$, decryption $m_i=D_k(c_i)$

$m_i \rightarrow$ [`Encrypt` E_k] $\rightarrow c_i \rightarrow$ [`Decrypt` D_k] $\rightarrow m_i$

Which of these is ECB encryption? Why?

Per-Block Random (PBR) mode

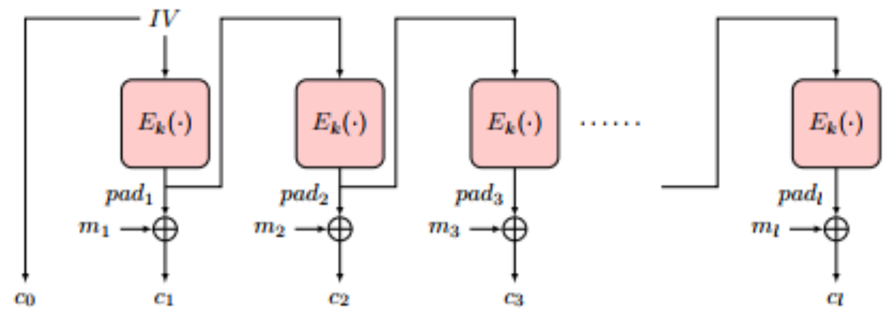
- A simple way to construct secure encryption from PRP/PRF
  - Not a standard mode – presented just for teaching
- $Enc_k(m)=(r_i\ ,\ m_i \oplus E_k(r_i))$
  - $m_i : i^{th}$ block of bits
  - r_i : random block of bits
- $Dec_k((r_i\ ,\ m_i \oplus E_k(r_i)))=E_k(r_i) \oplus m_i \oplus E_k(r_i)= m_i$
- Wasteful: random block per plaintext block
- Confidentiality ? **Yes!**
  - Theorem: If (E, D) is a PRP, then (Enc, Dec) is a IND-CPA cryptosystem.
- Integrity? No: flip ciphertext bit ➔ flip corresponding plaintext bit

Encryption Modes of Operation

- We saw two...
- ECB (insecure!): $c_i = E_k(m_i)$
- Per-Block Random (PBR): $r_i \leftarrow \$,\ c_i = (r_i\ ,\ m_i \oplus E_k(r_i))$
- We'll see three more…
  - Output Feedback (OFB)
  - Cipher Feedback (CFB)
  - Cipher-block-chaining (CBC)
- Others exist (for encryption – and other tasks)
- All operate on **blocks** (e.g., 128 bits = 16 bytes)

Output-Feedback (OFB) Mode

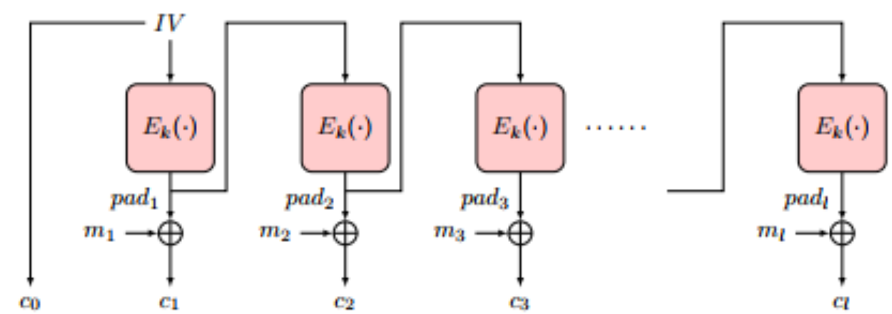
- Goal: encrypt long (multi-block) messages, with **less random bits**
  - Generate and send less random bits – cf. to per-block random
- How? Use random bits only for first block (`initialization vector')
  - To encrypt next blocks of message, use output of previous block
  - Namely, a **block-by-block stream cipher**
- Encryption: $pad_0 \leftarrow IV$, $pad_i \leftarrow E_k(pad_{i-1})$, $c_0 \leftarrow pad_0$, $c_i \leftarrow pad_i \oplus m_i$



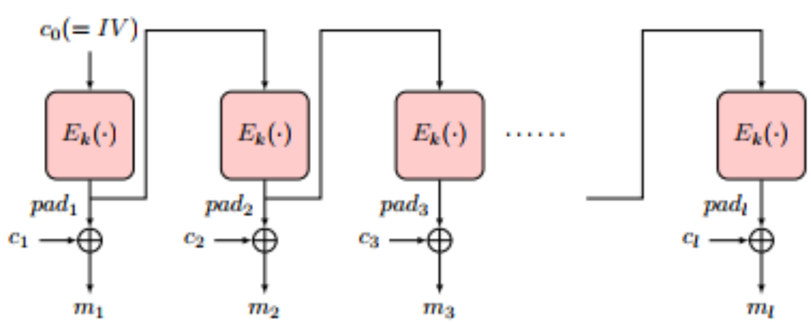
- Decryption: ?

Output-Feedback (OFB) Mode

- Goal: encrypt long (multi-block) messages, with **less random bits**
  - Generate <u>and send</u> less random bits – cf. to per-block random
- How? Use random bits only for first block (`initialization vector’)
  - To encrypt next blocks of message, use output of previous block
  - Namely, a **block-by-block stream cipher**
- Encryption: $pad_0 \leftarrow IV$, $pad_i \leftarrow E_k(pad_{i-1})$, $c_0 \leftarrow pad_0$, $c_i \leftarrow pad_i \oplus m_i$



- Decryption: $pad_0 \leftarrow c_0$, $pad_i \leftarrow E_k(p_{i-1})$, $m_i \leftarrow pad_i \oplus c_i$



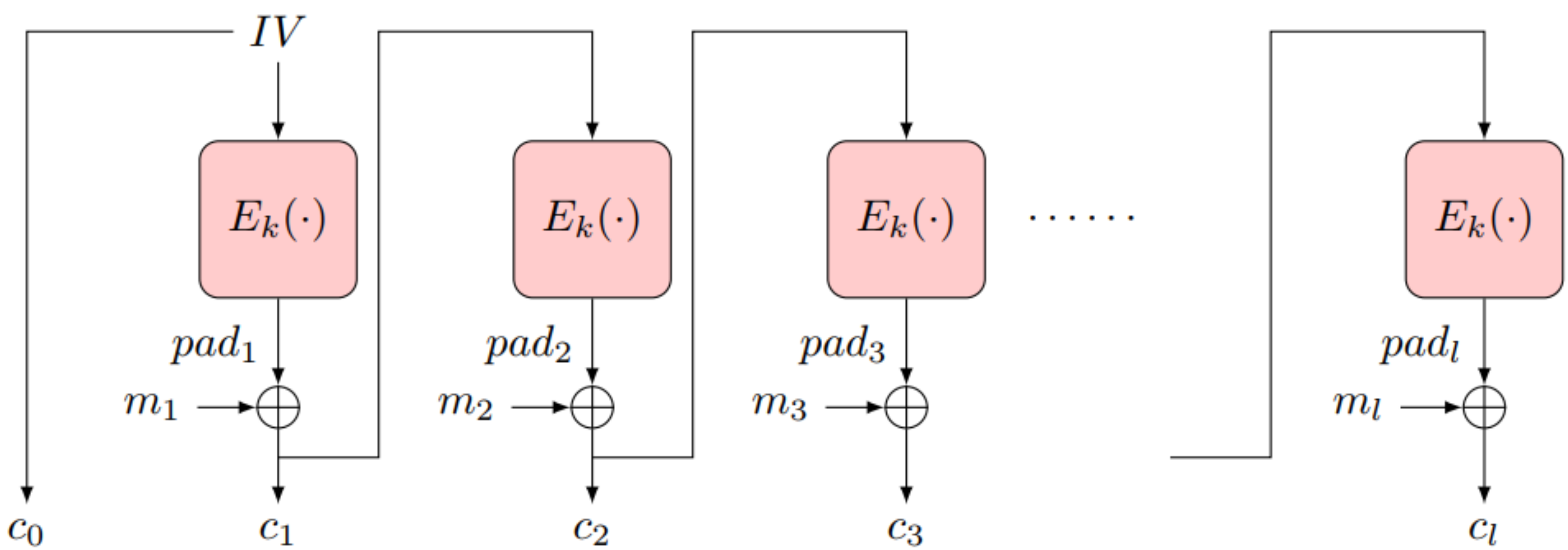
Output-Feedback (OFB) Mode

- Encryption: $pad_0 \leftarrow IV,\ \ pad_i \leftarrow E_k(pad_{i-1}),$
  $c_0 \leftarrow pad_{0},\ c_i \leftarrow pad_i \oplus m_i$
- Decryption: $pad_0 \leftarrow c_{0},\ pad_i \leftarrow E_k(p_{i-1}),$
  $m_i \leftarrow pad_i \oplus c_i$

- Offline pad computation: compute pad in advance
  - Online computation: only (parallelizable) XOR !
- Bit errors are bitwise **localized** (corrupt only one bit)
- No integrity:
  Flip ciphertext bit ➔ flip corresponding decrypted plaintext bit
- Can we protect integrity?

Cipher-Feedback Block (CFB) Encryption

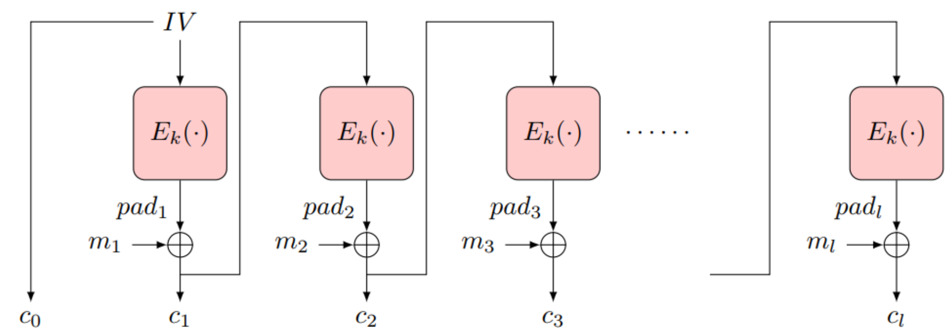
- Random first block c_0 (`initialization vector', *IV*)
- XOR 'pad' $E_k(c_0)$ with plaintext to obtain: $c_1 = m_1 \oplus E_k(c_0)$
- Repeat: $c_i = m_i \oplus E_k(c_{i-1})$



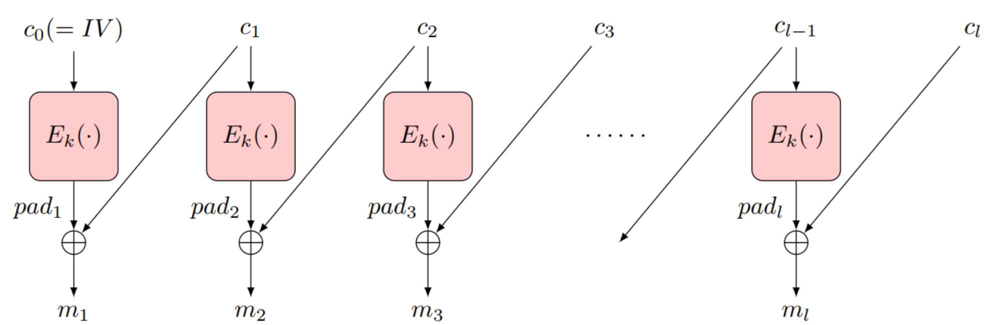
Cipher-Feedback Block (CFB) Encryption

- Random first block c_0 (`initialization vector', *IV*)
- XOR 'pad' $E_k(c_0)$ with plaintext to obtain: $c_1 = m_1 \oplus E_k(c_0)$
- Repeat: $c_i = m_i \oplus E_k(c_{i-1})$

CFB Encryption



CFB Decryption

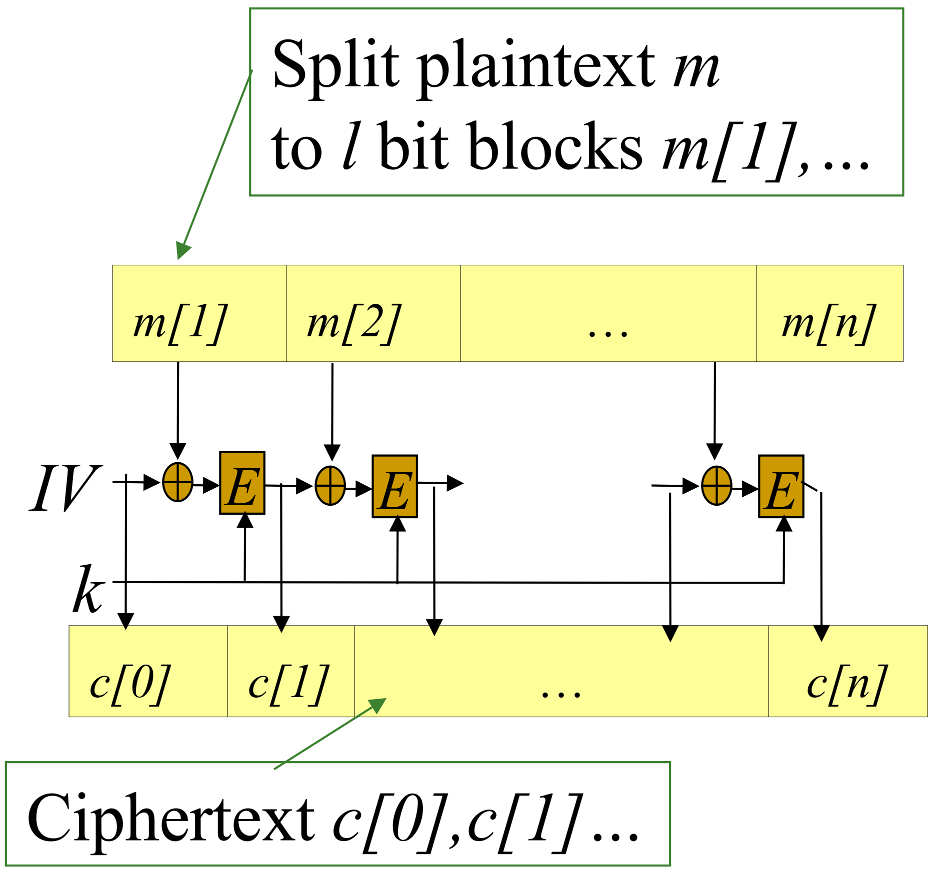


Cipher-Feedback Block (CFB) Encryption

- Random first block c_0 (`initialization vector', *IV*)
- XOR 'pad' $E_k(c_0)$ with plaintext to obtain: $c_1 = m_1 \oplus E_k(c_0)$
- Repeat: $c_i = m_i \oplus E_k(c_{i-1})$
- Ciphertext: $c_0, c_1 = m_1 \oplus E_k(c_0), \ldots, c_i = m_i \oplus E_k(c_{i-1})$
  - Can't pre-compute `pad' offline ☹
- Decryption: $c_i \oplus E_k(c_{i-1}) = m_i \oplus E_k(c_{i-1}) \oplus E_k(c_{i-1}) = m_i$
  - Parallelizable
  - Bit/block errors are 2-block **localized** (corrupt only 2 blocks)
- Integrity? A bit…
  - Flip ciphertext bit ➔ flip corresponding decrypted plaintext bit
  - **But also corrupt next plaintext block**
    - **Except for last block: no `next block'**
- Can we protect integrity (even) better?

Cipher Block Chaining (CBC) Mode

- Random first block c_0 (`initialization vector', *IV*)
- $i>0$: $c_i = E_k(c_{i-1} \oplus m_i)$
- Parallel decryption
  - But no offline precomputing
- Integrity: flip bit in $c[i]$ ➔ flip bit in $m[i+1]$ …
  But also corrupt $m[i]$
- May suffice to ensure integrity for many applications
- But not all!



Security of CBC mode (2)

- Thm: If block-cipher E is a (strong) pseudo-random permutation ➔CBC#E is IND-CPA-secure encryption
- Proof: omitted (crypto course ☺)
- Observation: CBC is Not IND-CCA-Secure
  - CCA (Chosen ciphertext attack), intuitively: attacker can choose ciphertext and get its decryption, except for the `challenge ciphertext'
  - Definition, details: crypto course
  - Exercise: show CBC is Not IND-CCA-Secure
- Feedback-CCA: practical variant of CCA
  - Just returns <ERROR, OK> for any ciphertext
  - Error – for incorrectly padded decryption (next)

Encryption: Final Words

- Basic goal of cryptography
- Focus: computationally-limited adversaries
- Principles:
  - Kerckhoff's: Known Design
  - Sufficient Key Space
  - Crypto Building Block: build schemes from simple, standard functions
    - Constructions & reductions: PRG→PRF→PRP→Enc
  - Secure system design: easy to use securely, hard to use incorrectly!

Encryption: Final Words...

- Many variants…
- One important example is Homomorphic encryption: $E(m_1+m_2)=EncAdd(E(m_1),E(m_2))$
  - Where EncAdd is an efficient algorithm
  - Fully-homomorphic : also E(m1*m2)=EncMult(E(m1), E(m2))
  - Very inefficient designs, huge keys… but lots of research!

Covered Material From the Textbook

- Will be updated later.

Thank You!