Updated Office Hours

- Tuesday: 10:30 AM-11:30 AM
- ~~Thursday: 10:30 AM-11:30 AM~~
- Friday: 10:30 AM-11:30 AM

Office: Lawson 1165

- Plain RSA Public Key: (N=1165,e=11) ☺
  - Challenge: Decrypt the ciphertext c=610

Cryptography
CS 555

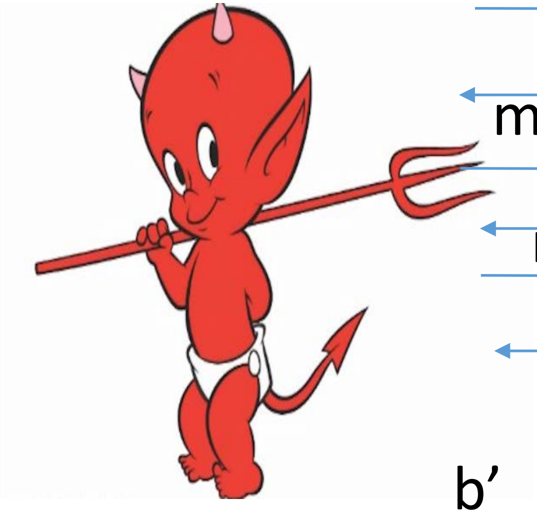
Week 3:
- Building CPA-Secure Encryption Schemes
- Pseudorandom Functions/Permutations
- Block Ciphers + Modes of Operation
- CCA-Security (definition)
- Message Authentication Codes [time permitting]

Readings: Katz and Lindell Chapter 3.5-3.7

Recap CPA-Security

- Defend against attacker's ability to influence messages that honest party encrypts
- Practical Importance: Battle of Midway
- CPA-Security Equivalence
  - Multiple vs Single Encryption Game
- Limitations
  - Passive vs Active Attacker
  - What if attacker can get honest party to (partially) decrypt some messages?

CPA-Security (Multiple Messages)



$m_{0,1}, m_{1,1}$

$c_1 = \mathrm{Enc}_K(m_{b,1})$

$m_{0,2}, m_{1,2}$

$c_2 = \mathrm{Enc}_K(m_{b,2})$

$m_{0,3}, m_{1,3}$

$c_3 = \mathrm{Enc}_K(m_{b,3})$

$\vdots$

b’

Random bit b

$\mathbf{K \leftarrow Gen(1^n)}$

$$\forall PPT\ A\ \exists \mu\ (\text{negligible})\ \text{s.t}$$

$$\Pr\left[PrivK_{A,\Pi}^{cpa}\right] \leq \frac{1}{2} + \mu(n)$$

CPA-Security and Message Length

Observation: Given a CPA-secure encryption scheme $\Pi = (\mathrm{Gen}, \mathrm{Enc}, \mathrm{Dec})$ that supports messages of a single bit ($\mathcal{M} = \{0,1\}$) it is easy to build a CPA-secure scheme $\Pi' = (\mathrm{Gen}', \mathrm{Enc}', \mathrm{Dec}')$ that supports messages m = $m_1,\ldots,m_n \in \{0,1\}^n$ of length n.

$$\mathrm{Enc}'_{\mathrm{k}}(m) = \left\langle \mathrm{Enc}_{\mathrm{k}}\,(m_1), \ldots, \mathrm{Enc}_{\mathrm{k}}\,(m_n) \right\rangle$$

Exercise: How would you prove Π' is CPA-secure?

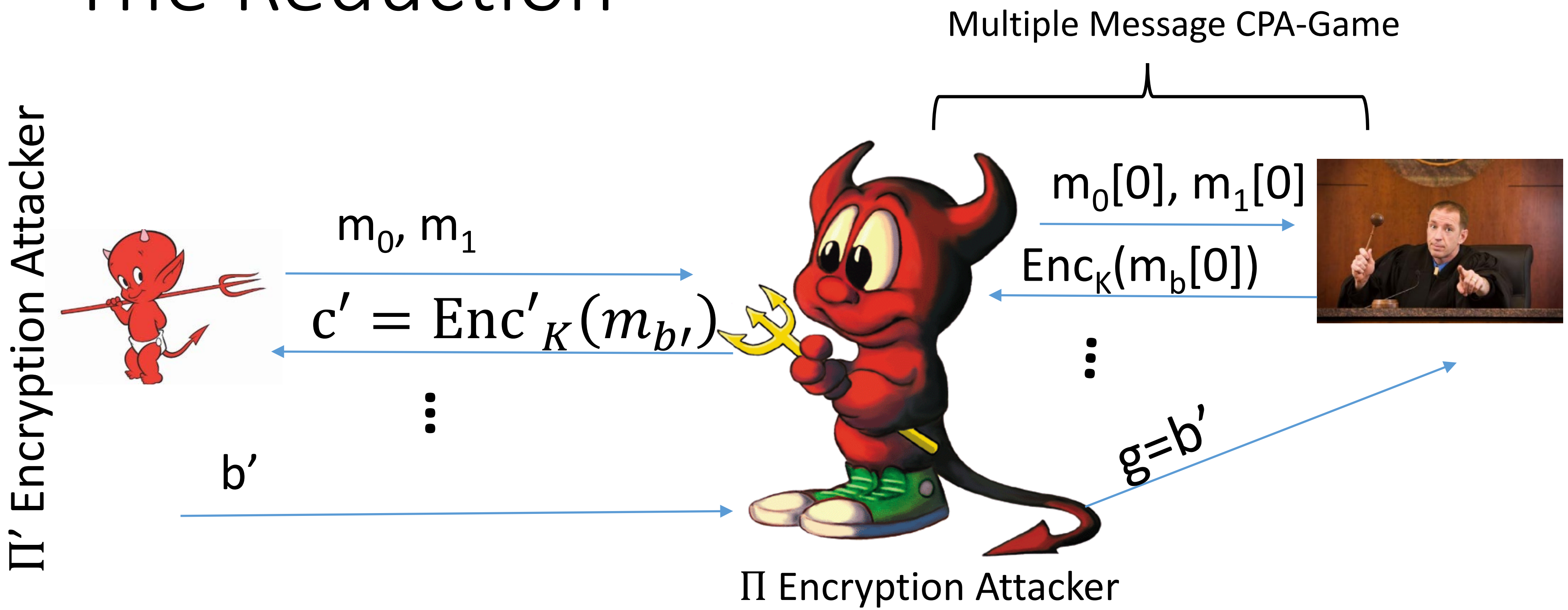
Security Reduction

- **Step 1:** Assume for contraction that we have a PPT attacker A that breaks CPA-Security.
- **Step 2:** Construct a PPT distinguisher D which breaks PRF security.

The Reduction

Random bit b

$K \leftarrow \mathrm{Gen}(1^n)$



$$\mathrm{Enc}'_{\mathrm{k}}(m) = \langle \mathrm{Enc}_{\mathrm{k}}\,(m_1), \ldots, \mathrm{Enc}_{\mathrm{k}}\,(m_n) \rangle$$

Week 3: Topic 1: Pseudorandom Functions and CPA-Security

Pseudorandom Function (PRF)

A keyed function F: $\{0,1\}^{\ell_{key}(n)} \times \{0,1\}^{\ell_{in}(n)} \to \{0,1\}^{\ell_{out}(n)}$, which “looks random” without the secret key k.

- $\ell_{key}(n)$ - length of secret key k
- $\ell_{in}(n)$ - length of input
- $\ell_{out}(n)$ - length of output

- Typically, $\ell_{key}(n)$=$\ell_{in}(n)$=$\ell_{out}(n)$ =n (unless otherwise specified)

- Computing $F_K(x)$ is efficient (polynomial-time)

PRF vs. PRG

- Pseudorandom Generator G is not a keyed function
- PRG Security Model: Attacker sees only output G(r)
  - Attacker who sees r can easily distinguish G(r) from random
- PRF Security Model: Attacker sees both inputs and outputs $(r_i, F_k(r_i))$
  - In fact, attacker can select inputs r_i
  - Attacker Goal: distinguish F from a truly random function

Truly Random Function

- Let **Func$_n$** denote the set of all functions $f: \{0,1\}^n \to \{0,1\}^n$.

- **Question**: How big is the set **Func$_n$**?
- **Hint**: Consider the lookup table.
  - 2^n entries in lookup table
  - n bits per entry
  - $n2^n$ bits to encode f∈**Func$_n$**
- **Answer**: $|$**Func$_n$**$| = 2^{n2^n}$ (by comparison only 2^n n-bit keys)

Truly Random Function

- Let $\mathbf{Func_n}$ denote the set of all functions $f: \{0,1\}^n \rightarrow \{0,1\}^n$.

- Can view entries in lookup table as populated in advance (uniformly)
  - **Space:** $n2^n$ bits to encode $f \in \mathbf{Func_n}$
- Alternatively, can view entries as populated uniformly "on-the-fly"
  - **Space:** $2n \times q(n)$ bits after $q(n)$ queries
    - To store past responses

Oracle Notation

- We use $A^{f(.)}$ to denote an algorithm A with oracle access to a function f.
- A may adaptively query f(.) on multiple different inputs $x_1, x_2, \ldots$ and A receives the answers $f(x_1), f(x_2), \ldots$
- However, A can only use f(.) as a blackbox (no peaking at the source code in the box)

PRF Security

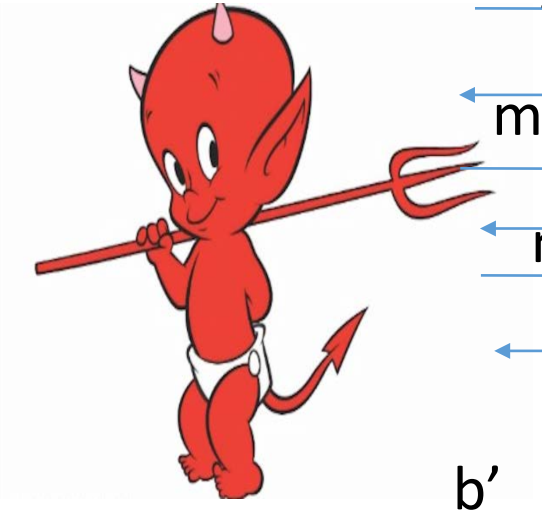
Definition 3.25: A keyed function $\mathrm{F}: \{0,1\}^n \times \{0,1\}^n \rightarrow \{0,1\}^n$ is a pseudorandom function if for all PPT distinguishers D there is a negligible function μ s.t.

$$\left|Pr\left[D^{F_k(.)}(1^n)\right] - Pr\left[D^{f(.)}(1^n)\right]\right| \leq \mu(n)$$

Notes:

- the first probability is taken over the uniform choice of $k \in \{0,1\}^n$ as well as the randomness of D.
- the second probability is taken over uniform choice of f ∈$\mathbf{Func_n}$as well as the randomness of D.
- D is *not* given the secret k in the first probability (otherwise easy to distinguish…how?)

PRF-Security as a Game



m_1

r_1

m_2

r_2

m_3

r_3

⋮

b'

Random bit b

$\mathbf{K \leftarrow Gen(1^n)}$

Truly random func R

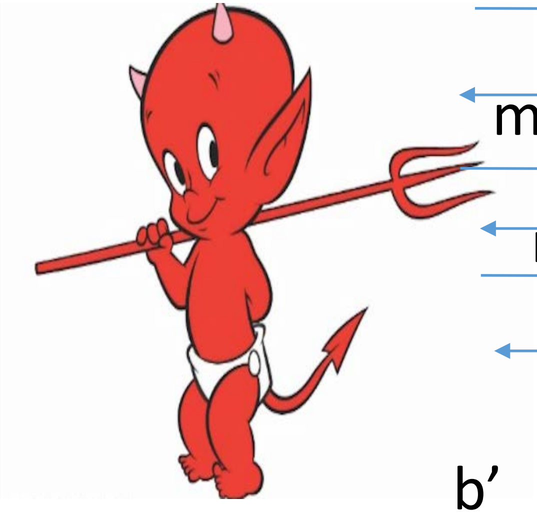
$\mathbf{r_i = F_K(m_i)}$ **if b=1**

$\mathbf{R(m_i)}$ **o.w**

$$\forall PPT\ A\ \exists \mu\ (\text{negligible})\ \text{s.t}$$

$$\Pr[A\ \ Guesses\ b' = b] \le \frac{1}{2} + \mu(n)$$

Reminder: CPA-Security (Single Message)



m_0, m_1

$c = Enc_K(m_b)$

m_2

$c_2 = Enc_K(m_2)$

m_3

$c_3 = Enc_K(m_3)$

⋮

b’

Random bit b

$\mathbf{K \leftarrow Gen(1^n)}$

$$\forall PPT\ A\ \exists \mu\ (\text{negligible})\ \text{s.t}$$

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

CPA-Secure Encryption

- Gen: on input 1^n pick uniform $k \in \{0,1\}^n$
- Enc: Input $k \in \{0,1\}^n$ and $m \in \{0,1\}^n$

  Output $c = \langle r, F_k(r) \oplus m \rangle$ for uniform $r \in \{0,1\}^n$

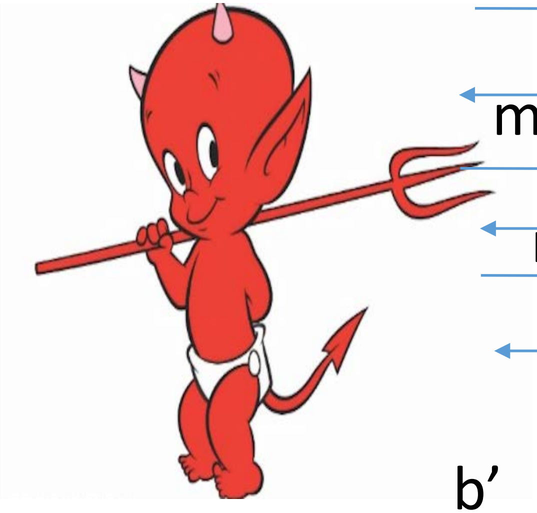
- Dec: Input $k \in \{0,1\}^n$ and $c = \langle r, s \rangle$

  Output $m = F_k(r) \oplus s$

How to begin proof?

Theorem: If F is a pseudorandom function, then (Gen,Enc,Dec) is a CPA-secure encryption scheme for messages of length n.

Breaking CPA-Security (Single Message)



m_0, m_1

$\langle r, F_k(r) \oplus m_b \rangle$

m_2

$\langle r, F_k(r) \oplus m_2 \rangle$

m_3

$\langle r, F_k(r) \oplus m_3 \rangle$

⋮

b’

Random bit b

$\mathbf{K \leftarrow Gen(1^n)}$

Assumption: $\exists$ PPT A, P (non − negligible) s.t

$$\Pr[A \;\; Guesses\; b' = b] \geq \frac{1}{2} + P(n)$$

Security Reduction

- **Step 1:** Assume for contraction that we have a PPT attacker A that breaks CPA-Security.
- **Step 2:** Construct a PPT distinguisher D which breaks PRF security.
- Distinguisher D^O (oracle O --- either f or F_k)
  - Simulate A
  - Whenever A queries its encryption oracle on a message m
    - Select random r
    - Return $c = \langle r, O(r) \oplus m \rangle$
  - Whenever A outputs messages m_0,m_1
    - Select random r and bit b
    - Return $c = \langle r, O(r) \oplus m_b \rangle$
  - Whenever A outputs b'
    - Output 1 if b=b'
    - Output 0 otherwise

Analysis: Suppose that O = f then

$$\Pr[D^{F_k} = 1] = \Pr\left[PrivK_{A,\Pi}^{cpa} = 1\right]$$

Suppose that O = f then

$$\Pr\left[D^{f} = 1\right] = \Pr\left[PrivK_{A,\widetilde{\Pi}}^{cpa} = 1\right]$$

where $\widetilde{\Pi}$ denotes the encryption scheme in which F_k is replaced by truly random f.

Security Reduction

- **Step 1:** Assume for contraction that we have a PPT attacker A that breaks CPA-Security.
- **Step 2:** Construct a PPT distinguisher D which breaks PRF security.
- Distinguisher D^O (oracle O --- either f or F_k)
  - Simulate A
  - Whenever A queries its encryption oracle on a message m
    - Select random r
    - Return $c = \langle r, O(r) \oplus m \rangle$
  - Whenever A outputs messages m_0,m_1
    - Select random r and bit b
    - Return $c = \langle r, O(r) \oplus m_b \rangle$
  - Whenever A outputs b'
    - Output 1 if b=b'
    - Output 0 otherwise

Analysis: Suppose that O = F_k then by PRF security, for some negligible function μ, we have

$$\left|\Pr\left[PrivK_{A,\Pi}^{cpa} = 1\right] - \Pr\left[PrivK_{A,\widetilde{\Pi}}^{cpa} = 1\right]\right|$$
$$= \left|\Pr[D^{F_k} = 1] - \Pr[D^{f} = 1]\right| \le \mu(n)$$

Implies: $\Pr\left[PrivK_{A,\widetilde{\Pi}}^{cpa} = 1\right] \ge \Pr\left[PrivK_{A,\Pi}^{cpa} = 1\right] - \mu(n)$

Security Reduction

- **Fact:** Pr $\left[PrivK_{A,\widetilde{\Pi}}^{cpa} = 1\right] \geq$ Pr$\left[PrivK_{A,\Pi}^{cpa} = 1\right]$-$\mu(n)$

- **Claim**: For any attacker A making at most q(n) queries we have

$$\Pr\left[PrivK_{A,\widetilde{\Pi}}^{cpa} = 1\right] \leq \frac{1}{2} + \frac{q(n)}{2^n}$$

Conclusion: For any attacker A making at most q(n) queries we have

$$\Pr\left[PrivK_{A,\Pi}^{cpa} = 1\right] \leq \frac{1}{2} + \frac{q(n)}{2^n} + \mu(n)$$

where $\frac{q(n)}{2^n} + \mu(n)$ is negligible.

Finishing Up

Claim: For any attacker A making at most q(n) queries we have

$$\Pr\left[PrivK_{A,\widetilde{\Pi}}^{cpa} = 1\right] \le \frac{1}{2} + \frac{q(n)}{2^n}$$

Proof: Let m_0,m_1 denote the challenge messages and let r* denote the random string used to produce the challenge ciphertext

$$c = \langle r^*, f(r^*) \oplus m_b \rangle$$

And let r_1,…,r_q denote the random strings used to produce the other ciphertexts $c_i = \langle r_i, f(r_i) \oplus m_i \rangle$.

If $\mathrm{r}^* \neq \mathrm{r}_1$,…,$\mathrm{r}_\mathrm{q}$then then c leaks no information about b (information theoretically).

Finishing Up

Claim: For any attacker A making at most q(n) queries we have

$$\Pr\left[PrivK_{A,\widetilde{\Pi}}^{cpa} = 1\right] \le \frac{1}{2} + \frac{q(n)}{2^n}$$

Proof: If $\mathrm{r}^* \neq \mathrm{r}_1,\ldots,\mathrm{r}_\mathrm{q}$ then then c leaks no information about b (information theoretically). We have

$$\Pr\left[PrivK_{A,\widetilde{\Pi}}^{cpa} = 1\right]$$

$$\le \Pr\left[PrivK_{A,\widetilde{\Pi}}^{cpa} = 1 \,\middle|\, \mathrm{r}^* \neq \mathrm{r}_1,\ldots,\mathrm{r}_\mathrm{q}\right] + \Pr\left[\mathrm{r}* \in \{\mathrm{r}_1,\ldots,\mathrm{r}_\mathrm{q}\}\right]$$

$$\le \frac{1}{2} + \frac{q(n)}{2^n}$$

Conclusion

$$\text{Enc}_\text{k}(\text{m}) = \langle r, F_k(r) \oplus m \rangle$$

$$\text{Dec}_\text{k}(\langle r, s \rangle) = F_k(r) \oplus s$$

PRF Security

For any attacker A making at most q(n) queries we have

$$\Pr\left[PrivK_{A,\Pi}^{cpa} = 1\right] \leq \frac{1}{2} + \frac{q(n)}{2^n} + \boxed{\mu(n)}$$

Are PRFs or PRGs more Powerful?

- Easy to construct a secure PRG from a PRF

$$G(s) = F_s(1) | \ldots | F_s(\ell)$$

- Construct a PRF from a PRG?
  - Tricky, but possible… (Katz and Lindell Section 7.5)

Construct PRF from PRG

Define: $G(s) = G_0(s) | G_1(s)$

PRF: $F_k(x) = G_{x_1}\left(\ldots G_{x_{n-1}}\left(G_{x_n}(k)\right)\right)$

Recursive Definition: $F_k(x) = H_k(x)$ where

$H_k(1) := G_1(k)$

$H_k(0) := G_0(k)$

$H_k(1|x) := G_1(H_k(x))$

$H_k(0|x) := G_0(H_k(x))$

Theorem: If G is a PRG then F_k is a PRF

Candidate PRG

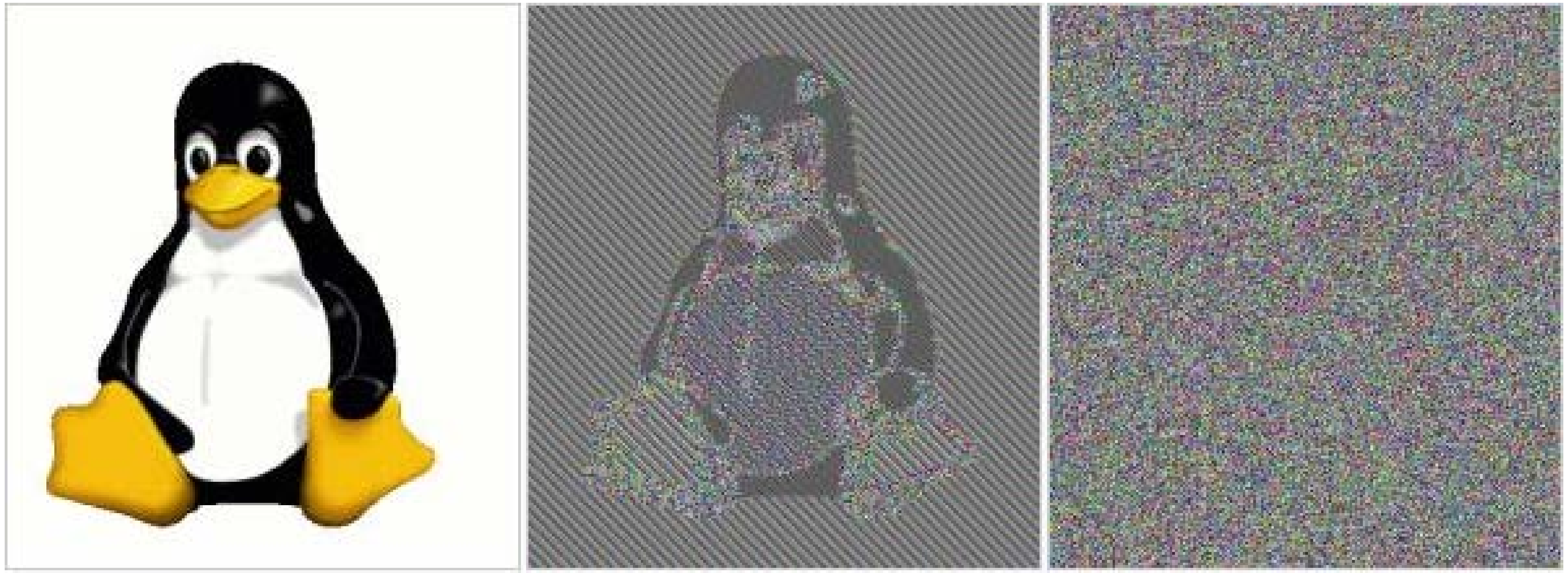
- **Notation:** Given string $x \in \{0,1\}^n$ and a subset $S \subset \{1, \ldots, n\}$ let $x_S \in \{0,1\}^{|S|}$ denote the substring formed by concatenating bits at the positions in S.
- **Example**: x=10110 and S = {1,4,5} x_S=110

$$P(x_1, x_2, x_3, x_4, x_5) = x_1 + x_2 + x_3 + x_4 x_5 \bmod 2$$

- Select random subsets $\mathbb{S} = S_1, \ldots, S_{\ell(n)} \subset \{1, \ldots, n\}$ of size $|S_i|=5$ and with $\ell(n) = n^{1.4}$

$$G_{\mathbb{S}}(x) = P\left(x_{S_1}\right) | \ldots | P\left(x_{S_{\ell(n)}}\right)$$

Week 3: Topic 2: Modes of Encryption, The Penguin and CCA security



Original image

Encrypted using ECB mode

Modes other than ECB result in pseudo-randomness

Pseudorandom Permutation

A keyed function $\mathrm{F}: \{0,1\}^n \times \{0,1\}^n \rightarrow \{0,1\}^n$, which ***is invertible*** and "looks random" without the secret key k.

- Similar to a PRF, but
- Computing $F_k(x)$ *and* $F_k^{-1}(x)$ is efficient (polynomial-time)

Definition 3.28: A keyed function $\mathrm{F}: \{0,1\}^n \times \{0,1\}^n \rightarrow \{0,1\}^n$ is a **strong pseudorandom permutation** if for all PPT distinguishers D there is a negligible function μ s.t.

$$\left| Pr\left[D^{F_k(.),F_k^{-1}(.)}(1^n)\right] - Pr\left[D^{f(.),f^{-1}(.)}(1^n)\right]\right| \leq \mu(n)$$

Pseudorandom Permutation

Definition 3.28: A keyed function $\mathrm{F}: \{0,1\}^n \times \{0,1\}^n \rightarrow \{0,1\}^n$ is a **strong pseudorandom permutation** if for all PPT distinguishers D there is a negligible function μ s.t.

$$\left|Pr\left[D^{F_k(.),F_k^{-1}(.)}(1^n)\right] - Pr\left[D^{f(.),f^{-1}(.)}(1^n)\right]\right| \leq \mu(n)$$

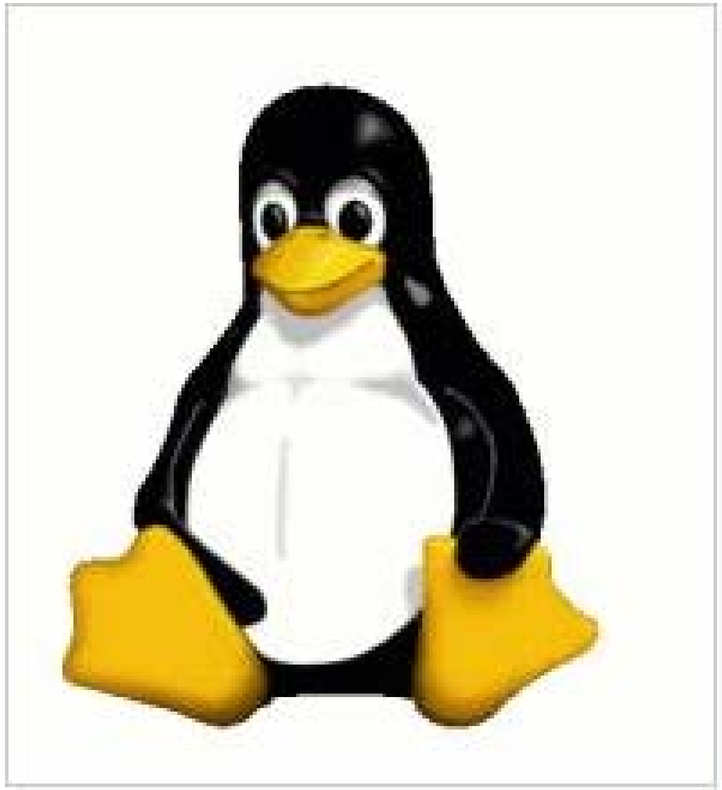
Notes:

- the first probability is taken over the uniform choice of $k \in \{0,1\}^n$ as well as the randomness of D.
- the second probability is taken over uniform choice of f $\in$ $\mathbf{Perm_n}$ as well as the randomness of D.
- D is *never* given the secret k
- However, D is given oracle access to (keyed) permutation and inverse

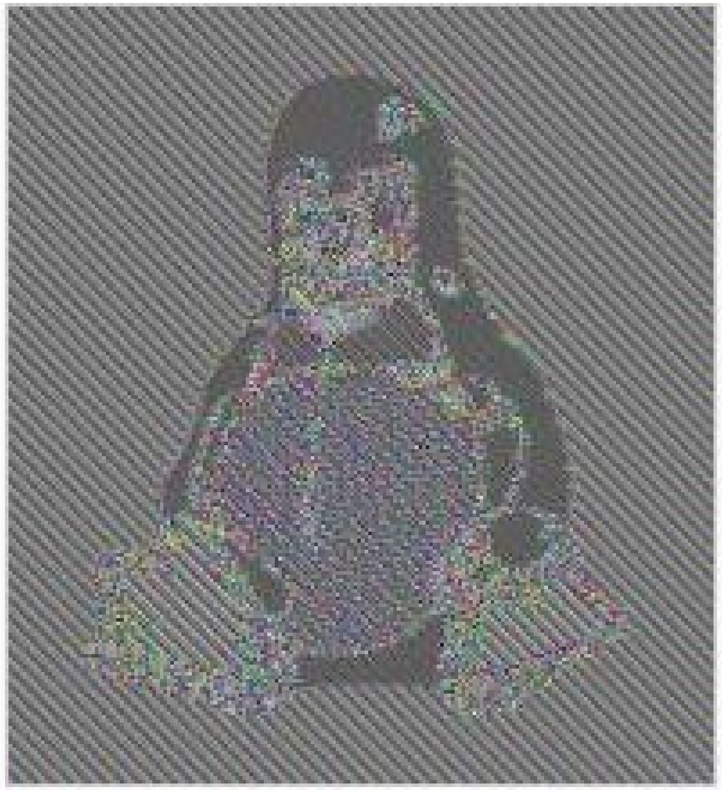
Electronic Code Book (ECB) Mode

- Uses strong PRP $F_k(x)$ and $F_k^{-1}(x)$
- Enc_k
  - **Input**: $m_1,\ldots,m_\ell$
  - **Output**: $\langle F_k(m_1), \ldots, F_k(m_\ell)\rangle$
- How to decrypt?
- Is this secure?
- **Hint**: Encryption is deterministic.
  - **Implication**: Not CPA-Secure
  - But, it gets even worse

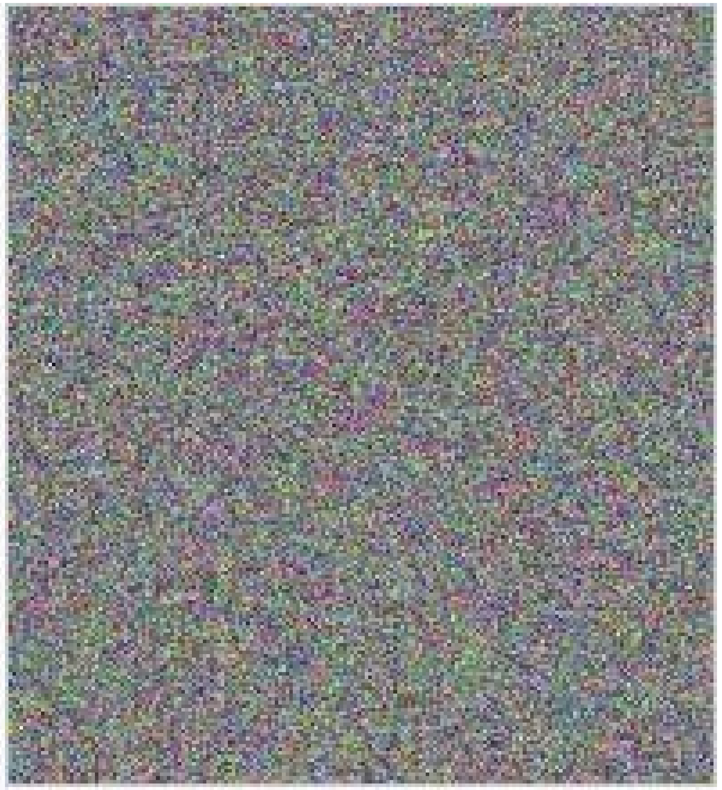
ECB Mode (A Failed Approach)



Original image



Encrypted using ECB mode



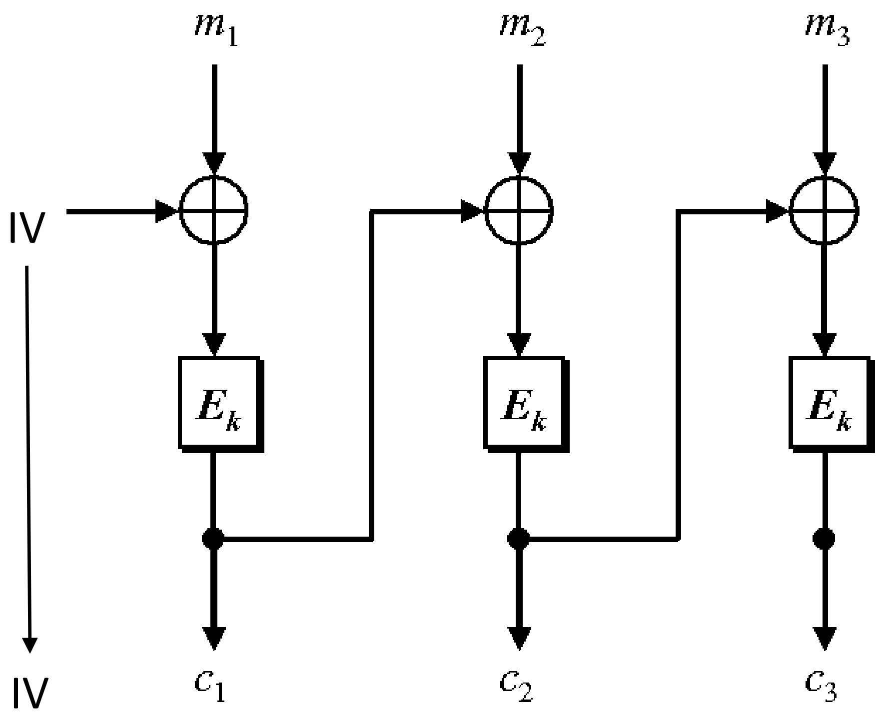
Modes other than ECB result in pseudo-randomness

The Penguin Principle

If you can still see the penguin after "encrypting" the image something is very very wrong with the encryption scheme.

Cipher Block Chaining

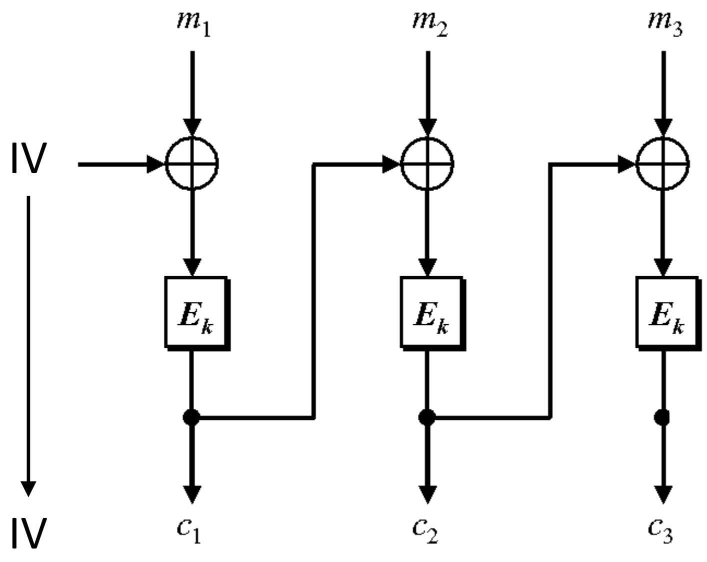
- CBC-Mode (below) is CPA-secure if E_k is a PRP

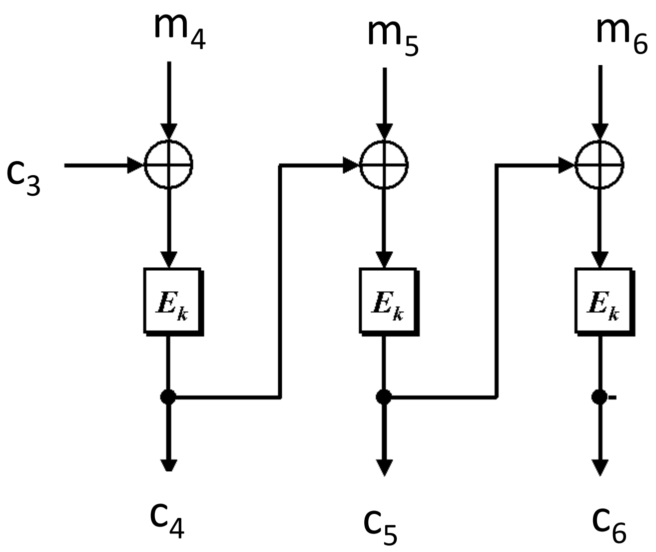


Reduces bandwidth!

Message: 3n bits
Ciphertext: 4n bits

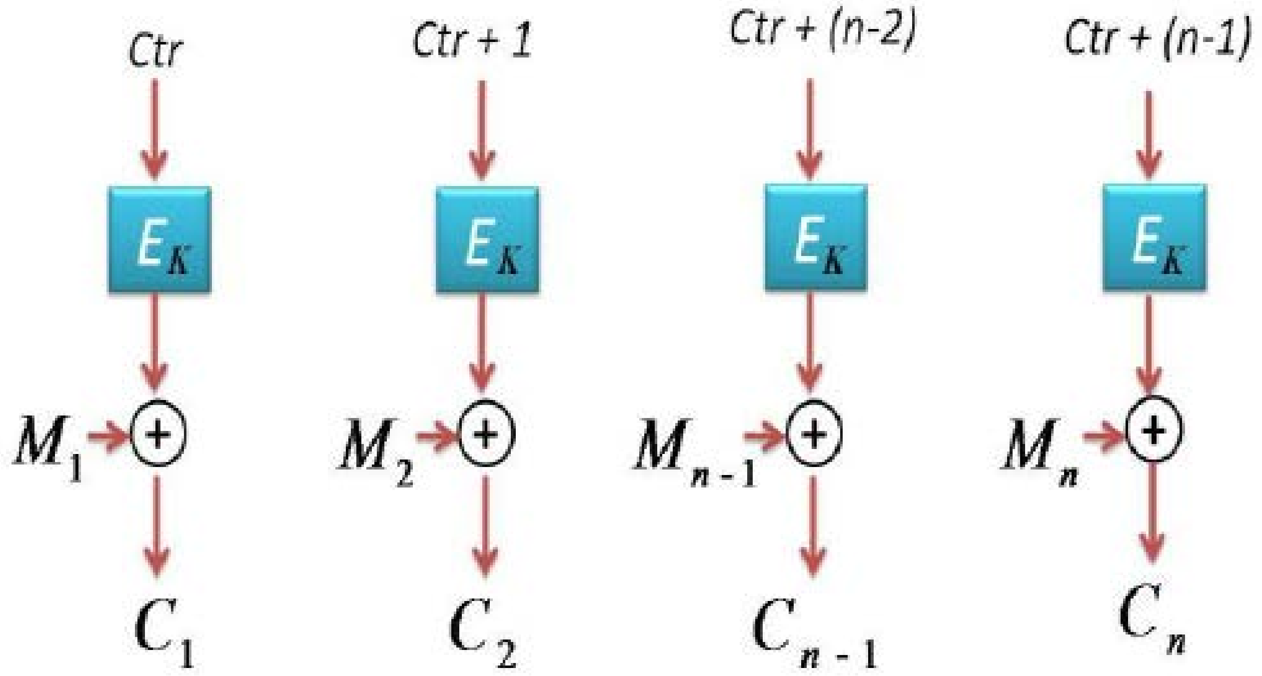
Chained CBC-Mode





- First glance: seems similar to CBC-Mode and reduces bandwidth
- Vulnerable to CPA-Attack! (Set $\mathrm{m}_4 = \mathrm{IV} \oplus c_3 \oplus m_1'$ and c_4=c_1 iff m_1=m_1’)
- **Moral**: Be careful when tweaking encryption scheme!

Counter Mode



- Input: $m_1,\dots,m_n$
- Output: $c = (ctr, c_1,c_2,\dots,c_n)$ where ctr is chosen uniformly at random
- **Theorem**: If E_k is PRF then counter mode is CPA-Secure
- **Advantages**: Parallelizable encryption/decryption

Week 3: Topic 3: CCA-Security

Chosen Ciphertext Attacks

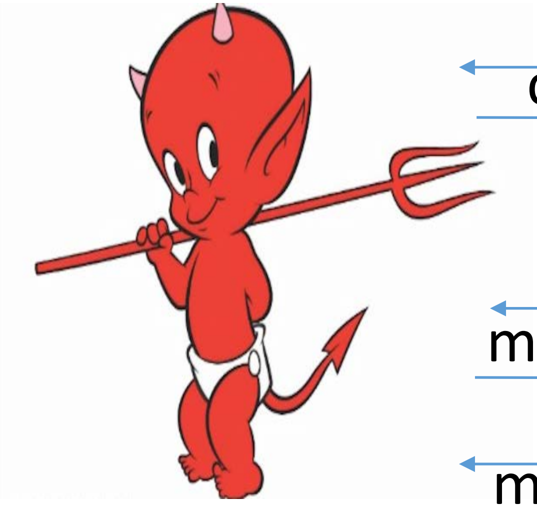
- Sometimes an attacker has ability to obtain (partial) decryptions of ciphertexts of its choice.
- CPA-Security does not model this ability.

Examples:

- An attacker may learn that a ciphertext corresponds to an ill-formed plaintext based on the reaction (e.g., server replies with "invalid message").
- Monitor enemy behavior after receiving and encrypted message.
- **Authentication Protocol:** Send $Enc_k(r)$ to recipient who authenticates by responding with r.

CCA-Security (Ind-CC

We could set $m_0 = m_{-1}$ or $m_1 = m_{-2}$



m_{-1}

$c_{-1} = Enc_K(m_{-1})$

c_{-2}

$m_{-2} = Dec_K(c_{-2})$

⋮

m_0, m_1

$c = Enc_K(m_b)$

m_3

$c_3 = Enc_K(m_3)$

c_3

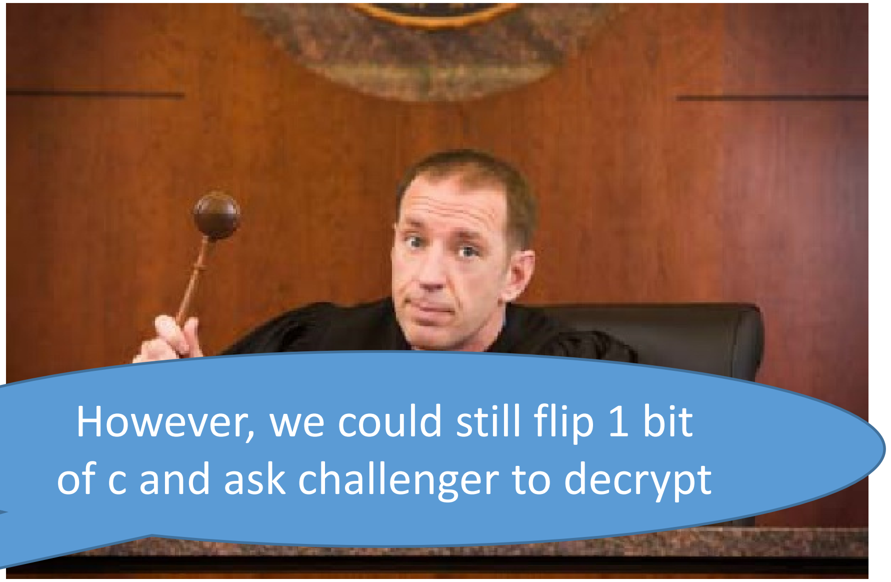
$m_3 = Dec_K(m_3)$

$c_4 = c$

⋮

“No Way!”

b'



However, we could still flip 1 bit of c and ask challenger to decrypt

Random bit b

K = Gen(.)

CCA-Security $\left(PrivK_{A,\Pi}^{cca}(n)\right)$

1. Challenger generates a secret key k and a bit b
2. Adversary (A) is given oracle access to Enc_k and Dec_k
3. Adversary outputs m_0,m_1
4. Challenger sends the adversary c=$Enc_k(m_b)$.
5. Adversary maintains oracle access to Enc_k and Dec_k ,however the adversary is not allowed to query $Dec_k(c)$.
6. Eventually, Adversary outputs b'.

$$PrivK_{A,\Pi}^{cca}(n) = 1 \text{ if } \mathrm{b} = \mathrm{b}'; \text{otherwise } 0.$$

CCA-Security: For all PPT A exists a negligible function negl(n) s.t.

$$\Pr\left[PrivK_{A,\Pi}^{cca}(n) = 1\right] \le \frac{1}{2} + negl(n)$$

CCA-Security

Definition 3.33: An encryption scheme Π is CCA-secure if for all PPT A there is a negligible function negl(n) such that

$$\Pr\left[PrivK_{A,\Pi}^{cca}(n) = 1\right] \leq \frac{1}{2} + negl(n)$$

CPA-Security doesn't imply CCA-Security

$$\text{Enc}_k(m) = \langle r, F_k(r) \oplus m \rangle$$

Attacker: Selects $m_0 = 0^n$ and $m_1 = 1^n$

Attacker Receives: $c = \langle r, s \rangle$ where $s = F_k(r) \oplus m_b$

Attacker Queries: $\text{Dec}_k(c')$ for

$$c' = \langle r, s \oplus 10^{n-1} \rangle$$

Attacker Receives: 10^{n-1} (if b=0) or 01^{n-1} (if b=1)

Example Shows: CPA-Security doesn't imply **CCA** Security (Why?)

Attacks in the Wild

- Padding Oracle Attack
- Length of plaintext message must be multiple of block length
- Popular fix PKCS #5 padding
  - 4 bytes of padding (0x04040404)
  - 3 bytes of padding (0x030303)
- “Bad Padding Error”
  - Adversary submits ciphertext(s) and waits to if this error is produced
  - Attacker can repeatedly modify ciphertext to reveal original plaintext piece by piece!

Example

M="hello…please keep this message secret"+0x030303

$$\mathrm{C} = \langle r, s = F_k(r) \oplus m \rangle$$

- $\mathrm{C}' = \langle r, F_k(r) \oplus m \oplus 0\text{x}0000 \ldots .30000 \rangle$

Ask to decrypt C'

- If we added < 3 bits of padding C' can be decrypted.
- Otherwise, we will get a decryption error.

Once we know we have three bits of padding we can set

$$\mathrm{C}'' = \langle r, s = F_k(r) \oplus 0\text{x}0000 \ldots .30303 \oplus 0\text{x}0 \ldots \mathbf{gg}040404 \rangle$$

If C'' decrypts then we can infer the last byte "t" from $\mathbf{gg} \oplus 0\text{x}04$.

CCA-Security

- **Gold Standard**: CCA-Security is strictly stronger than CPA-Security

- If a scheme has indistinguishable encryptions under one chosen-ciphertext attack then it has indistinguishable multiple encryptions under chosen-ciphertext attacks.
- None of the encryption schemes we have considered so far are CCA-Secure ☹
- CCA-Security implies non-malleability (message integrity)
  - An attacker who modifies a ciphertext c produces c’ which is either
    - Invalid, or
    - Decrypted message is unrelated to original message

CPA-Secure Encryption

$$\mathrm{Enc}_k(\mathrm{m}) = \langle r, F_k(r) \oplus m \rangle$$

$$\mathrm{Dec}_k(\langle r, s \rangle) = F_k(r) \oplus s$$

Drawbacks:

- Encryption is for fixed length messages only
- Length of ciphertext is twice as long as message
- Attacker can still tamper with ciphertexts to flip bits of plaintext

Stream Ciphers/Block Ciphers

CCA Security

Stream Ciphers Modes

- What if we don't know the length of the message to be encrypted a priori?
  - Stream Cipher: $G_\infty(s, 1^n)$ outputs n pseudorandom bits as follows
  - **Initial State**: st_0 = **Initialize**(s)
  - **Repeat**
    - (y_i, st_i)=GetBits(st_{i-1})
    - Output y_i
- **Synchronized Mode**
  - Message sequence: $m_1, m_2, \ldots$
  - Ciphertext sequence: $c_i = m_i \oplus y_i$ (same length as ciphertext!)
  - "CPA-like" security follows from cipher security (must stop after n-bits)
  - Deterministic encryption, what gives???
  - Requires both parties to maintain state (not good for sporadic communication)

Stream Ciphers Modes

- What if we don't want to keep state?
- **Unsynchronized Mode**
  - Message sequence: $m_1, m_2, \ldots$
  - Ciphertext sequence: $c_i = \langle IV, m_i \oplus G_\infty(s, IV, 1^{|m_i|}) \rangle$
  - CPA-Secure if $F_k(IV) = G_\infty(k, IV, 1^n)$ is a (weak) PRF.
  - No shared state, but longer ciphertexts….

Next Class

- Read Katz and Lindell 4.1-4.2
- Message Authentication Codes (MACs) Part 1

Week 3: Topic 4: Message Authentication Codes (Part 1)

Recap

- CPA-Security vs. CCA-Security
- PRFs

Today's Goals:

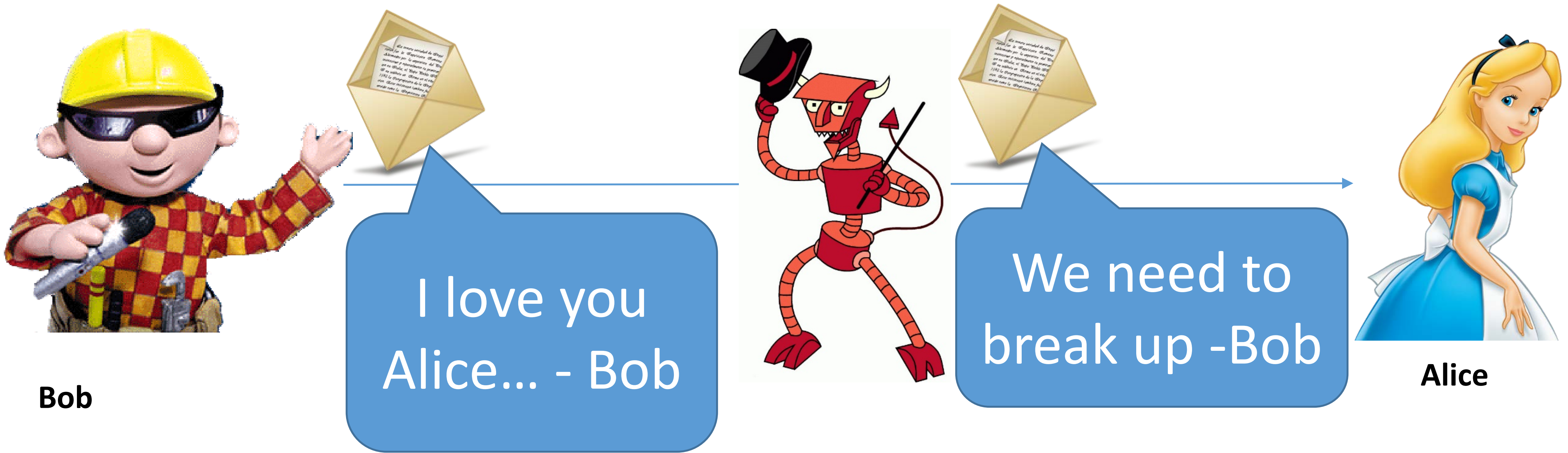
- Introduce Message Authentication Codes (MACs)
    - Key tool in Construction of CCA-Secure Encryption Schemes
- ~~Build Secure MACs~~

What Does It Mean to "Secure Information"

- Confidentiality (Security/Privacy)
  - Only intended recipient can see the communication

What Does It Mean to “Secure Information”

- Confidentiality (Security/Privacy)
  - Only intended recipient can see the communication
- Integrity (Authenticity)
  - The message was actually sent by the alleged sender



Message Authentication Codes

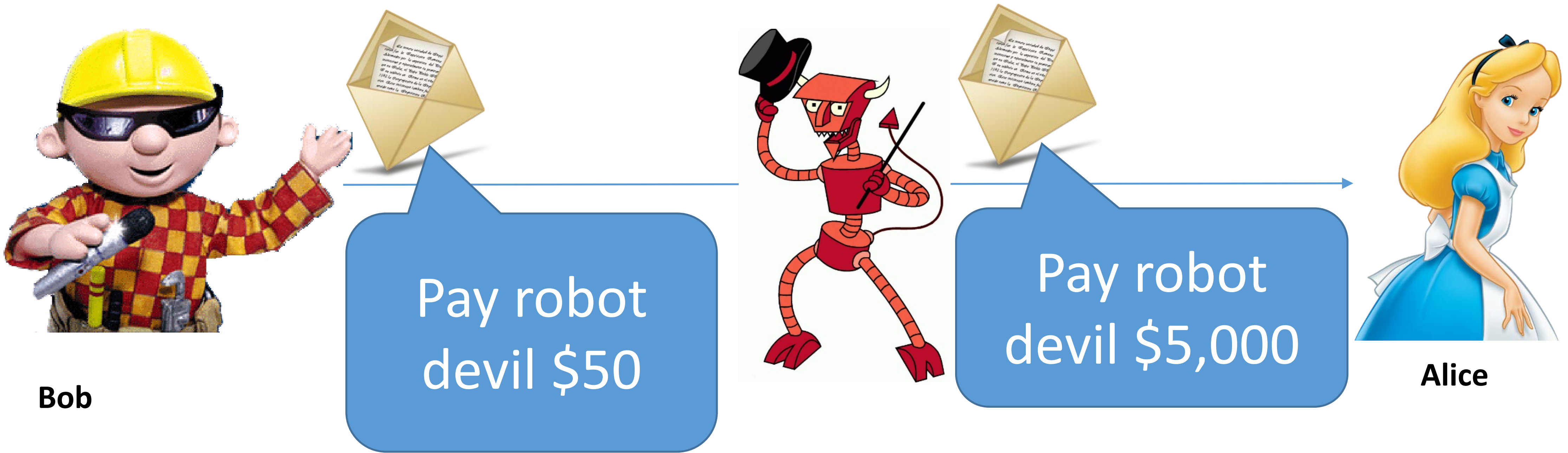
- CPA-Secure Encryption: Focus on Secrecy
  - But does not promise integrity

- Message Authentication Codes: Focus on Integrity
  - But does not promise secrecy

- CCA-Secure Encryption: Requires Integrity and Secrecy

What Does It Mean to “Secure Information”

- Integrity (Authenticity)
  - The message was actually sent by the alleged sender
  - And the received message matches the original



Error Correcting Codes?

- Tool to detect/correct a *small* number of random errors in transmission

- **Examples:** Parity Check, Reed-Solomon Codes, LDPC, Hamming Codes …

- Provides no protection against a malicious adversary who can introduce an arbitrary number of errors

- Still useful when implementing crypto in the real world (Why?)

Modifying Ciphertexts

$$\text{Enc}_\text{k}(\text{m}) = c = \langle r, F_k(r) \oplus m \rangle$$

$$c' = \langle r, F_k(r) \oplus m \oplus y \rangle$$

$$\text{Dec}_\text{k}\,(c') = F_k(r) \oplus F_k(r) \oplus m \oplus y = m \oplus y$$

If attacker knows original message he can forge c’ to decrypt to any message he wants.

Even if attacker doesn’t know message he may find it advantageous to flip certain bits (e.g., decimal places)

Message Authentication Code Syntax

Definition 4.1: A message authentication code (MAC) consists of three algorithms

- $\mathrm{Gen}(1^n; R)$ (Key-generation algorithm)
  - Input: security parameter 1^n (unary) and random bits R
  - Output: Secret key $\mathrm{k} \in \mathcal{K}$
- $\mathrm{Mac_k}(m; R)$ (Tag Generation algorithm)
  - Input: Secret key $\mathrm{k} \in \mathcal{K}$ and message $\mathrm{m} \in \mathcal{M}$ and random bits R
  - Output: a tag t
- $\mathrm{Vrfy_k}(m, t)$ (Verification algorithm)
  - Input: Secret key $\mathrm{k} \in \mathcal{K}$, a message m and a tag t
  - Output: a bit b (b=1 means "valid" and b=0 means "invalid")

- Invariant?

Message Authentication Code Syntax

Definition 4.1: A message authentication code (MAC) consists of three algorithms

- $\mathrm{Gen}(1^n; R)$ (Key-generation algorithm)
  - Input: security parameter 1^n (unary) and random bits R
  - Output: Secret key $\mathrm{k} \in \mathcal{K}$
- $\mathrm{Mac}_{\mathrm{k}}(m; R)$ (Tag Generation algorithm)
  - Input: Secret key $\mathrm{k} \in \mathcal{K}$ and message $\mathrm{m} \in \mathcal{M}$ and random bits R
  - Output: a tag t
- $\mathrm{Vrfy}_{\mathrm{k}}(m, t)$ (Verification algorithm)
  - Input: Secret key $\mathrm{k} \in \mathcal{K}$, a message m and a tag t
  - Output: a bit b (b=1 means “valid” and b=0 means “invalid”)

- Invariant?

Message Authentication Code Syntax

Definition 4.1: A message authentication code (MAC) consists of three algorithms $\Pi = (\mathrm{Gen}, \mathrm{Mac}, \mathrm{Vrfy})$

- $\mathrm{Gen}(1^n; R)$ (Key-generation algorithm)
  - Input: security parameter 1^n (unary) and random bits R
  - Output: Secret key $\mathrm{k} \in \mathcal{K}$
- $\mathrm{Mac}_{\mathrm{k}}(m; R)$ (Tag Generation algorithm)
  - Input: Secret key $\mathrm{k} \in \mathcal{K}$ and message $\mathrm{m} \in \mathcal{M}$ and random bits R
  - Output: a tag t
- $\mathrm{Vrfy}_{\mathrm{k}}(m, t)$ (Verification algorithm)
  - Input: Secret key $\mathrm{k} \in \mathcal{K}$, a message m and a tag t
  - Output: a bit b (b=1 means "valid" and b=0 means "invalid")

$$\mathrm{Vrfy}_{\mathrm{k}}(m, \mathrm{Mac}_{\mathrm{k}}(m; R)) = 1$$

General vs Fixed Length MAC

$$\mathcal{M} = \{0,1\}^*$$

versus

$$\mathcal{M} = \{0,1\}^{\ell(n)}$$

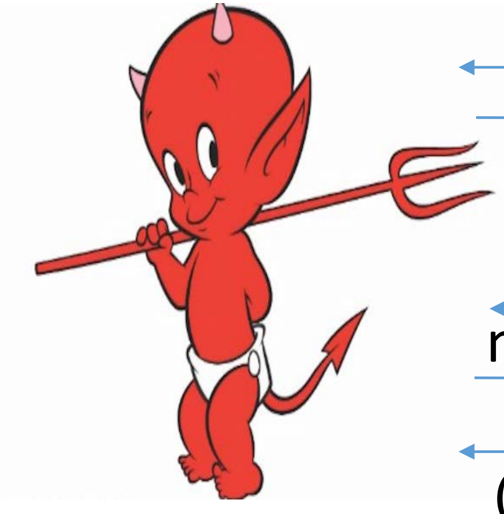
Deterministic MACs

- **Canonical Verification Algorithm**

$$\mathrm{Vrfy_k}(m,t) = \begin{cases} 1 & \text{if } t = \mathrm{Mac_k}(m) \\ 0 & \text{otherwise} \end{cases}$$

- "All real-world MACs use canonical verification" – page 115

MAC Authentication Game $\left(\text{Macforge}_{A,\Pi}(n)\right)$



m_1

$\mathbf{t_1 = Mac_K(m_1)}$

m_2

$\mathbf{t_2 = Mac_K(m_2)}$

⋮

m_q

$\mathbf{t_q = Mac_K(m_q)}$

(m,t) s.t $m \notin \{m_1, \ldots, m_q\}$

$\text{Macforge}_{A,\Pi}(n) = \text{Vrfy}_{\text{k}}(m,t)$

K = Gen(.)

$$\forall PPT\ A\ \exists \mu\ (\text{negligible})\ \text{s.t}$$
$$\Pr\left[\text{Macforge}_{A,\Pi}(n) = 1\right] \le \mu(n)$$

Discussion

- Is the definition too strong?
  - Attacker wins if he can forge any message
  - Does not necessarily attacker can forge a “meaningful message”
  - “Meaningful Message” is context dependent
  - Conservative Approach: Prove Security against more powerful attacker
  - Conservative security definition can be applied broadly
- Replay Attacks?
  - **t=Mac$_K$(“Pay Bob \$1,000 from Alice’s bank account”)**
  - Alice cannot modify message to say \$10,000, but…
  - She may try to replay it 10 times

Replay Attacks

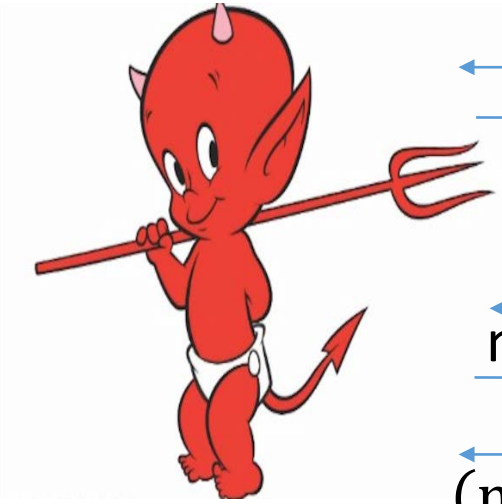
- MACs alone do not protect against replay attacks (they are stateless)

- Common Defenses:
  - Include Sequence Numbers in Messages (requires synchronized state)
    - Can be tricky over a lossy channel
  - Timestamp Messages
    - Double check timestamp before taking action

Strong MACs

- Previous game ensures attacker cannot generate a valid tag for a new message.
- However, attacker may be able to generate a second valid tag t' for a message m after observing (m,t)
- Strong MAC: attacker cannot generate second valid tag, even for a known message

Strong MAC Authentication ($\mathrm{Macsforge}_{A,\Pi}(n)$)



m_1

t_1 = Mac$_K$(m_1)

m_2

t_2 = Mac$_K$ (m_2)

⋮

m_q

t_q = Mac$_K$(m_q)

(m, t) s.t $(\mathrm{m}, \mathrm{t}) \notin \{(\mathrm{m}_1, \mathrm{t}_1), \ldots, (\mathrm{m}_q, \mathrm{t}_q)\}$

$\mathrm{Macsforge}_{A,\Pi}(n) = \mathrm{Vrfy}_{\mathrm{k}}(m, t)$

K = Gen(.)

$$\forall PPT\ A\ \exists \mu\ (\text{negligible})\ \text{s.t}$$
$$\Pr[\mathrm{Macsforge}_{A,\Pi}(n) = 1] \leq \mu(n)$$

Strong MAC vs Regular MAC

Proposition 4.4: Let $\Pi = (\mathrm{Gen}, \mathrm{Mac}, \mathrm{Vrfy})$ be a secure MAC that uses canonical verification. Then Π is a strong MAC.

"All real-world MACs use canonical verification" – page 115

Should attacker have access to $\mathbf{Vrfy_K}$(.) oracle in games?

(e.g., CPA vs CCA security for encryption)

Irrelevant if the MAC uses canonical verification!

Timing Attacks (Side Channel)

Naïve Canonical Verification Algorithm

Input: m,t'

t=Mac_K(m)

for i=1 to tag-length

  if t[i] != t'[i] **then**

    return 0

return 1

Example

t= 1 0 1 0 1 1 1 0

t'= 1 0 1 0 1 0 1 1

Returns 0 after 8 steps

Timing Attacks (Side Channel)

Naïve Canonical Verification Algorithm

Input: m,t'

$t = Mac_K(m)$

for i=1 to tag-length

  if t[i] != t'[i] **then**

    return 0

return 1

Example

t= 1 0 1 0 1 1 1 0

t'= 0 0 1 0 1 0 1 0

Returns 0 after 1 step

Timing Attack

- MACs used to verify code updates for Xbox 360
- Implementation allowed different rejection times (side-channel)
- Attacks exploited vulnerability to load pirated games onto hardware
- **Moral**: Ensure verification is time-independent

Improved Canonical Verification Algorithm

Input: m,t’

B=1

t=Mac$_K$(m)

for i=1 to tag-length

  if t[i] != t’[i] **then**

    B=0

  else (dummy op)

return B

Example

t= 1 0 1 0 1 1 1 0

t’= 0 0 1 0 1 0 1 0

Returns 0 after 8 steps

Side-Channel Attacks

- Cryptographic Definition
  - Attacker only observes outputs of oracles (Enc, Dec, Mac) and nothing else
- When attacker gains additional information like timing (not captured by model) we call it a side channel attack.

Other Examples

- Differential Power Analysis
- Cache Timing Attack
- Power Monitoring
- Acoustic Cryptanalysis
- …many others

Next Class

- Read Katz and Lindell 4.3
- Message Authentication Codes (MACs) Part 2
  - Constructing Secure MACs