# 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

**Fall 2018**

## Recap

- Using PRGs to achieve Semantic Security (Single Message Eavesdropping)
- Multiple Message Eavesdropping Experiment
  - Impossible to achieve with stateless/deterministic encryption scheme
- *Chosen Plaintext Attacks* and CPA-Security

## Recap CPA-Security

- Defend against evesdropping attacker's ability to <u>influence</u> messages that honest party encrypts
  - More powerful than *known plaintext attacks* (<u>knows</u> vs <u>controls</u> encrypted message)

- Historical Importance: Battle of Midway

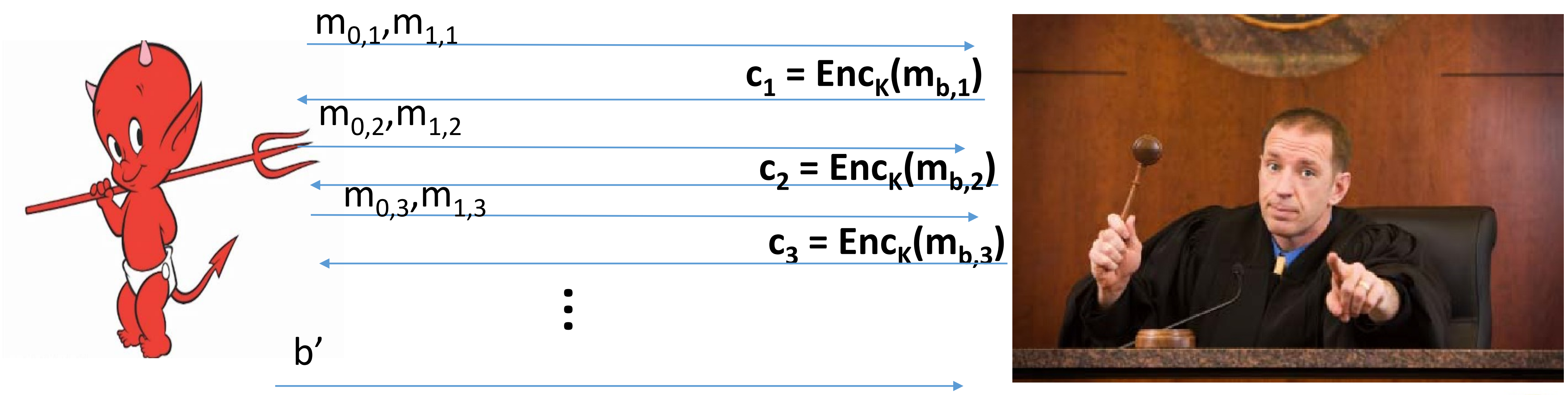
- CPA-Security Equivalence
  - Multiple vs Single Encryption Game

- Limitations of Threat Model
  - Passive vs Active Attacker
  - What if attacker can get honest party to (partially) decrypt some messages?

## Chosen Plaintext Attacks (Examples)

- CPA-attacker <u>influences</u> messages that honest party encrypts
- Eve sends Bob a document/e-mail expecting that Bob will encrypt it and forward it to Alice
- Eve registers herself in a database expecting that Bob (employee) will forward the encrypted database to her boss.
- Eve generate important news that Bob will encrypt and pass on to Alice
  - Plant objects at specific GPS coordinates
  - Broadcast Message (Battle of Midway)

# CPA-Security (Multiple Messages)


$m_{0,1}, m_{1,1}$
$c_1 = \mathrm{Enc}_K(m_{b,1})$
$m_{0,2}, m_{1,2}$
$c_2 = \mathrm{Enc}_K(m_{b,2})$
$m_{0,3}, m_{1,3}$
$c_3 = \mathrm{Enc}_K(m_{b,3})$
⋮
b'

**Random bit b**

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

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

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

## CPA-Security

**Theorem**: An encryption scheme $\Pi = (Gen, Enc, Dec)$ that is CPA-Secure for single encryptions is also CPA-secure for multiple encryptions.

- We will simply say CPA-security for simplicity
- To show CPA-Security it suffices to show CPA-security for single encryptions.
- To reason about security of a protocol using $\Pi$ we can use game with multiple encryptions.

# CPA-Security

- CPA-security vs Multiple Message Encryption
  - CPA-security is stronger guarantee
  - Attacker can select messages adaptively
- CPA-security: minimal security notion for a modern cryptosystem

- Limitations of CPA-Security: Does not model and adversary who
  - Attempts to modify messages
  - Can get honest party to (partially) decrypt some messages

## 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 $\Pi$' is CPA-secure?

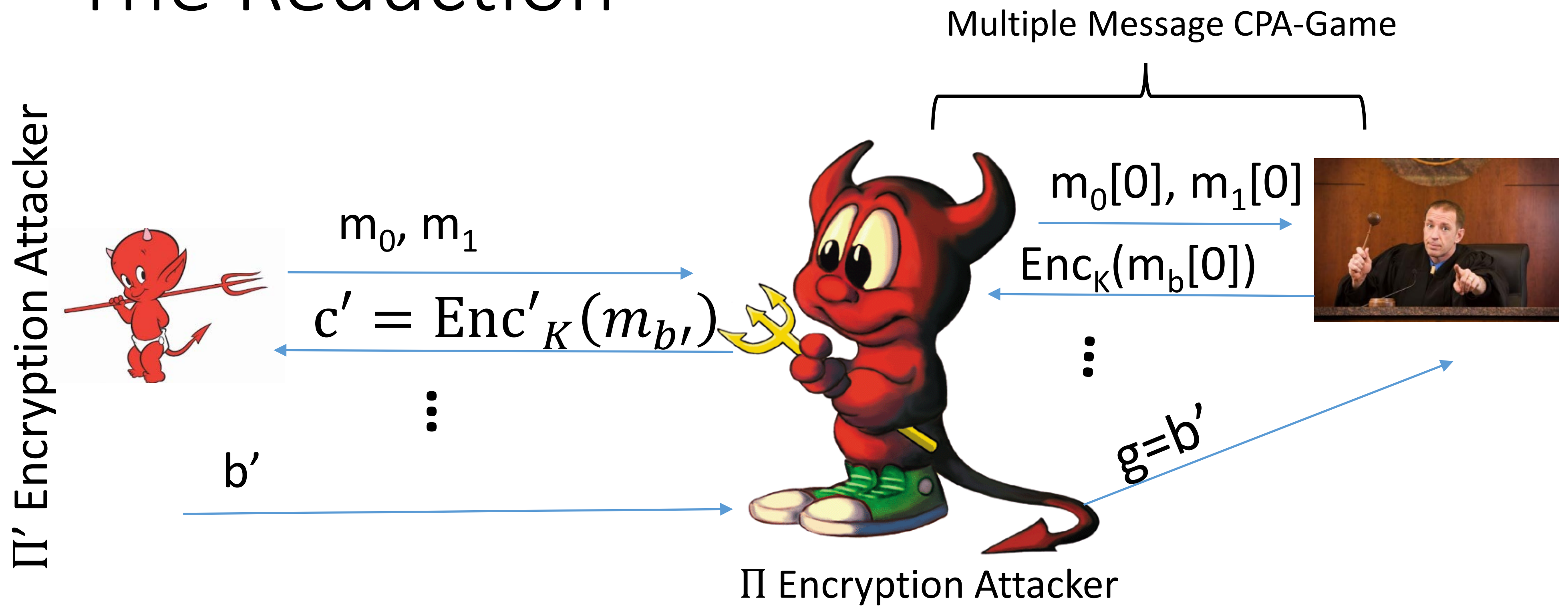
## Security Reduction

- **Step 1:** Assume for contraction that we have a PPT attacker A' that breaks CPA-Security of $\Pi$'.
- **Step 2:** Construct a PPT attacker A that breaks CPA-Security of $\Pi$.

# 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 $\mathrm{F}: \{0,1\}^{\ell_{key}(n)} \times \{0,1\}^{\ell_{in}(n)} \rightarrow \{0,1\}^{\ell_{out}(n)}$, 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 $f: \{0,1\}^n \rightarrow \{0,1\}^n$.

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

$2^n$

| $x$ | $\boldsymbol{f(x)}$ |
|---|---|
| $0 \ldots 00$ | $\boldsymbol{f(0 \ldots 00)}$ |
| $0 \ldots 01$ | $\boldsymbol{f(0 \ldots 01)}$ |
| $0 \ldots 10$ | $\boldsymbol{f(0 \ldots 10)}$ |
| … | … |
| $1 \ldots 11$ | $\boldsymbol{f(1 \ldots 11)}$ |

# 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 prior responses
- Alternate view is often useful in security reductions
  - Doesn’t require time to fully specify $f \in \mathbf{Func_n}$

## 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

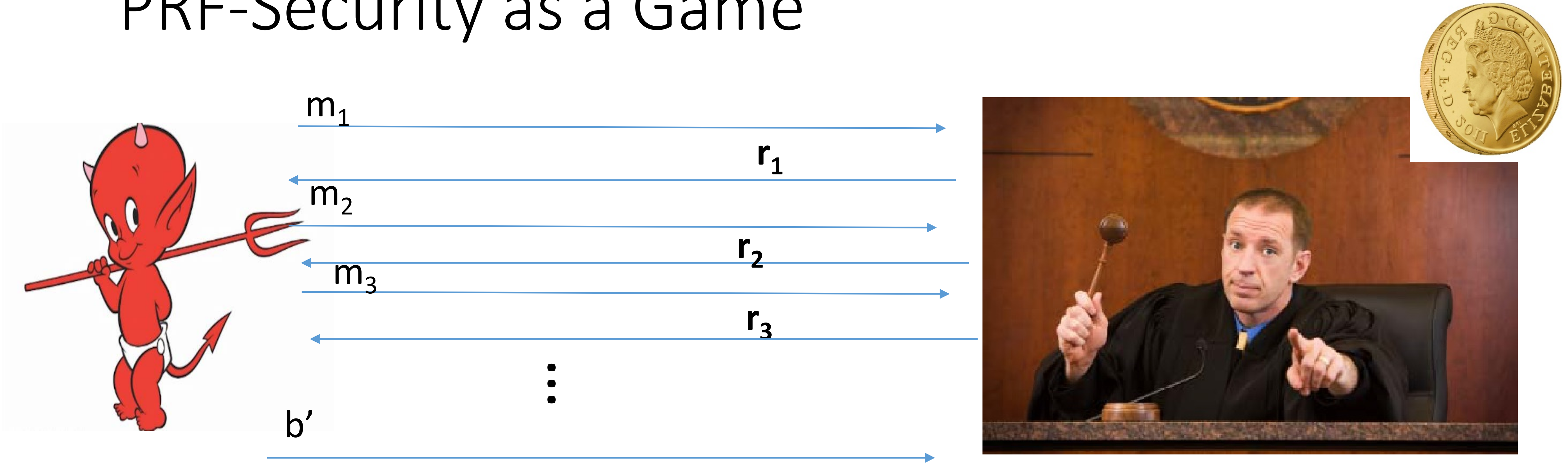
**Definition 3.25:** A keyed function $\mathrm{F}: \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ is a pseudorandom function if for all PPT distinguishers D there is a negligible function $\mu$ s.t.

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

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 ∈**Func**$_{\mathbf{n}}$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



**Random bit b**

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

**Truly random func R**

**$r_i = F_K(m_i)$ if b=1**

**$R(m_i)$ o.w**

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

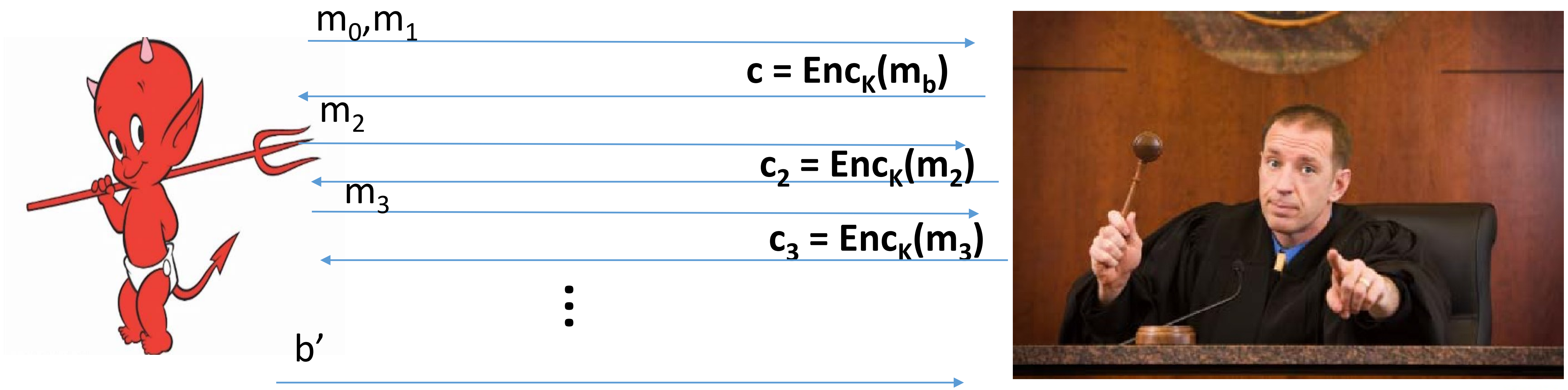
$$\Pr[A\ \ Guesses\ b' = b] \le \frac{1}{2} + \mu(n)$$

## PRF Security Concrete Version

**Definition 3.25:** A keyed function $\mathrm{F}\colon \{0,1\}^n \times \{0,1\}^n \rightarrow \{0,1\}^n$ is a $\big(t(n), q(n), \varepsilon(n)\big)$*-secure pseudorandom function* if **for all** distinguishers D **running in time at most $\boldsymbol{t(n)}$** and **making at most $\boldsymbol{q(n)}$ queries** we have

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

# Reminder: CPA-Security (Single Message)


$m_0, m_1$
$c = \mathrm{Enc}_K(m_b)$
$m_2$
$c_2 = \mathrm{Enc}_K(m_2)$
$m_3$
$c_3 = \mathrm{Enc}_K(m_3)$
⋮
$b'$

**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 $r \in \{0,1\}^n$

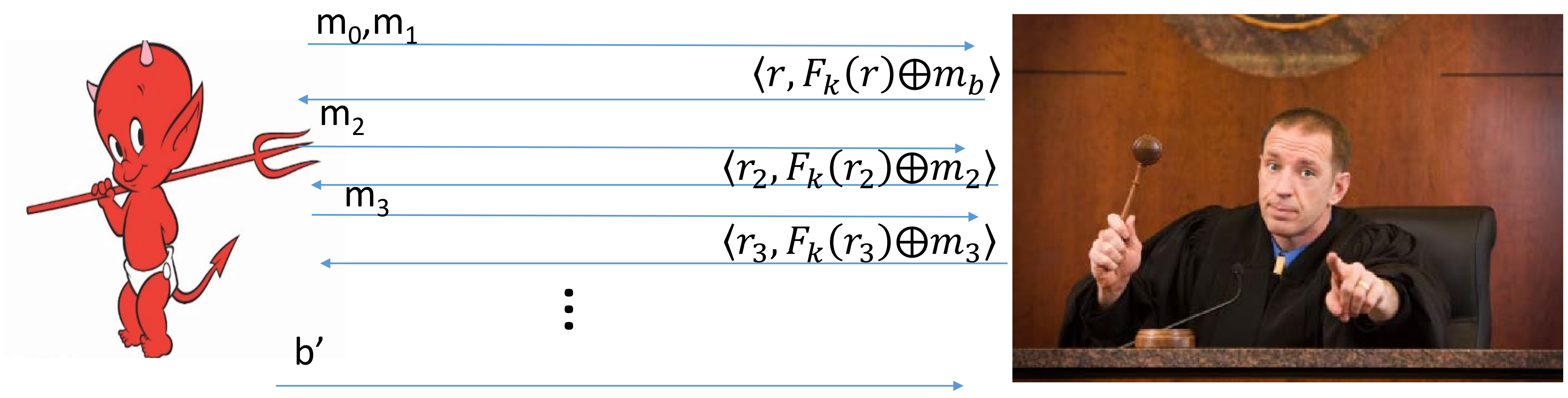
- Dec: Input $k \in \{0,1\}^n$ and $c = \langle r, s \rangle$

  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)


$m_0, m_1$
$\langle r, F_k(r) \oplus m_b \rangle$
$m_2$
$\langle r_2, F_k(r_2) \oplus m_2 \rangle$
$m_3$
$\langle r_3, F_k(r_3) \oplus m_3 \rangle$
⋮
b'

**Random bit b**

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

Assumption: $\exists$ PPT $A$, P $(\text{non}-\text{negligible})$ 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 $\mu$, we have

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

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

## Security Reduction

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

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

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

**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

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

**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 $\mathrm{r}_1$,…,$\mathrm{r}_\mathrm{q}$ 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

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

**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

$$\Pr\left[PrivK_{A,\tilde{\Pi}}^{cpa} = 1\right]$$
$$\leq \Pr\left[PrivK_{A,\tilde{\Pi}}^{cpa} = 1 \middle| \mathrm{r}^* \neq \mathrm{r}_1,\ldots,\mathrm{r}_\mathrm{q}\right] + \Pr\left[\mathrm{r}* \in \left\{\mathrm{r}_1,\ldots,\mathrm{r}_\mathrm{q}\right\}\right]$$
$$\leq \frac{1}{2} + \frac{q(n)}{2^n}$$

## Conclusion

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

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

PRF Security

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

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

**Suggested Exercise:** Work out concrete version of security proof

## 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)

## PRFs from PRGs

**Theorem:** Suppose that there is a PRG G with expansion factor $\ell(n) = 2n$. Then there is a secure PRF.

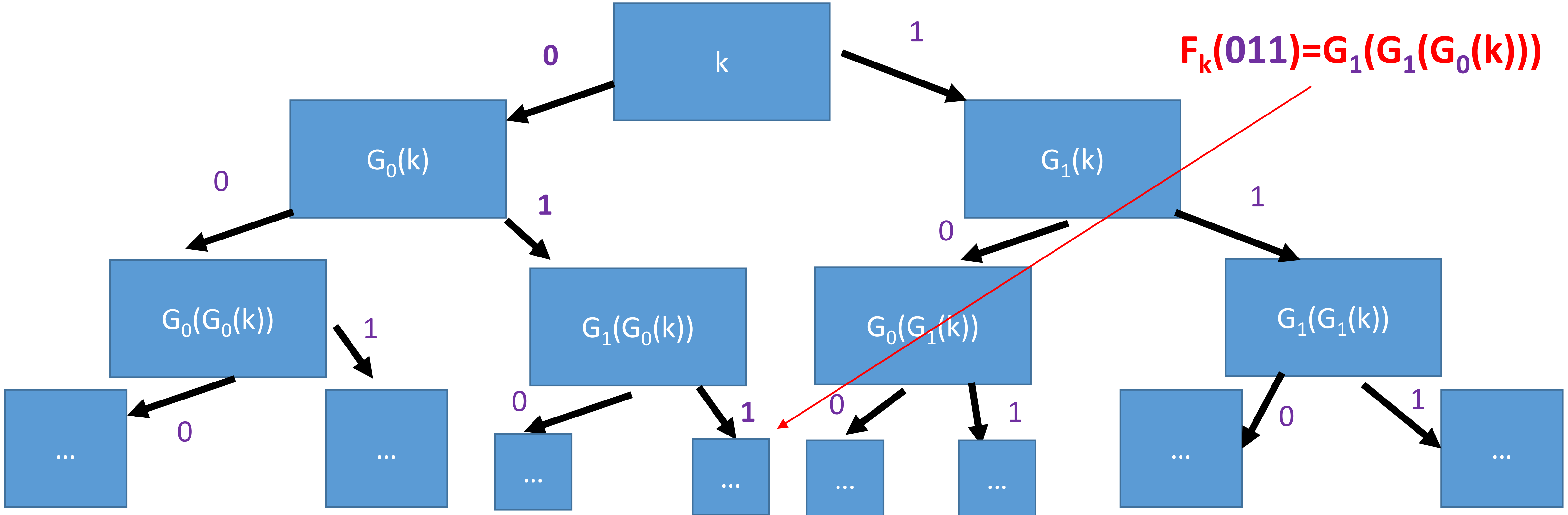
**Let $G(x) = G_0(x)||G_1(x)$ (first/last n bits of output)**

$$F_K(x_1, \dots, x_n) = G_{x_n}\left(\dots\left(G_{x_2}\left(G_{x_1}(K)\right)\right)\dots\right)$$

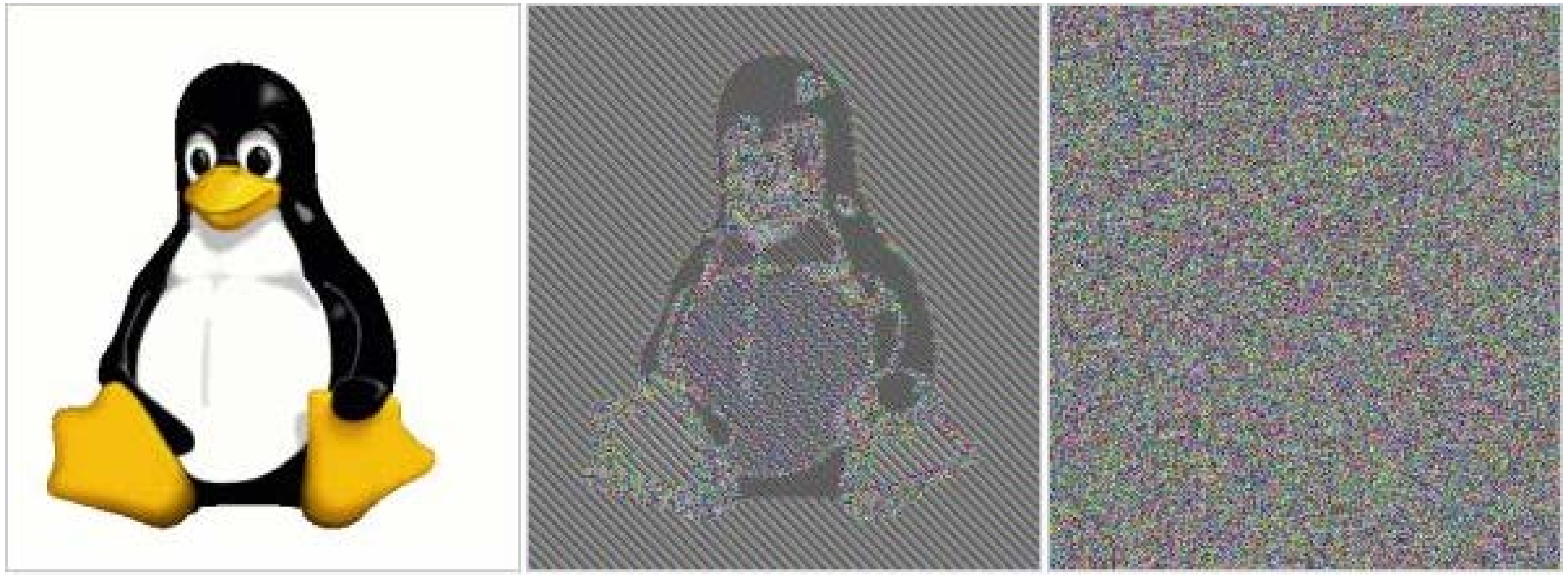
**Theorem:** If G is a PRG then $F_k$ is a PRF

# PRFs from PRGs

**Theorem:** Suppose that there is a PRG G with expansion factor $\ell(n) = 2n$. Then there is a secure PRF.



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

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

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

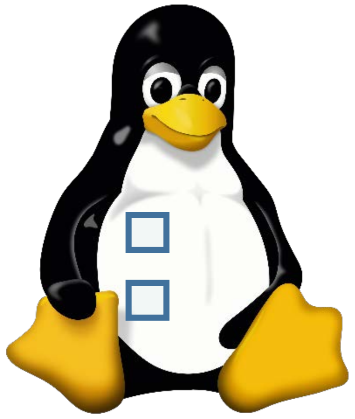
$$\left| Pr\left[\boxed{D^{F_k(.),F_k^{-1}(.)}}(1^n)\right] - Pr\left[\boxed{D^{f(.),f^{-1}(.)}}(1^n)\right]\right| \le \mu(n)$$

Notes:

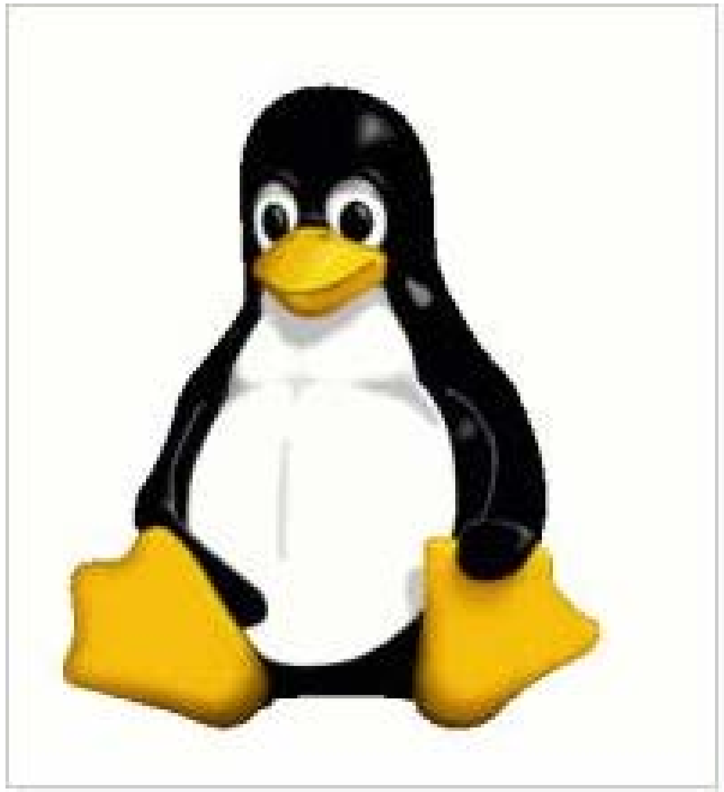
- the first probability is taken over the uniform choice of $k \in \{0,1\}^n$ as well as the randomness of D.
- the second probability is taken over uniform choice of f ∈**Perm$_n$**as well as the randomness of D.
- D is *never* given the secret k
- However, D is given oracle access to (keyed) permutation <u>and inverse</u>
- ~~Strong~~ **pseudorandom permutation**: attacker doesn't get oracle access to inverse
- Can build strong pseudorandom permutation given pseudorandom function

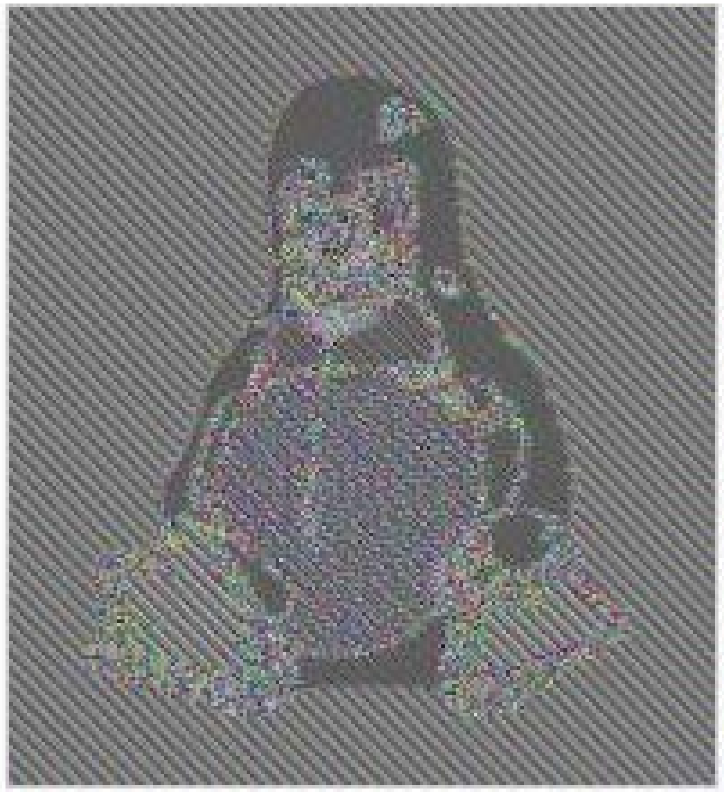
# Electronic Code Book (ECB) Mode

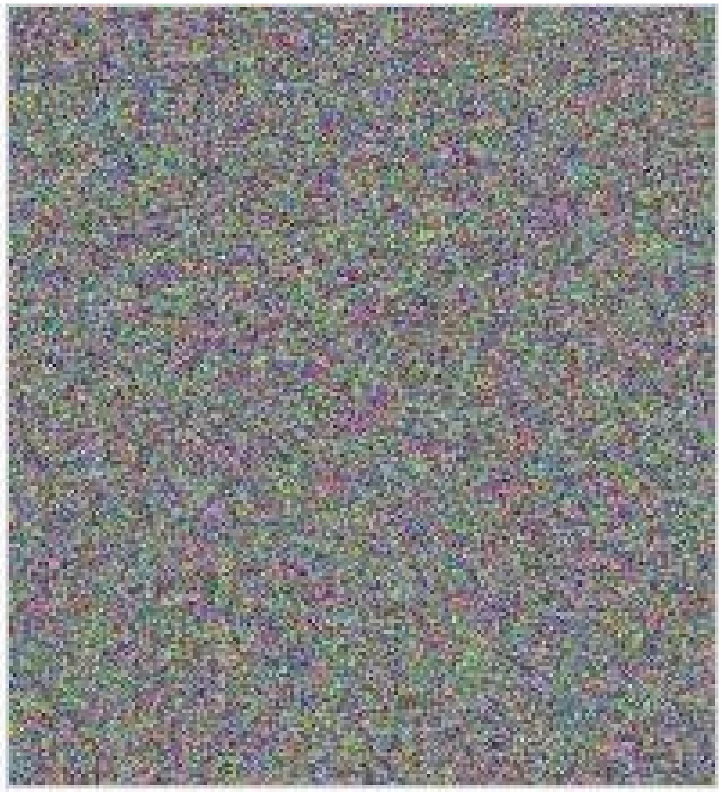
- Uses strong PRP $F_k(x)$ and $F_k^{-1}(x)$
- $Enc_k$
  - **Input**: $m_1,\ldots,m_\ell$
  - **Output**: $\langle F_k(m_1), \ldots, F_k(m_\ell)\rangle$
- How to decrypt?
- Is this secure?
- **Hint**: Encryption is deterministic.
  - **Implication**: Not CPA-Secure
  - But, it gets even worse



## ECB Mode (A Failed Approach)


Original image


Encrypted using ECB mode

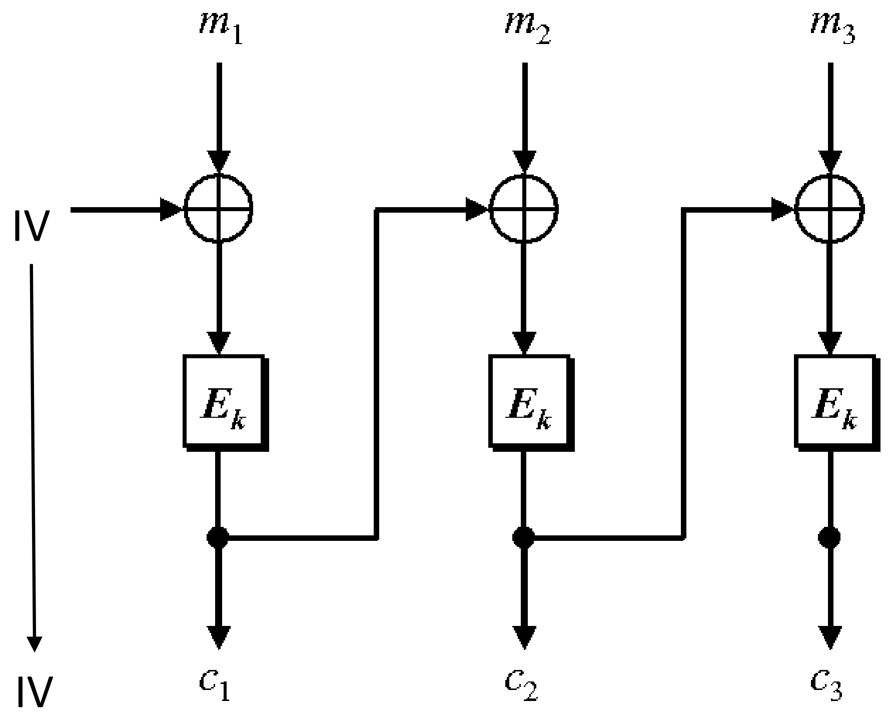

Modes other than ECB result in pseudo-randomness

## 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

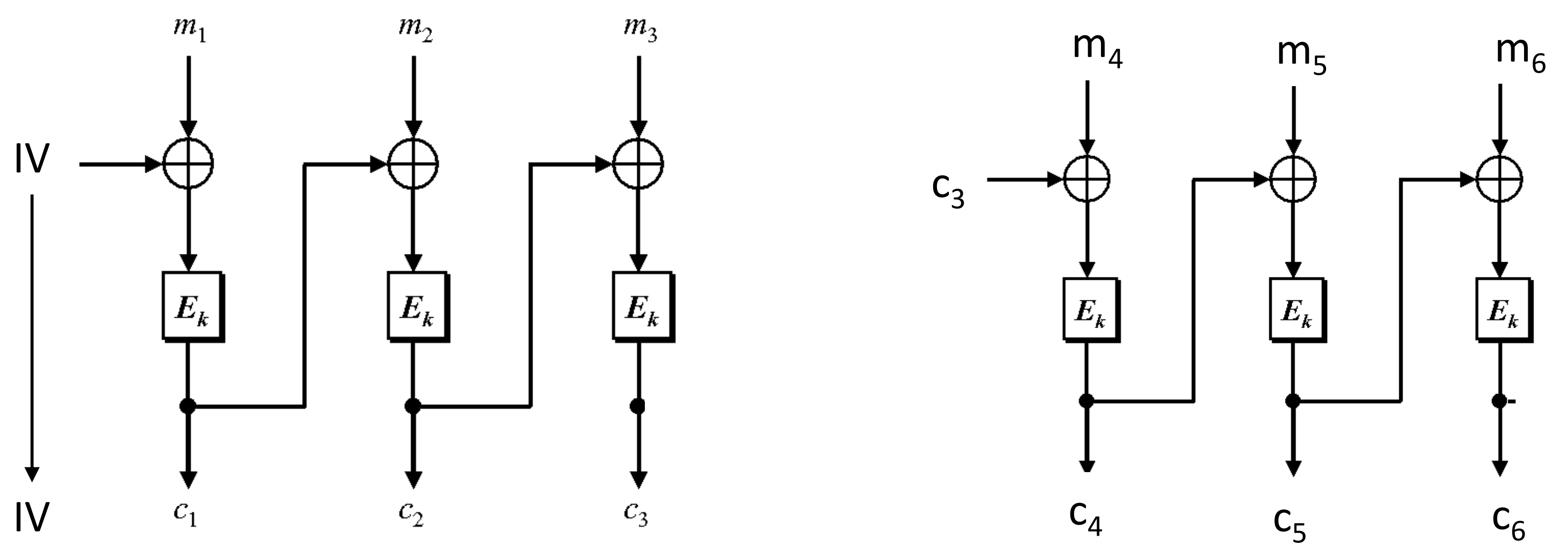
- CBC-Mode (below) is CPA-secure if $E_k$ is a PRP



Reduces bandwidth!

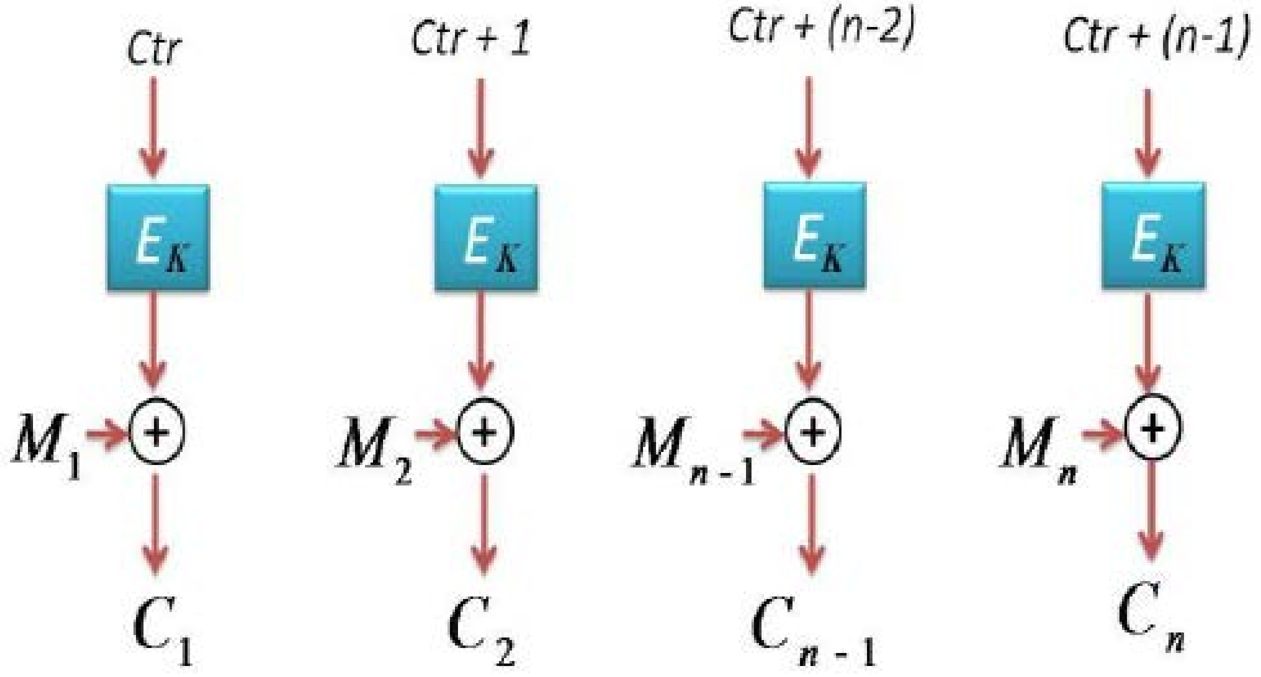
Message: 3n bits
Ciphertext: 4n bits

# 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!

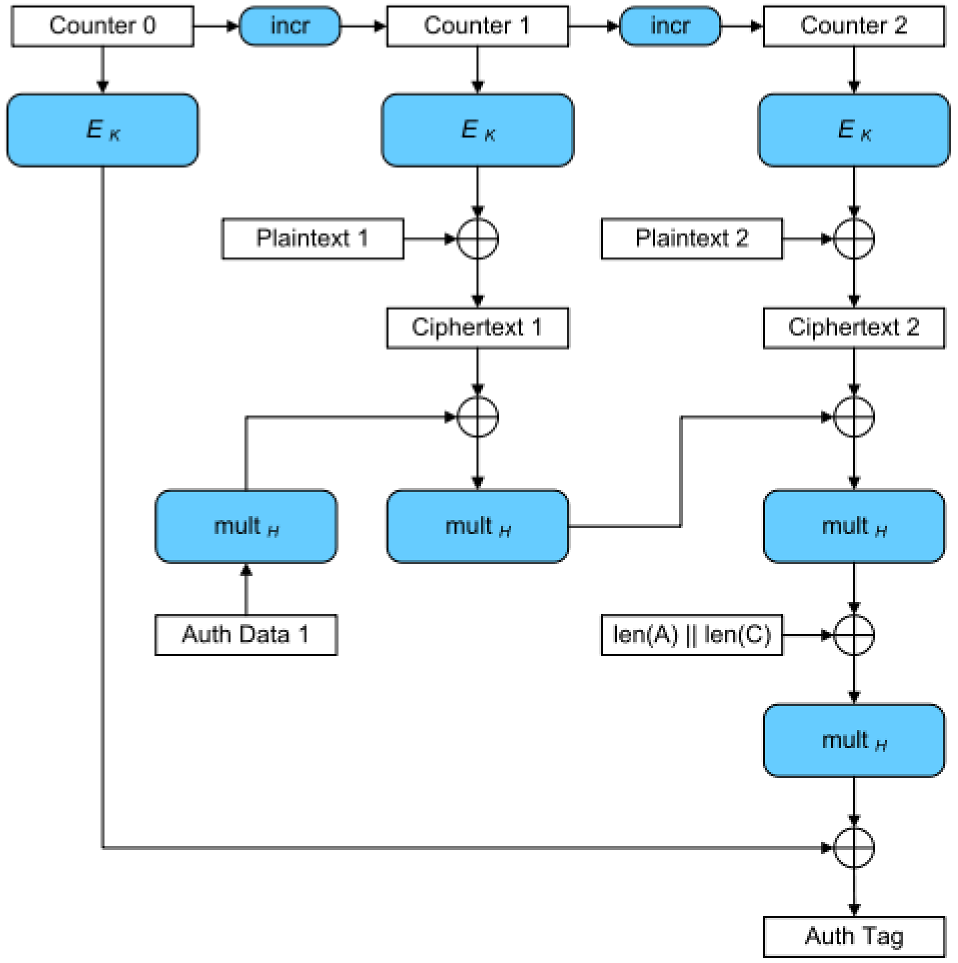
## Counter Mode



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

# Galois Counter Mode (GCM)

- AES-GCM is CCA-secure (> CPA-security)
- **Bonus:** Authentication Encryption with Associated Data
  - Ensure integrity of ciphertext
  - Attacker cannot even generate new/valid ciphertext!
  - Ensures attacker cannot tamper with associated packet data
    - Source IP
    - Destination IP
    - Why can't these values be encrypted?
- Encryption is largely parallelizable!



## Week 3: Topic 3: CCA-Security

## Chosen Ciphertext Attacks

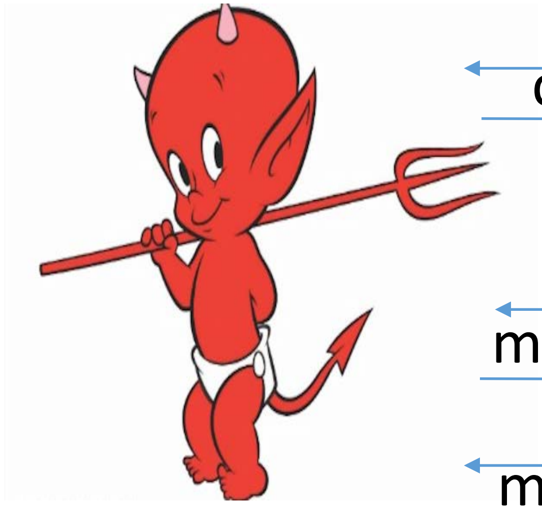
- Sometimes an attacker has ability to obtain (partial) decryptions of ciphertexts of its choice.
- CPA-Security does not model this ability.

**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 an encrypted message.
- **Authentication Protocol:** Send $\mathrm{Enc}_k(r)$ to recipient who authenticates by responding with r.

## CCA-Security (Ind-CCA2)

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



$m_{-1}$

$c_{-1} = \mathrm{Enc}_K(m_{-1})$

$c_{-2}$

$m_{-2} = \mathrm{Dec}_K(c_{-2})$

⋮

$m_0, m_1$

$c = \mathrm{Enc}_K(m_b)$

$m_2$

$c_2 = \mathrm{Enc}_K(m_2)$

$c_3$

$m_3 = \mathrm{Dec}_K(c_3)$

$c_4 = c$

⋮

**"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 $\mathrm{Enc}_k$ and $\mathrm{Dec}_k$
3. Adversary outputs $m_0$,$m_1$
4. Challenger sends the adversary $c=\mathrm{Enc}_k(m_b)$.
5. Adversary maintains oracle access to $\mathrm{Enc}_k$ and $\mathrm{Dec}_k$ ,however the adversary is not allowed to query $\mathrm{Dec}_k(c)$.
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.

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

## CCA-Security

**Definition 3.33:** An encryption scheme $\Pi$ 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

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

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

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

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

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

Attacker Receives: $\mathrm{m} = \begin{cases} 10^{n-1} & \text{if } b = 0 \\ 01^{n-1} & \text{if } b = 1 \end{cases}$

**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 (e.g., 16 bytes)
- Popular fix PKCS #5 padding
    - **1 bytes** of padding (**0x01**)
    - **2 bytes** of padding (**0x0202**)
    - **3 bytes** of padding (**0x030303**)
    - **4 bytes** of padding (**0x04040404**)
    - Invalid: (0x020303)
- “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

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

- $\mathrm{C}' = \langle r, F_k(r) \oplus M \oplus \text{0x0000 …. .30000} \rangle$

Ask to decrypt C'

- If we added < 3 bytes of padding?
  - → C' can be decrypted.
- If we added $\geq$ 3 bytes of padding?
  - →We will get a decryption error (bad padding)!

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

$$\mathrm{C}'' = \langle r, s = F_k(r) \oplus (M \oplus \text{0x0000 …. .030303}) \oplus \text{0x0 …}\, \mathbf{gg}040404 \rangle$$

If C'' decrypts then we can infer the byte **t** s.t. $M = x \parallel \boldsymbol{t} \parallel (0x030303)$ since $\mathbf{t} \oplus \mathbf{gg} = 0\text{x}04$.

# CCA-Security

- CCA-Security is strictly stronger than CPA-Security
- **Note:** 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 ☹
- Achieving CCA-Security?
  - Useful to guarantee integrity of the ciphertext
  - **Idea:** If attacker cannot generate valid new ciphertext c' (distinct from ciphertext obtained via eavesdropping) then ability to query decryption oracle is useless!
  - CCA-Security requires *non-malleability.*
  - **Intuition:** if attacker tampers with ciphertext c then c' is either invalid or m' is unrelated to m
  - Let $c = \mathrm{Enc}_{\mathrm{K}}(\mathrm{m}_{\mathrm{b}})$. Suppose attacker could generate a new valid ciphertext $c' \neq c$ such that $m'$ is related to $\mathrm{m}_{\mathrm{b}}$ the but not message $m_{1-b}$
    - How can the attacker win the CCA-Security game?
    - Ask for decryption of c' and check if $m'$ is related to $m_1$ or $m_0$

## 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:**

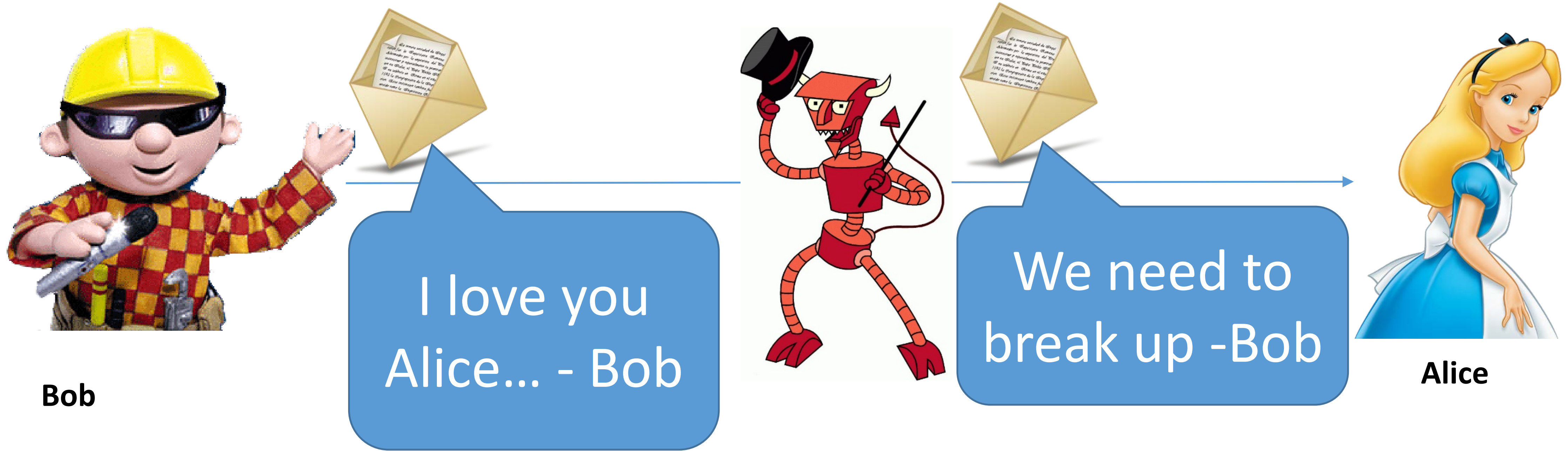
- Introduce Message Authentication Codes (MACs)
  - Key tool in Construction of CCA-Secure Encryption Schemes
- ~~Build Secure MACs~~

# 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

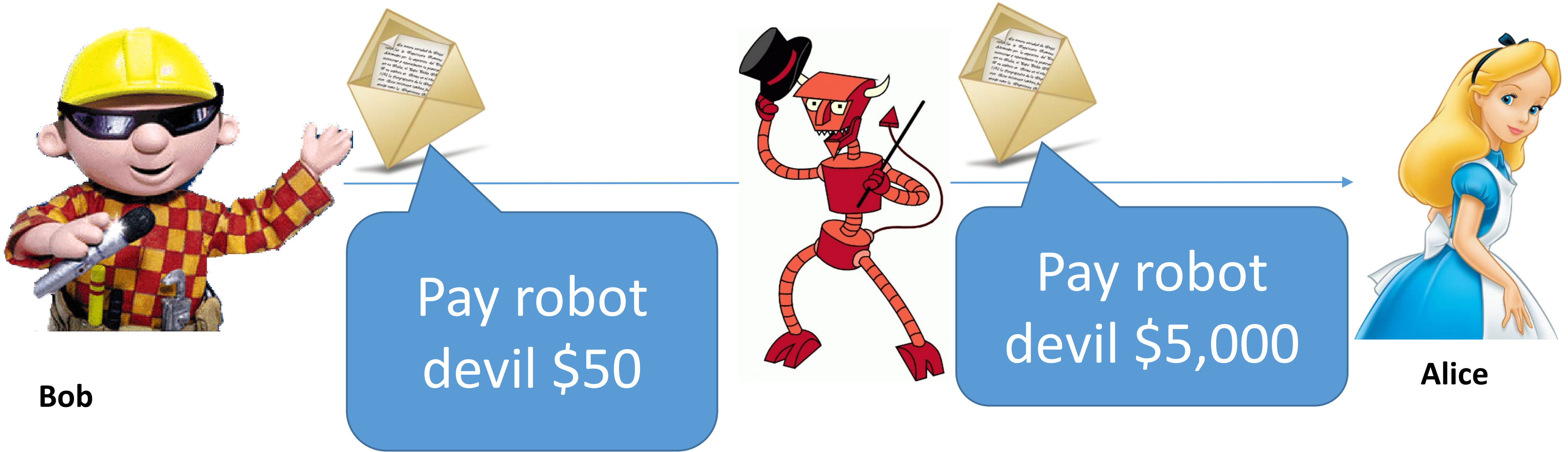
- CPA-Secure Encryption: Focus on Secrecy
  - But does not promise integrity

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

- CCA-Secure Encryption: Requires Secrecy and Non-Malleability (if attacker tampers with ciphertext it is either invalid or unrelated to original one)

## 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

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

$$versus$$

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

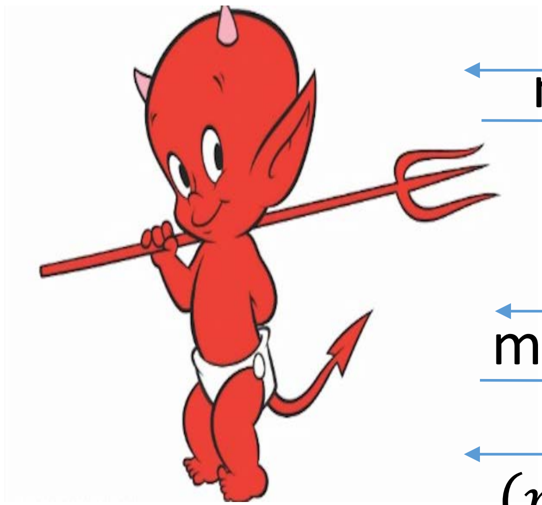
## Deterministic MACs

- **Canonical Verification Algorithm**

$$\mathrm{Vrfy}_{\mathrm{k}}(m,t) = \begin{cases} 1 & \text{if } t = \mathrm{Mac}_{\mathrm{k}}(m) \\ 0 & \text{otherwise} \end{cases}$$

- “All real-world MACs use canonical verification” – page 115

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



$m_1$

$t_1 = \text{Mac}_K(m_1)$

$m_2$

$t_2 = \text{Mac}_K(m_2)$

⋮

$m_q$

$t_q = \text{Mac}_K(m_q)$

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

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

**K = Gen(.)**

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

## 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

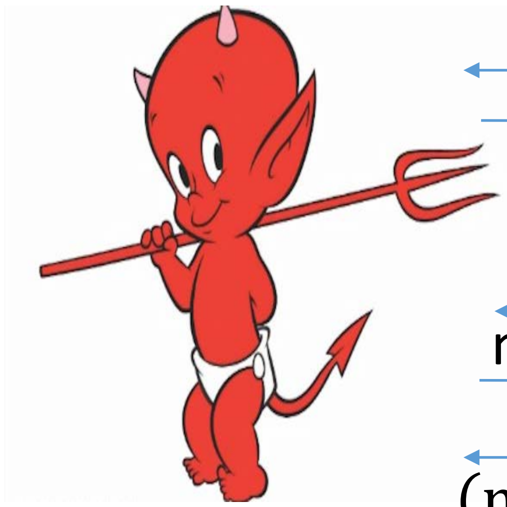
## Replay Attacks

- 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 ($\text{Mac}\textcolor{red}{\text{s}}\text{forge}_{A,\Pi}(n)$)



$m_1$ $\longrightarrow$

$\longleftarrow$ $t_1 = \text{Mac}_K(m_1)$

$m_2$ $\longrightarrow$

$\longleftarrow$ $t_2 = \text{Mac}_K(m_2)$

⋮

$m_q$ $\longrightarrow$

$\longleftarrow$ $t_q = \text{Mac}_K(m_q)$

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

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

**K = Gen(.)**

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

# 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 $\Pi$ 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