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

Reminder: Homework 3 is due on Thursday at 11:59PM on Gradescope

Spring 2021

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
- PRF Security: $\forall PPT\ D\ (\text{distinguisher})\ \exists \mu(.)(\text{negligible}))$

$$\text{ADV}_{\text{D}} := \left| Pr\left[D^{Fk(.)}(1^n)\right] - Pr\left[D^{f(.)}(1^n)\right] \right| \leq \mu(n)$$

where $f(.)$ is a truly random function and PRF key k is picked randomly

Recap CPA-Security

- Defend against eavesdropping attacker's ability to influence messages that honest party encrypts
  - More powerful than *known plaintext attacks* (knows vs controls 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 influences 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)

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

PRF Security

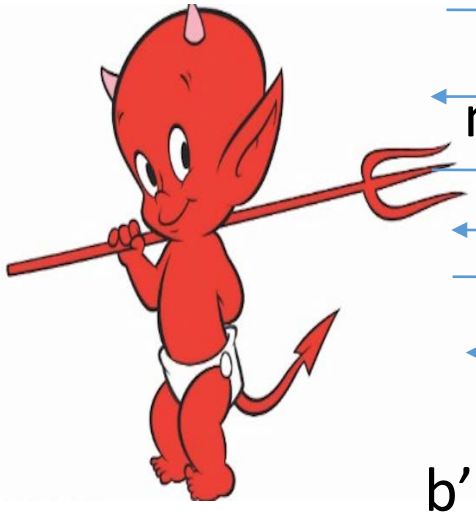
Definition 3.25: A keyed function $\mathrm{F}: \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ is a pseudorandom function if for all PPT distinguishers D there is a negligible function μ s.t.

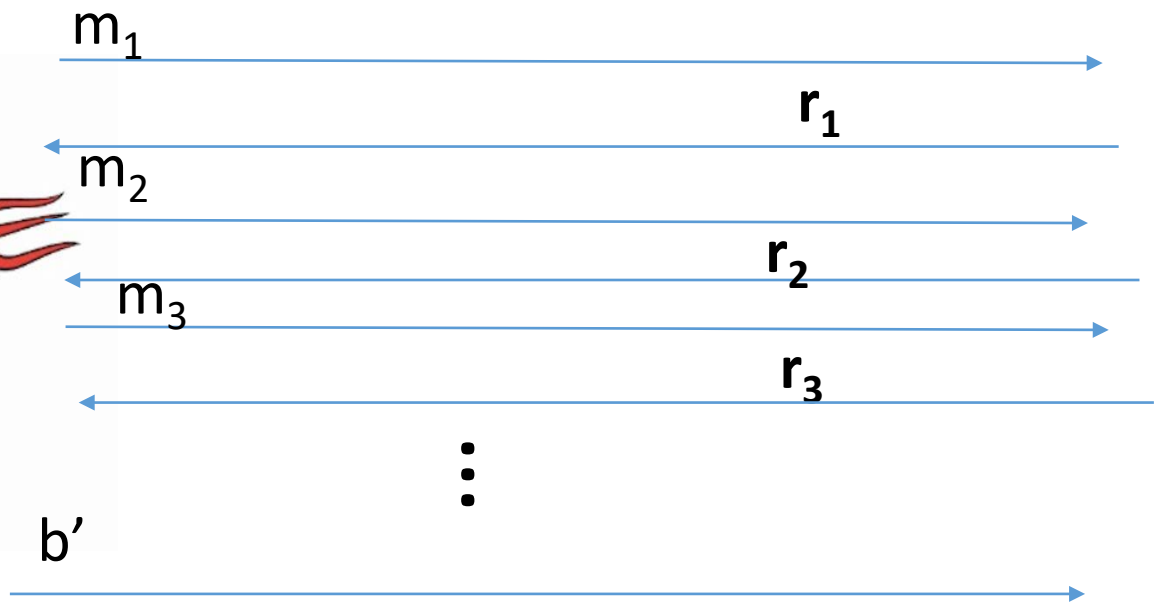
$$\left|Pr\left[D^{F_k(\cdot)}(1^n)\right] - Pr\left[D^{f(\cdot)}(1^n)\right]\right| \leq \mu(n)$$

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**$_n$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





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

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

Random bit b

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

Truly random func R

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

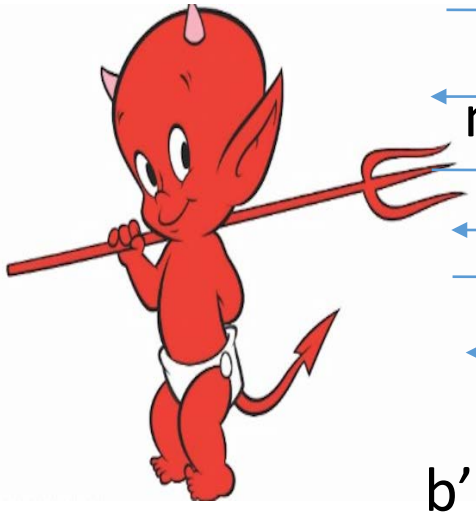
$R(m_i)$ o.w

PRF Security Concrete Version

Definition 3.25: A keyed function $\mathrm{F}: \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ 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

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

Reminder: CPA-Security (Single Message)



m_0, m_1

$c = Enc_K(m_b)$

m_2

$c_2 = Enc_K(m_2)$

m_3

$c_3 = Enc_K(m_3)$

$\vdots$

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

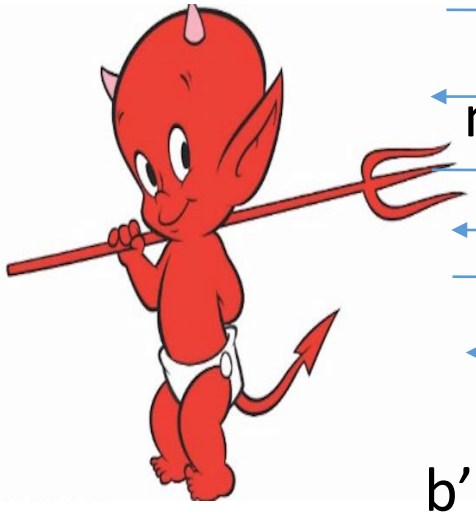
- **Dec:** Input $k \in \{0,1\}^n$ and $c = \langle r, s \rangle$

  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

$\langle r_3, F_k(r_3) \oplus m_3 \rangle$

$\vdots$

b’

Random bit b

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

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 and query O(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

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

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 and query O(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: By PRF security, for some negligible function μ, we have

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

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

Security Reduction

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

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

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

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

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

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

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

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 $r^* \neq r_1$,…,$r_q$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

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

Proof: If $r^* \neq r_1,\ldots,r_q$ then then c leaks no information about b (information theoretically). We have

$$\Pr\left[PrivK^{cpa}_{A,\widetilde{\Pi}} = 1\right]$$
$$\leq \Pr\left[PrivK^{cpa}_{A,\widetilde{\Pi}} = 1 \middle| r^* \neq r_1,\ldots,r_q\right] + \Pr\left[r* \in \{r_1,\ldots,r_q\}\right]$$
$$\leq \frac{1}{2} + \frac{q(n)}{2^n}$$

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

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

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.

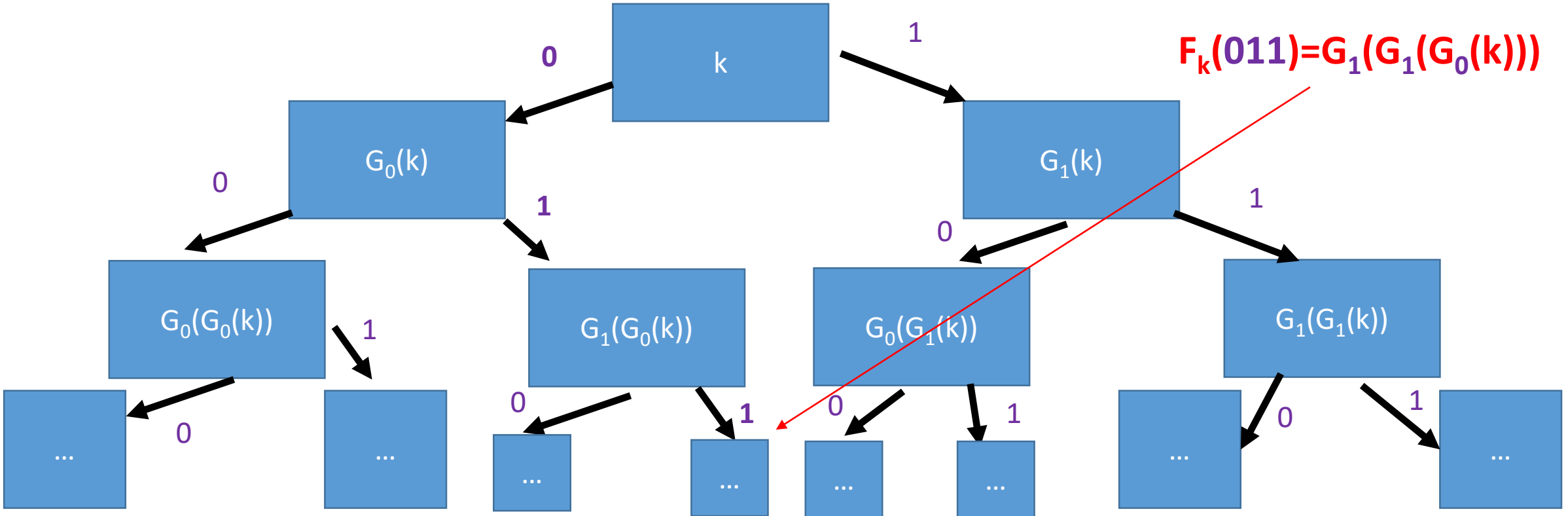
Let $G(x) = G_0(x)||G_1(x)$ (first/last n bits of output)

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

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.



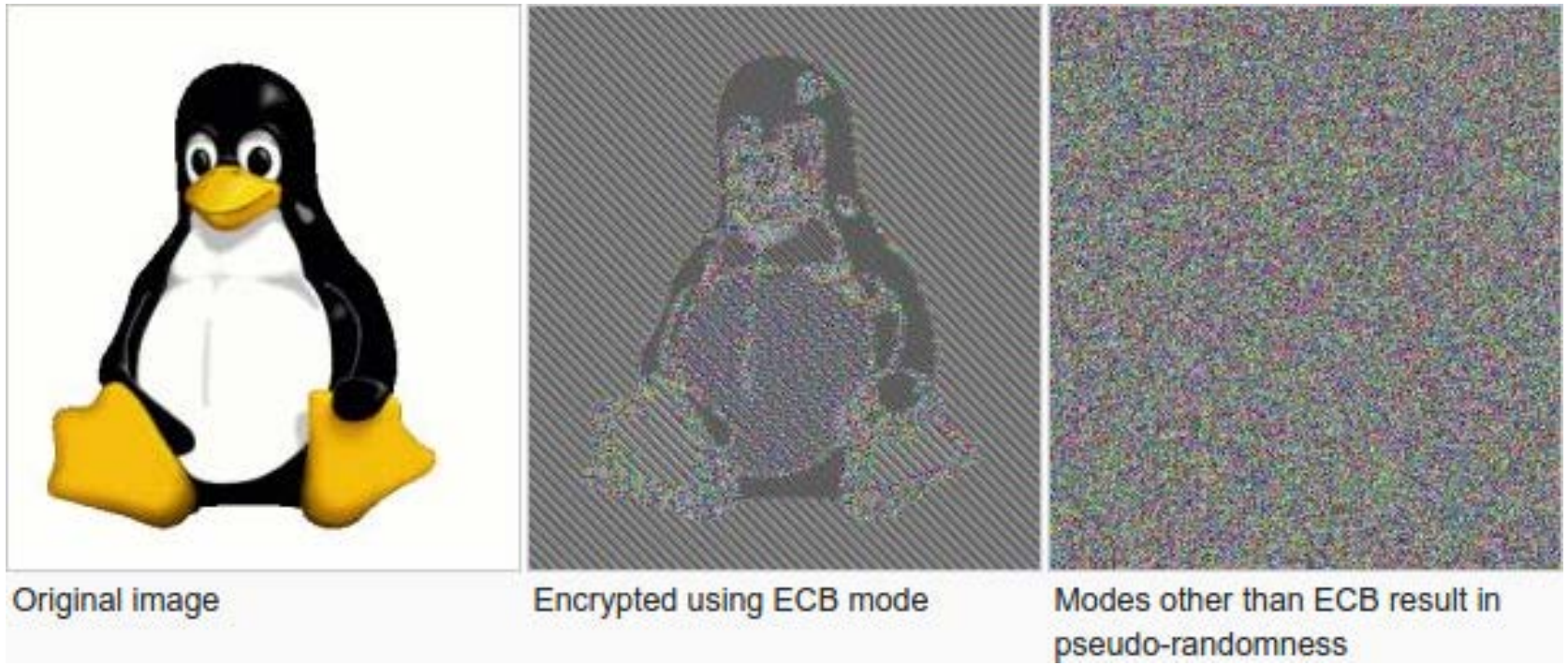
Stream Ciphers Modes

- 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: $c_i = \langle IV, m_i \oplus G_\infty(k, IV, 1^{|m_i|}) \rangle$
  - CPA-Secure if $F_k(IV) = G_\infty(k, IV, 1^n)$ is a (weak) PRF.
  - No shared state, but longer ciphertexts….

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 $\mathrm{F_k}(\mathrm{x})$ *and* $\mathrm{F_k^{-1}}(\mathrm{x})$ 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 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 $F: \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ is a **strong pseudorandom permutation** if for all PPT distinguishers D there is a negligible function μ s.t.

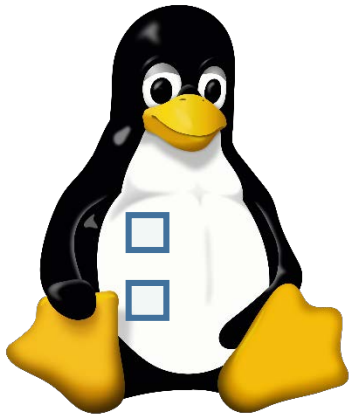
$$\left| Pr\left[D^{F_k(.),F_k^{-1}(.)}(1^n) \right] - Pr\left[D^{f(.),f^{-1}(.)}(1^n) \right] \right| \leq \mu(n)$$

Notes:

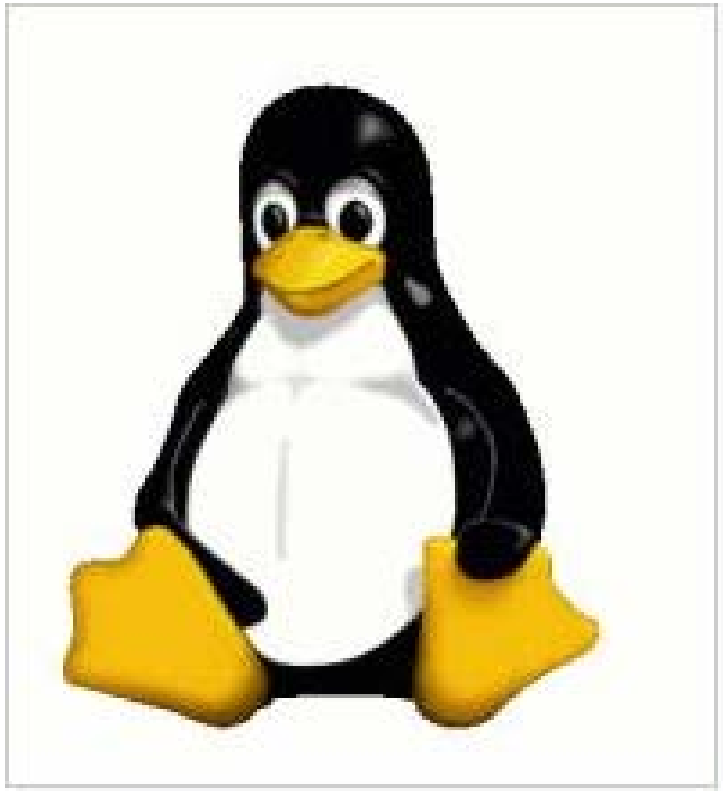
- the first probability is taken over the uniform choice of $k \in \{0,1\}^n$ as well as the randomness of D.
- the second probability is taken over uniform choice of f ∈**Perm$_n$**as well as the randomness of D.
- D is *never* given the secret k
- However, D is given oracle access to (keyed) permutation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

Electronic Code Book (ECB) Mode

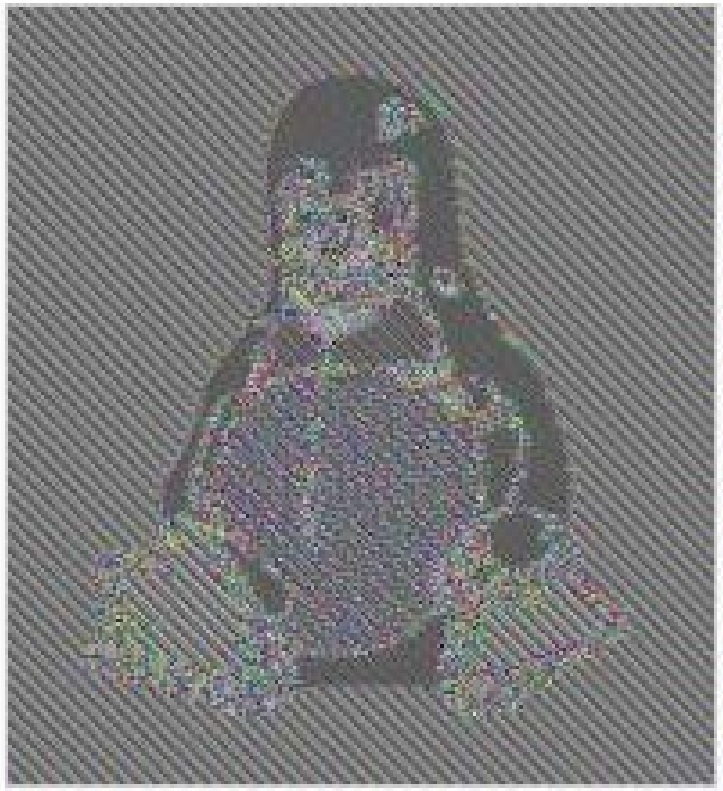
- Uses strong PRP $F_k(x)$ and $F_k^{-1}(x)$
- Enc_k
  - **Input**: $m_1,\ldots,m_\ell$
  - **Output**: $\langle F_k(m_1), \ldots, F_k(m_\ell)\rangle$
- How to decrypt?
- Is this secure?
- **Hint**: Encryption is deterministic.
  - **Implication**: Not CPA-Secure
  - But, it gets even worse



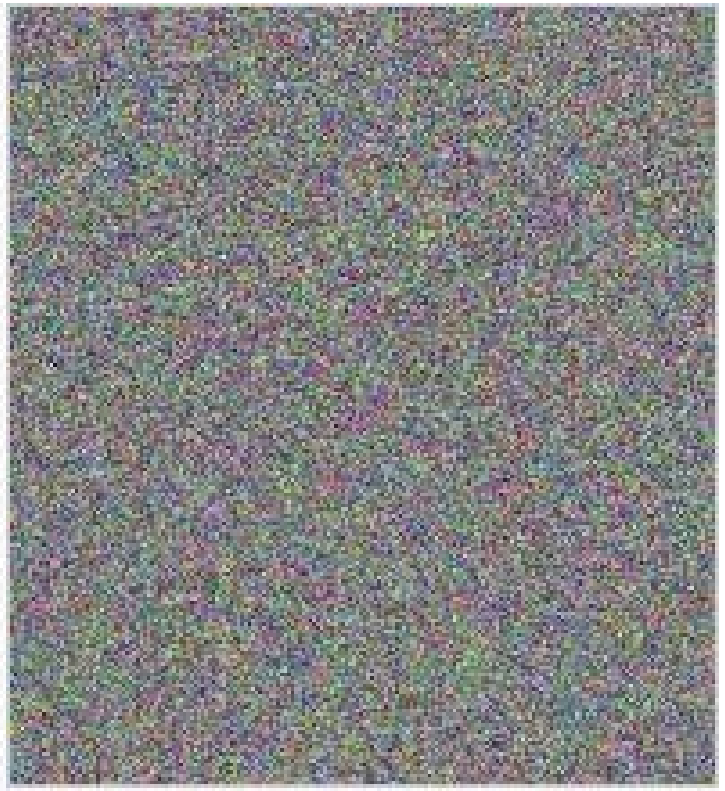
ECB Mode (A Failed Approach)



Original image



Encrypted using ECB mode



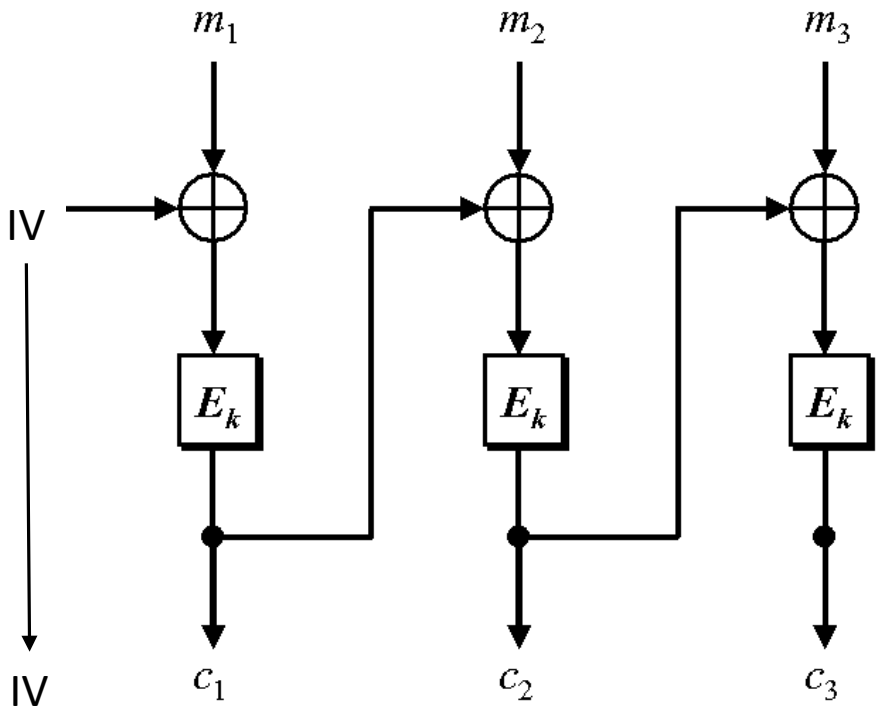
Modes other than ECB result in pseudo-randomness

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

How to decrypt? (**Hint:** Can be done in parallel)

Recap

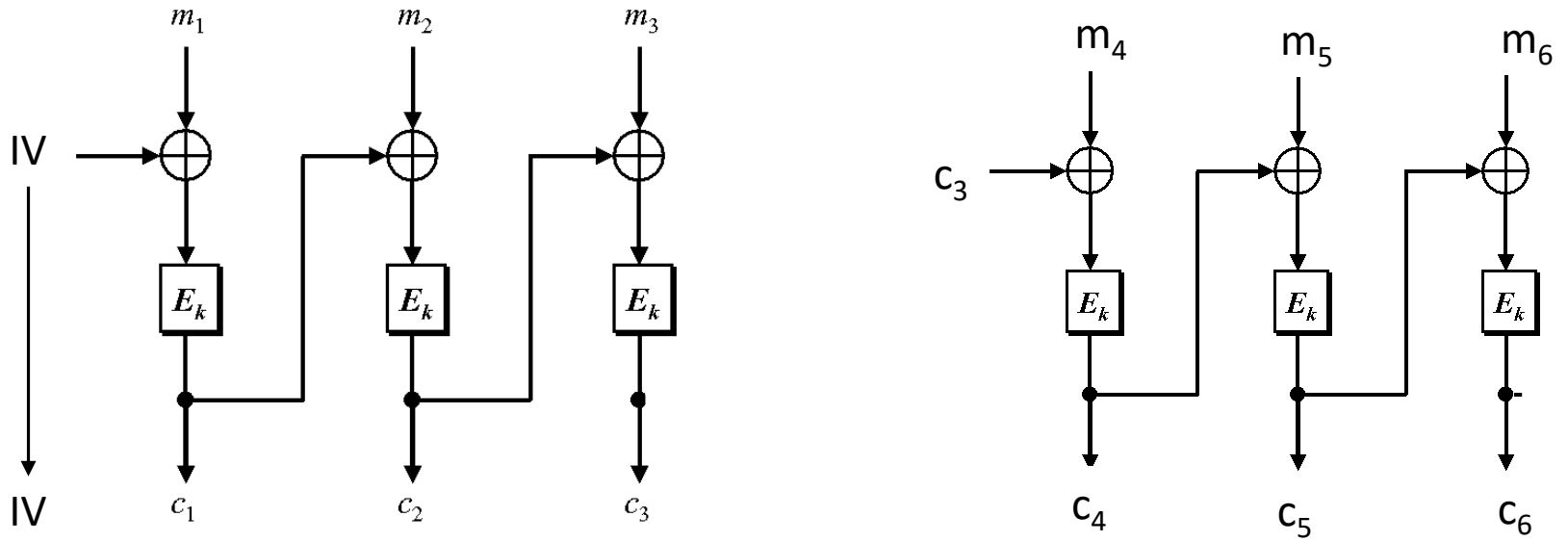
- PRFs/PRPs
- CPA-Secure Encryption + Security Reduction

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

- ECB Mode + Penguin Principle
- CBC Mode (CPA Secure)
- Chained CBC-Mode

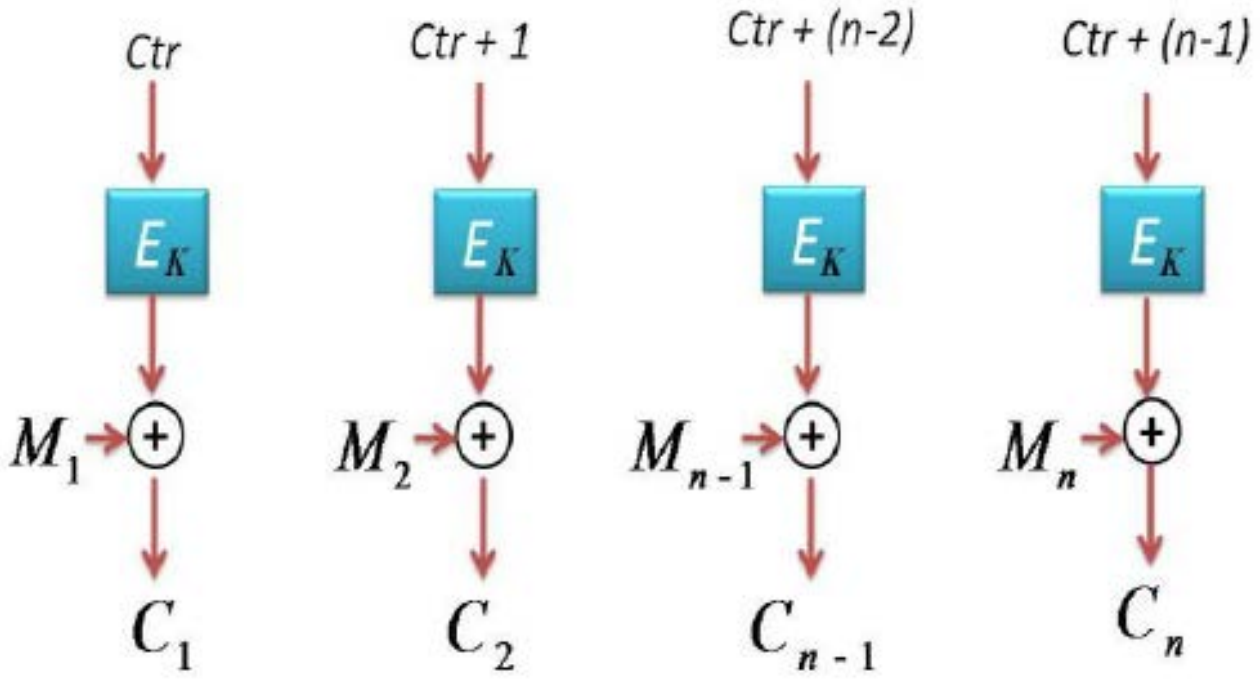
Reminder: HW Due Tonight 11:59PM on Gradescope

Chained CBC-Mode



- First glance: seems similar to CBC-Mode and reduces bandwidth
- Vulnerable to CPA-Attack! (Set $m_4 = IV \oplus c_3 \oplus m_1'$ and $c_4=c_1$ iff $m_1=m_1'$)
- **Moral**: Be careful when tweaking encryption scheme!

Counter Mode

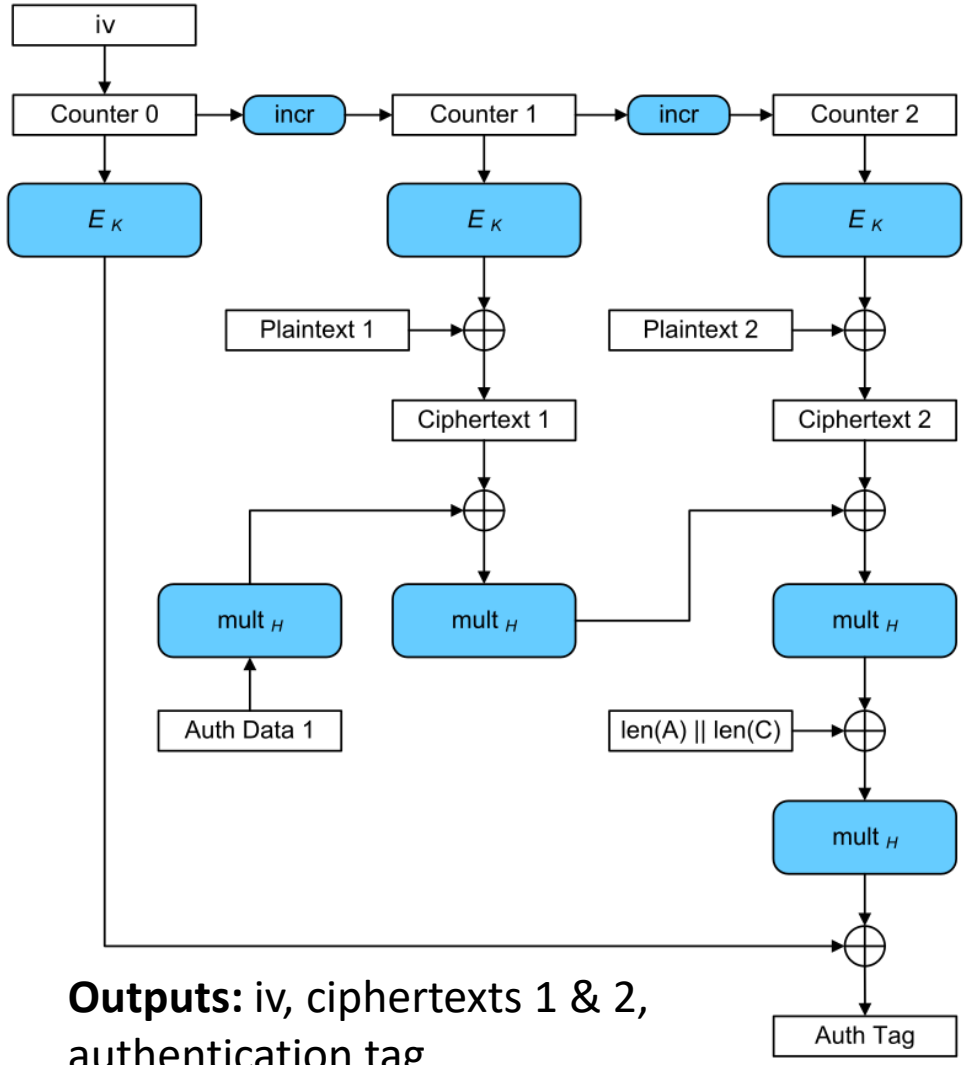


- **Input:** $m_1,\ldots,m_n$
- **Output:** $c = (ctr, c_1,c_2,\ldots,c_n)$ where ctr is chosen uniformly at random
- **Theorem**: If E_k is PRF (or PRP) then counter mode is CPA-Secure
- **Advantages**: Parallelizable encryption/decryption

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!

Input: plaintexts 1 & 2



Outputs: iv, ciphertexts 1 & 2, authentication tag

Week 3: Topic 3: CCA-Security

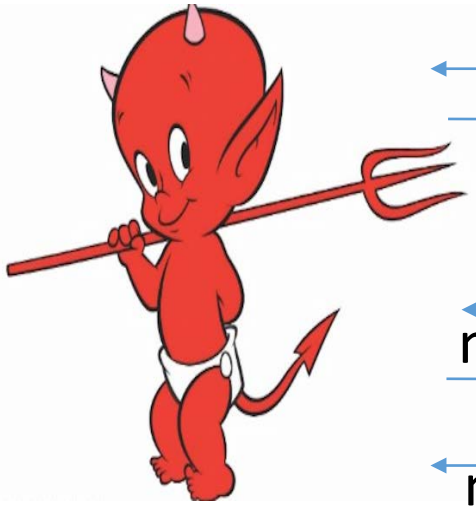
Chosen Ciphertext Attacks

- 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 $Enc_k(r)$ to recipient who authenticates by responding with r.

CCA-Security (Ind-CCA2)



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_2

$c_2 = Enc_K(m_2)$

c_3

$m_3 = Dec_K(c_3)$

$c_4 = c$

"No Way!"

b'

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

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.

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

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

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

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

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

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

- $C' = \langle r, F_k(r) \oplus M \oplus 0x0000 \ldots .30303 \oplus 0x00000 \ldots 0202 \rangle$

Ask to decrypt C'

- If we added 3 bytes of padding?
  - → C' can be decrypted (Looks like the message M' = M + 0x0301 with 1 byte padding).
- If we added $>$ 3 (or < 3) bytes of padding?
  - →We will get a decryption error (bad padding)!

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

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

If C'' decrypts then we can infer the byte **t** s.t. $M = x \parallel \boldsymbol{t} \parallel (0x030303)$ since $\mathbf{t} \oplus \mathbf{gg} = 0x04$.

Example

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

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

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

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

If C'' decrypts then we can infer the byte **t** s.t. $M = x \parallel \boldsymbol{t} \parallel (0x030303)$ since $\mathbf{t} \oplus \mathbf{gg} = 0x04$.

Question: How do we infer the next byte **t'**?

Answer: Set $C''' = C'' \oplus 0x0 \ldots \mathbf{gg}05050505$

if decryption is successful then $\mathbf{t}' \oplus \mathbf{gg} = 0x05$

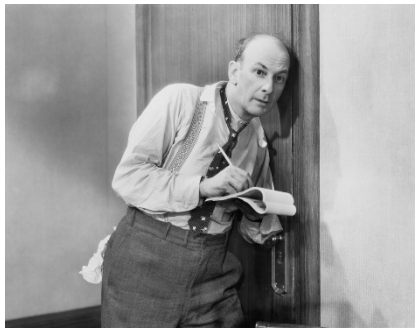
CCA-Security

- 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 $c = \mathrm{Enc}_K(m_b)$. Suppose attacker could generate a new valid ciphertext $c' \neq c$ such that m' is related to m_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

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

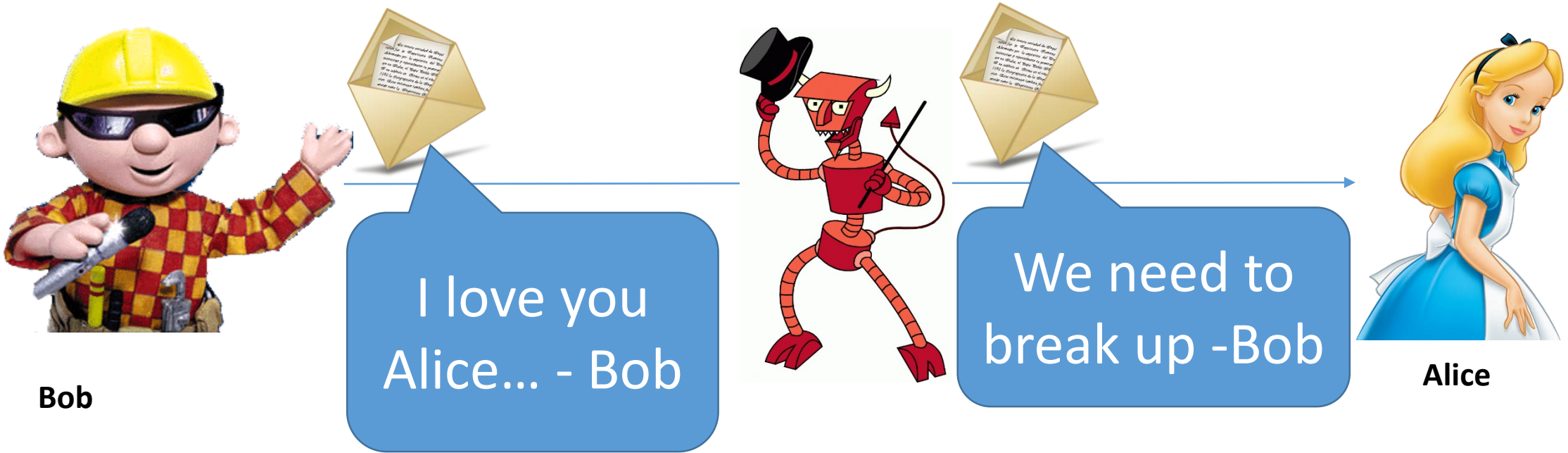
What Does It Mean to “Secure Information”

- 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

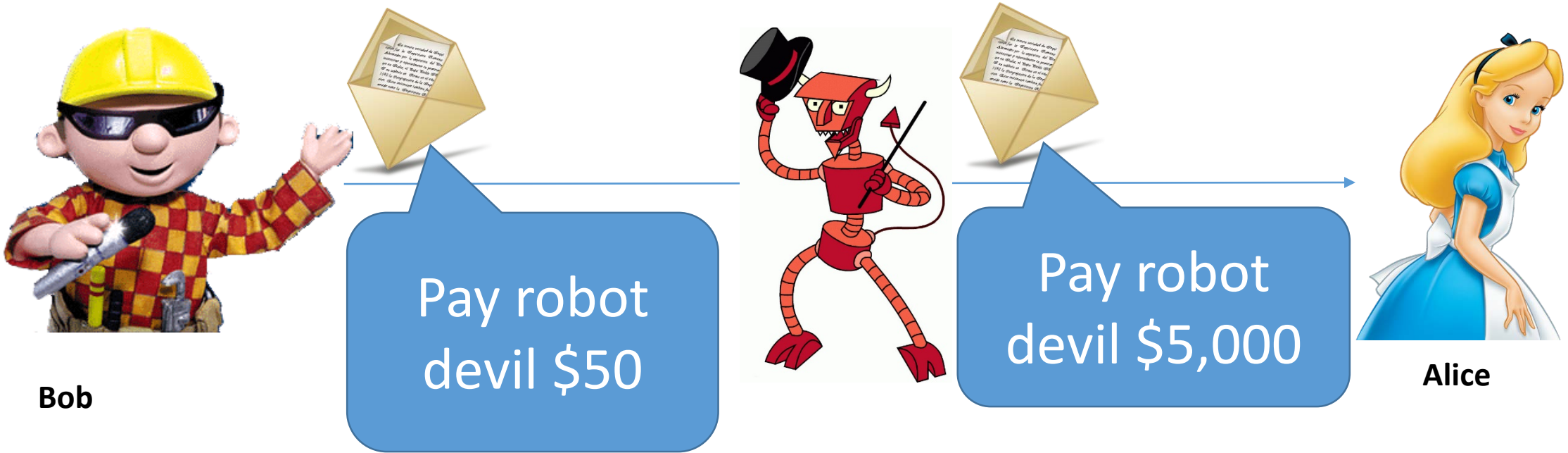


Message Authentication Codes

- 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

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

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

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

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 $\mathrm{k} \in \mathcal{K}$
- $\mathrm{Mac}_{\mathrm{k}}(m; R)$ (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

- $\mathrm{Gen}(1^n; R)$ (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

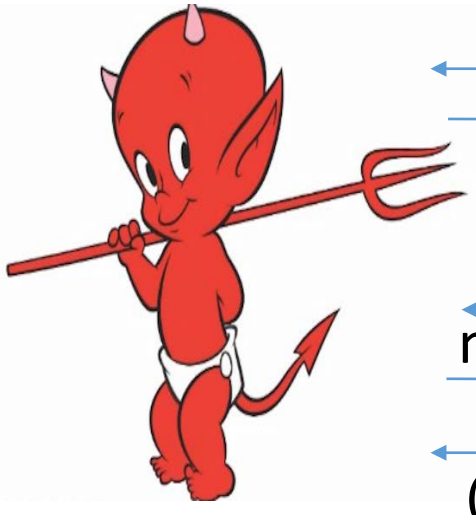
Definition 4.1: A message authentication code (MAC) consists of three algorithms $\Pi = (\mathrm{Gen}, \mathrm{Mac}, \mathrm{Vrfy})$

- $\mathrm{Gen}(1^n; R)$ (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")

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

Security Goal (Informal): Attacker should not be able to forge a valid tag t' for new message m' that s/he wants to send.

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



m_1

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

m_2

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

$\vdots$

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(.)

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

$\Pr[\text{Macforge}_{A,\Pi}(n) = 1] \leq \mu(n)$

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

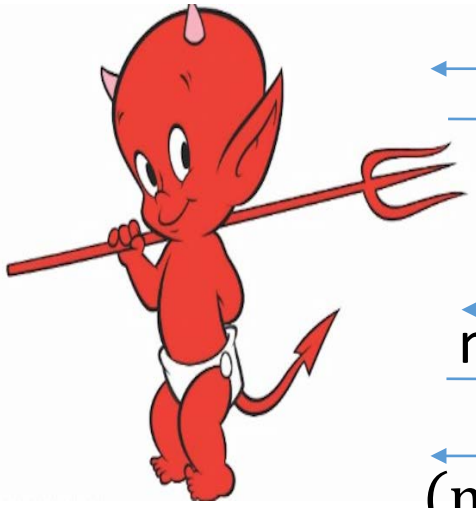
- MACs alone do not protect against replay attacks (they are stateless)

- 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 ($\mathrm{Macsforge}_{A,\Pi}(n)$)



m_1

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

m_2

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

$\vdots$

m_q

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

(m, t) s.t $(m, t) \notin \{(m_1, t_1), \ldots, (m_q, t_q)\}$

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

K = Gen(.)

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

$\Pr[\mathrm{Macsforge}_{A,\Pi}(n) = 1] \leq \mu(n)$

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

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 1 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 1 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

Recap

- Data Integrity
- Message Authentication Codes
- Side-Channel Attacks
- ~~Build Secure MACs~~
- ~~Construct CCA-Secure Encryption Scheme~~

Current Goal:

- Build a Secure MAC
  - Key tool in Construction of CCA-Secure Encryption Schemes

General vs Fixed Length MAC

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

versus

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

Strong MAC Construction (Fixed Length)

Simply uses a secure PRF F

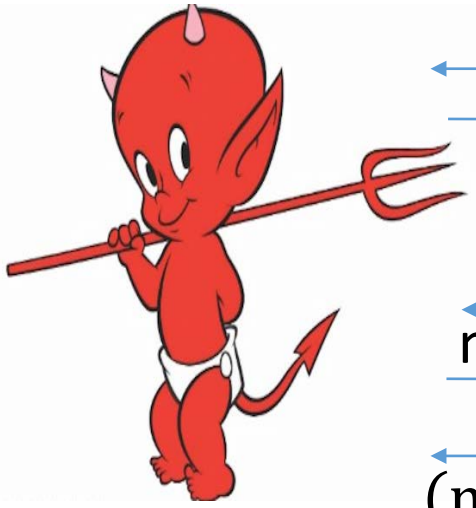
$$\mathrm{Mac}_k(m) = \mathrm{F}_K(m)$$

Question: How to verify the a MAC?

Canonical Verification Algorithm…

$$\mathrm{Vrfy}_k(m, t) = \begin{cases} 1 & \text{if } t = \mathrm{F}_K(m) \\ 0 & \text{otherwise} \end{cases}$$

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



m_1

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

m_2

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

$\vdots$

m_q

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

(m, t) s.t $(m, t) \notin \{(m_1, t_1), \ldots, (m_q, t_q)\}$

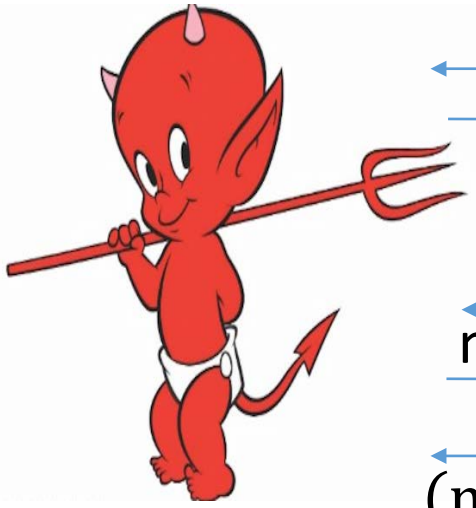
$\mathrm{Macsforge}_{A,\Pi}(n) = \mathrm{Vrfy}_k(m, t)$

K = Gen(.)

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

$\Pr[\mathrm{Macsforge}_{A,\Pi}(n) = 1] \leq \mu(n)$

Concrete Version: $(t(n), q(n), \varepsilon(n))$-secure MAC



m_1

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

m_2

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

$\vdots$

m_q

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

(m, t) s.t $(m, t) \notin \{(m_1, t_1), \ldots, (m_q, t_q)\}$

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

K = Gen(.)

$$\forall\, A \text{ with } \left(\text{time}(A) \le t(n), \text{queries}(A) \le q(n)\right)$$

$$\Pr\left[\text{Macsforge}_{A,\Pi}(n) = 1\right] \le \varepsilon(n)$$

Strong MAC Construction (Fixed Length)

$$\mathrm{Mac}_k(m) = \mathrm{F}_K(m)$$

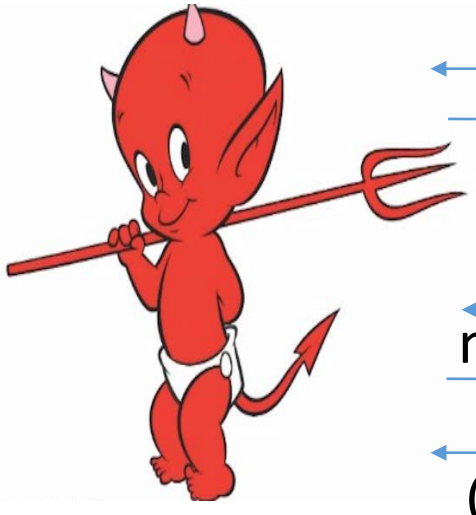
$$\mathrm{Vrfy}_k(m, t) = \begin{cases} 1 & \text{if } t = \mathrm{F}_K(m) \\ 0 & \text{otherwise} \end{cases}$$

Theorem 4.6: If F is a PRF then this is a secure (fixed-length) MAC for messages of length n.

Proof: Start with attacker who breaks MAC security and build an attacker who breaks PRF security (contradiction!)

Sufficient to start with attacker who breaks regular MAC security (why?)

Breaking MAC Security $(\text{Macforge}_{A,\Pi}(n))$



m_1

$\boldsymbol{t_1 = F_K(m_1)}$

m_2

$\boldsymbol{t_2 = F_K(m_2)}$

$\vdots$

m_q

$\boldsymbol{t_q = F_K(m_q)}$

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

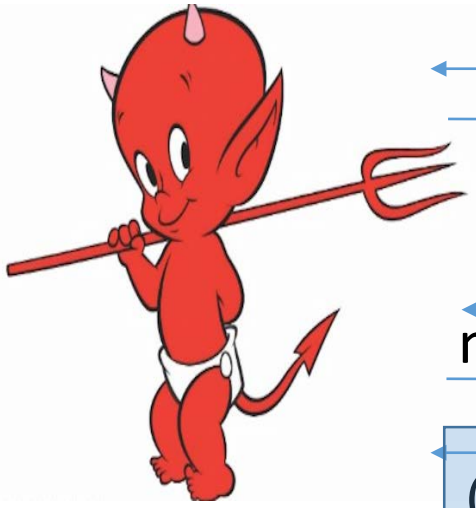
$\text{Macforge}_{A,\Pi}(n) = \text{Vrfy}_k(m, t)$

K = Gen(.)

$\exists PPT\ A\ and\ g(.)$ (positive/non negligible) s. t

$$\Pr[\text{Macforge}_{A,\Pi}(n) = 1] > g(n)$$

A Similar Game (Macforge$_{A,\tilde{\Pi}}$



m_1

$t_1 = f(m_1)$

m_2

$t_2 = f(m_2)$

$\vdots$

m_q

$t_q = f(m_q)$

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

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

Why? Because f(m) is distributed uniformly in $\{0,1\}^n$ so $\Pr[f(m)=t]=2^{-n}$

Truly Random Function
f $\in$ Func$_n$

Claim: $\forall A$ $(not\ just\ PPT)$

$$\Pr\left[\text{Macforge}_{A,\tilde{\Pi}}(n) = 1\right] \leq 2^{-n}$$

PRF Distinguisher D

- Given oracle O (either F_K or truly random f)
- Run PPT Macforge adversary A
- When adversary queries with message m, respond with O(m)
- Output 1 if attacker wins (otherwise 0)

- If O = f then

$$\Pr[D^{O}(1^n) = 1] = \Pr\left[\text{Macforge}_{A,\tilde{\Pi}}(n) = 1\right] \leq 2^{-n}$$

- If O=F_K then

$$\Pr[D^{O}(1^n) = 1] = \Pr\left[\text{Macforge}_{A,\Pi}(n) = 1\right] > g(n)$$

PRF Distinguisher D

- If O = f then

$$\Pr[D^{O}(1^n) = 1] = \Pr\left[\text{Macforge}_{A,\tilde{\Pi}}(n) = 1\right] \leq 2^{-n}$$

- If O=F_K then

$$\Pr[D^{O}(1^n) = 1] = \Pr\left[\text{Macforge}_{A,\Pi}(n) = 1\right] > g(n)$$

Advantage:

$$|\Pr[D^{F_K}(1^n) = 1] - \Pr[D^{f}(1^n) = 1]| > g(n) - 2^{-n}$$

Note that $g(n) - 2^{-n}$ is non-negligible and D runs in PPT if A does.

Strong MAC Construction (Fixed Length)

$$\mathrm{Mac}_k(m) = \mathrm{F}_K(m)$$

$$\mathrm{Vrfy}_k(m, t) = \begin{cases} 1 & \text{if } t = \mathrm{F}_K(m) \\ 0 & \text{otherwise} \end{cases}$$

Theorem 4.6: If F is a PRF then this is a secure (fixed-length) MAC for messages of length n.

Strong MAC Construction (Fixed Length)

$$\mathrm{Mac}_k(m) = \mathrm{F}_K(m)$$

$$\mathrm{Vrfy}_k(m,t) = \begin{cases} 1 & \text{if } t = \mathrm{F}_K(m) \\ 0 & \text{otherwise} \end{cases}$$

Theorem (Concrete): If F is a $(t(n), q(n), \varepsilon(n))$-secure PRF then the above construction is a $(t(n) - O(n), q(n), \varepsilon(n) + 2^{-n})$-secure MAC for $\mathcal{M} = \{0,1\}^n$ (messages of length n).

Example: F is a $\left(2^n, 2^{n/2}, 2^{-n}\right)$-secure PRF-→ the above MAC construction is $\left(2^n - O(n), 2^{n/2}, 2^{-n+1}\right)$-secure

Strong MAC Construction (Fixed Length)

$$\mathrm{Mac}_k(m) = \mathrm{F}_K(m)$$

$$\mathrm{Vrfy}_k(m,t) = \begin{cases} 1 & \text{if } t = \mathrm{F}_K(m) \\ 0 & \text{otherwise} \end{cases}$$

Theorem (Concrete): If F is a $(t(n), q(n), \varepsilon(n))$-secure PRF then the above construction is a $(t(n) - O(n), q(n), \varepsilon(n) + 2^{-n})$-secure MAC for $\mathcal{M} = \{0,1\}^n$ (messages of length n).

Limitation: What if we want to authenticate a longer message? $\mathcal{M} = \{0,1\}^*$

MACs for Arbitrary Length Messages

- Building Block Π'=(Mac',Vrfy'), a secure MAC for length n messages

First: A few failed attempts

Let m = m_1,…,m_d where each m_i is n bits and let $t_i = \text{Mac}'_K(m_i)$

$$\text{Mac}_K(m) = \langle t_1, \ldots, t_d \rangle$$

m_1 = *"I love you"*

m_2 = "I will never say that"

m_3 = "you are stupid"

What is wrong?

Block-reordering attack

$$\text{Mac}_K(m_d,\ldots,m_1) = \langle t_d, \ldots, t_1 \rangle$$

MACs for Arbitrary Length Messages

- Building Block Π'=(Mac',Vrfy'), a secure MAC for length n messages

Attempt 2

Let m = m_1,…,m_d where each m_i is n bits and let $t_i = \text{Mac}'_K(i \parallel m_i)$

$$\text{Mac}_K(m) = \langle t_1, \ldots, t_d \rangle$$

Addresses block-reordering attack.

Any other concerns?

Truncation attack!

$$\text{Mac}_K(m_1,\ldots,m_{d-1}) = \langle t_1, \ldots, t_{d-1} \rangle$$

Suppose $m_1, \ldots, m_{d-1}, m_d =$ "I don't like you. I LOVE you!"

MACs for Arbitrary Length Messages

- Building Block Π'=(Mac',Vrfy'), a secure MAC for length n messages

Attempt 3

Let m = $m_1,\ldots,m_d$ where each m_i is n bits and m has length $\ell = nd$

Let $t_i = \mathrm{Mac}'_K(i \parallel \ell \parallel m_i)$

$$\mathrm{Mac}_K(m) = \langle t_1, \ldots, t_d \rangle$$

Addresses truncation.

Any other concerns?

Mix and Match Attack!

MACs for Arbitrary Length Messages

Let m = m_1,…,m_d where each m_i is n bits and m has length $\ell = nd$

Let m′ = m'_1,…,m'_d where each m'_i is n bits and m has length $\ell = nd$

Let $t_i = \mathrm{Mac}'_K(i \parallel \ell \parallel m_i)$ and $t'_i = \mathrm{Mac}'_K(i \parallel \ell \parallel m_i\,')$

$$\mathrm{Mac}_K(m) = \langle t_1, \ldots, t_d \rangle$$

$$\mathrm{Mac}_K(m') = \langle t'_1, \ldots, t'_d \rangle$$

Mix and Match Attack!

$$\mathrm{Mac}_K(m_1, m'_2, m_3, \ldots) = \langle t_1, t'_2, t_3, \ldots \rangle$$

m_1 = *"What will I say to Eve?"*

m_2 = "You are evil and vile."

m_3 = "Please leave me alone!"

m_4 = "Your sworn enemy - BOB"

$t = \langle t_1, t_2, , t_3, t_4 \rangle$

m_1' = *"Dear Alice"*

m_2' = "You are wonderful."

m_3' = "I can't wait to see you!"

m_4' = "XOXOXOXOXO - BOB"

$t' = \langle t_1', t_2', , t_3', t_4' \rangle$

m_1' = *"Dear Alice"*

m_2 = "You are evil and vile."

m_3 = "Please leave me alone!"

m_4 = "Your sworn enemy - BOB"

$t'' = \langle t_1', t_2, , t_3, t_4 \rangle$

MACs for Arbitrary Length Messages

- A non-failed approach ☺
- Building Block Π'=(Mac',Vrfy'), a secure MAC for length n messages
- Let m = m_1,…,m_d where each m_i is n/4 bits and m has length $\ell < 2^{n/4}$

Mac_K(m)=

- Select random $\frac{n}{4}$ bit nonce r
- Let $t_i = Mac'_K(r \parallel \ell \parallel i \parallel m_i)$ for i=1,…,d
  - (**Note**: encode i and ℓ as $\frac{n}{4}$ bit strings)
- **Output** $\langle r, t_1, \ldots, t_d \rangle$

MACs for Arbitrary Length Messages

$\text{Mac}_K(m)$=

- Select random n/4 bit string r
- Let $t_i = \text{Mac}'_K(r \parallel \ell \parallel i \parallel m_i)$ for i=1,…,d
  - (Note: encode i and ℓ as n/4 bit strings)
- **Output** $\langle r, t_1, \ldots, t_d \rangle$

Theorem 4.8: If Π' is a secure MAC for messages of fixed length n, above construction $\Pi = (\text{Mac}, \text{Vrfy})$ is secure MAC for arbitrary length messages.

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~~