# Homework 1

- Due: Thursday at 3PM (beginning of class)

**Q4 Typo:** $\varepsilon_t = \frac{1.5t}{2^n}$ (not $\varepsilon_t = \frac{1.5t}{2^t}$)

Please Typeset Your Solutions (LaTeX, Word etc…)

- You may collaborate, but must write up your own solutions in your own words

# Tidbits

- The use of the names "Alice and Bob" in crypto originates from the seminal 1978 RSA paper of Ron Rivest, Adi Shamir and Leonard Adleman (see https://en.wikipedia.org/wiki/Alice_and_Bob).

- **Electronic Code Book Mode (ECB):** named after convention physical codebooks, which usually consists of a lookup table for encryption/decryption (see https://en.wikipedia.org/wiki/Codebook).

# Cryptography
# CS 555

**Week 4:**

- Message Authentication Codes
- CBC-MAC
- Authenticated Encryption + CCA Security

**Readings:** Katz and Lindell Chapter 4.1-4.4

# Recap

- Chosen Plaintext Attacks/Chosen Ciphertext Attacks
  - CPA vs CCA-security
- Keyed Pseudorandom Functions and Permutations
  - Achieving CPA-Security
- Blockciphers and Modes of Operation

# 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 $m_0$

# Week 4: Topic 1:
# Message Authentication Codes

# Current Goals

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

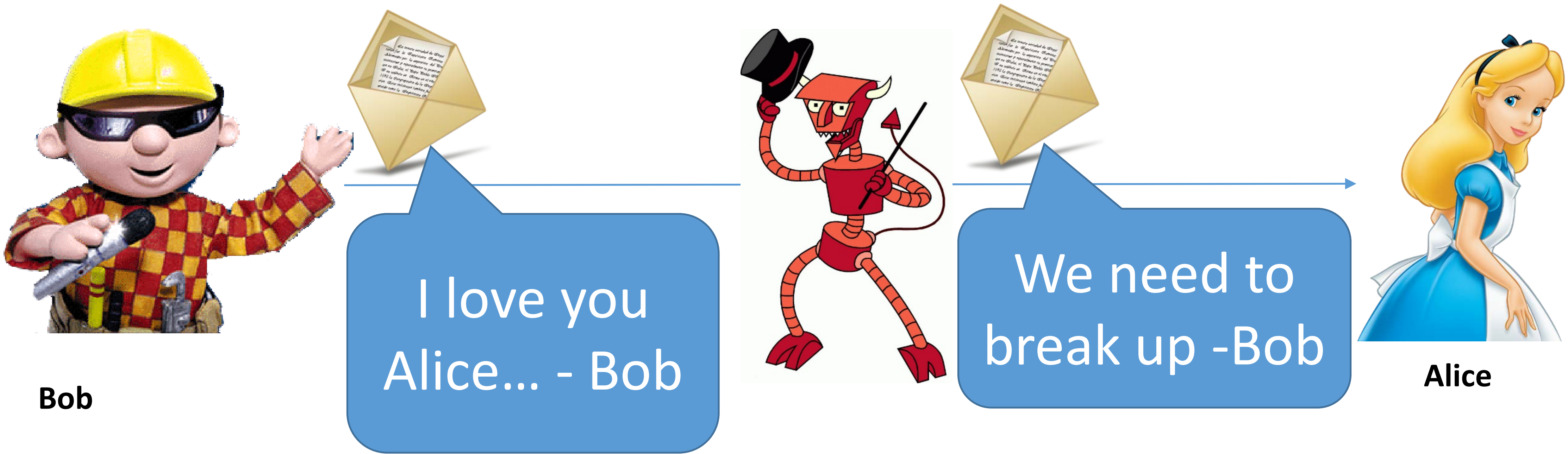
# What Does It Mean to “Secure Information”

- 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

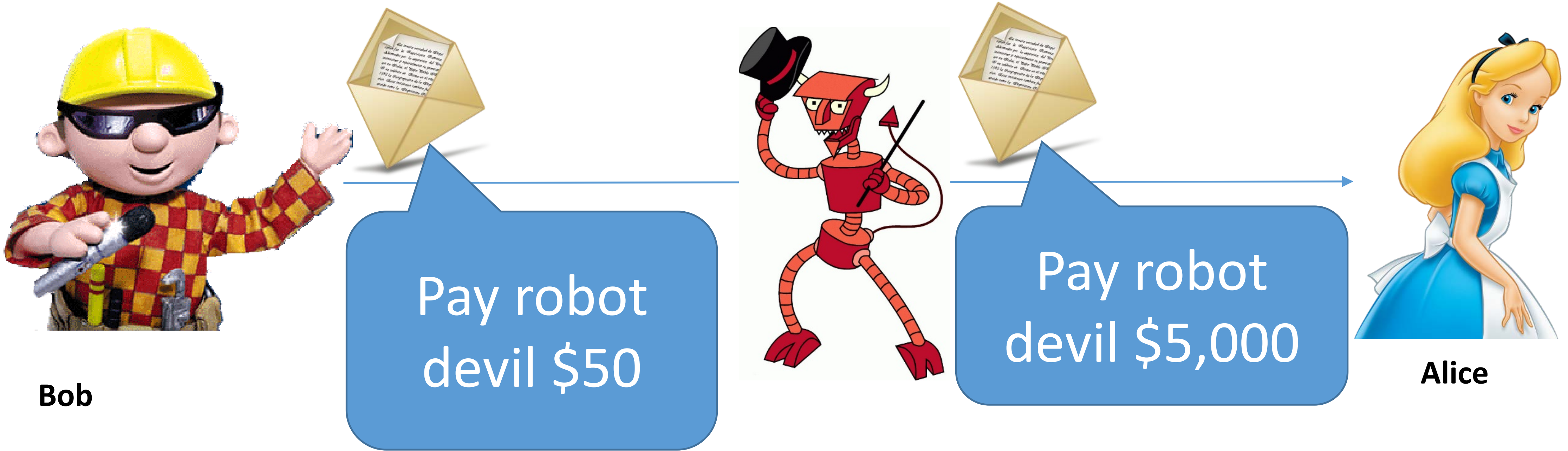
- CPA-Secure Encryption: Focus on Secrecy
  - But does not promise integrity

- 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

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

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

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

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

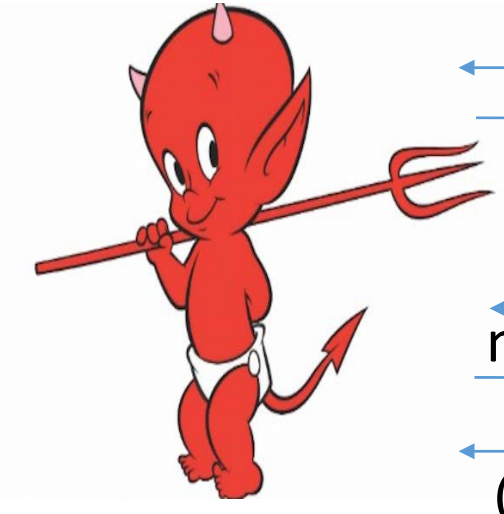
**Definition 4.1**: A message authentication code (MAC) consists of three algorithms $\Pi = (\mathrm{Gen}, \mathrm{Mac}, \mathrm{Vrfy})$

- $\mathrm{Gen}(1^n; R)$ (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 ($\mathrm{Macforge}_{A,\Pi}(n)$)



$m_1$

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

$m_2$

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

$\vdots$

$m_q$

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

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

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

**K = Gen(.)**

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

# 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

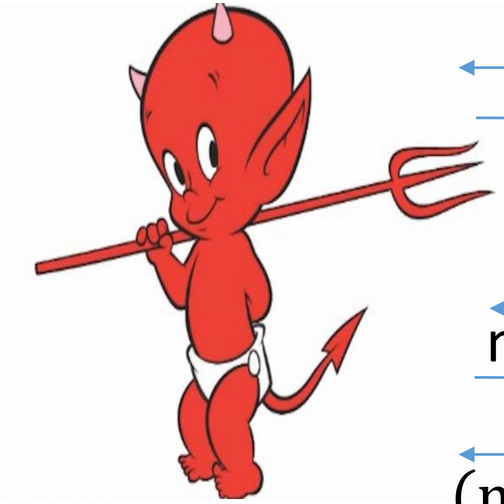
- MACs alone do not protect against replay attacks (they are stateless)

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



$\mathrm{m}_1$

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

$\mathrm{m}_2$

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

⋮

$\mathrm{m}_q$

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

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

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

**K = Gen(.)**

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

# 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

# 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 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

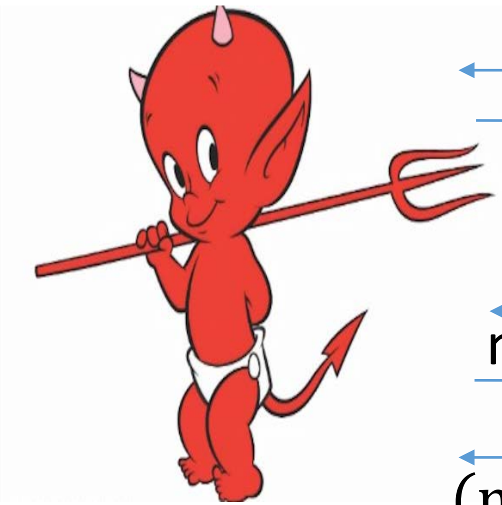
$$\mathrm{Mac}_{\mathrm{k}}(m) = \mathrm{F}_{\mathrm{K}}(m)$$

**Question:** How to verify the a MAC?

Canonical Verification Algorithm…

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

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



$m_1$

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

$m_2$

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

⋮

$m_q$

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

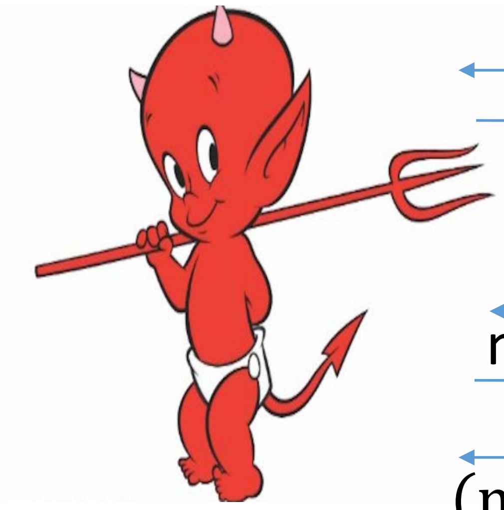
$(\mathrm{m}, \mathrm{t})$ s.t $(\mathrm{m}, \mathrm{t}) \notin \{(\mathrm{m}_1, \mathrm{t}_1), \ldots, (\mathrm{m}_q, \mathrm{t}_q)\}$

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

**K = Gen(.)**

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

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



$m_1$

**$t_1$ = Mac$_K$($m_1$)**

$m_2$

**$t_2$ = Mac$_K$ ($m_2$)**

⋮

$m_q$

**$t_q$ = Mac$_K$($m_q$)**

$(\mathrm{m}, \mathrm{t})$ 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(.)**

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

$$\Pr\big[\mathrm{Macsforge}_{A,\Pi}(n) = 1\big] \le \varepsilon(n)$$

# Strong MAC Construction (Fixed Length)

$$\mathrm{Mac}_{\mathrm{k}}(m) = \mathrm{F}_{\mathrm{K}}(m)$$

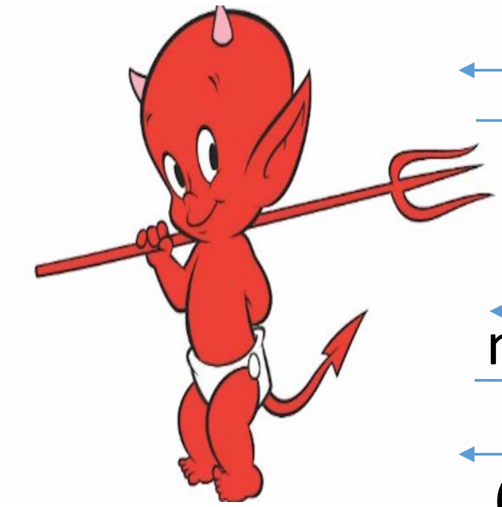
$$\mathrm{Vrfy}_{\mathrm{k}}(m, t) = \begin{cases} 1 & \text{if } t = \mathrm{F}_{\mathrm{K}}(m) \\ 0 & \quad\text{otherwise} \end{cases}$$

**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 $\left(\mathrm{Macforge}_{A,\Pi}(n)\right)$



$\mathrm{m}_1$

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

$\mathrm{m}_2$

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

⋮

$\mathrm{m}_q$

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

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

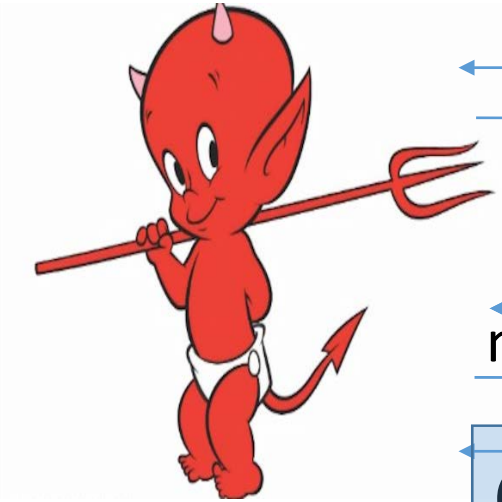
$\mathrm{Macforge}_{A,\Pi}(n) = \mathrm{Vrfy}_{\mathrm{k}}(m,t)$

**K = Gen(.)**

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

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

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



$m_1$

$t_1 = f(m_1)$

$m_2$

$t_2 = f(m$

$\vdots$

$m_q$

$t_q = f(m_q)$

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

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

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

**Truly Random Function**

**$f \in \mathrm{Func}_n$**



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

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

# 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

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

- 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)

$$\mathrm{Mac}_{\mathrm{k}}(m) = \mathrm{F}_{\mathrm{K}}(m)$$

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

**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)

$$\mathrm{Mac}_{\mathrm{k}}(m) = \mathrm{F}_{\mathrm{K}}(m)$$

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

**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 $(2^n, 2^{n/2}, 2^{-n})$-secure PRF-→ the above MAC construction is $(2^n - O(n), 2^{n/2}, 2^{-n+1})$-secure

# Homework 1: Due Now

# Strong MAC Construction (Fixed Length)

$$\mathrm{Mac}_{\mathrm{k}}(m) = \mathrm{F}_{\mathrm{K}}(m)$$

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

**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 $\Pi'$=(Mac',Vrfy'), a secure MAC for length n messages

**First: A few failed attempts**

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

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

What is wrong?

Block-reordering attack

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

$m_1$ = *"I love you"*

$m_2$ = "I will never say that"

$m_3$ = "you are stupid"

# MACs for Arbitrary Length Messages

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

**Attempt 2**

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

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

Addresses block-reordering attack.

Any other concerns?

Truncation attack!

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

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

# MACs for Arbitrary Length Messages

- Building Block $\Pi'$=(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 = $m_1,\ldots,m_d$ where each $m_i$ is n bits and m has length $\ell = nd$

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

Let $t_i = \mathrm{Mac}'_K(i \parallel \ell \parallel m_i)$ 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 $\Pi$'=(Mac',Vrfy'), a secure MAC for length n messages
- Let m = $m_1,\ldots,m_d$ 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 $\ell$ 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 $\ell$ as n/4 bit strings)
- **Output** $\langle r, t_1, \dots, t_d \rangle$

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

# Coming Soon

- CBC-MAC and Authenticated Encryption
- Read Katz and Lindell 4.4-4.5

# Week 3

Topics 2&3: Authenticated Encryption + CCA-Security

# Recap

- Message Authentication Codes
- Secrecy vs Confidentiality

**Today's Goals:**

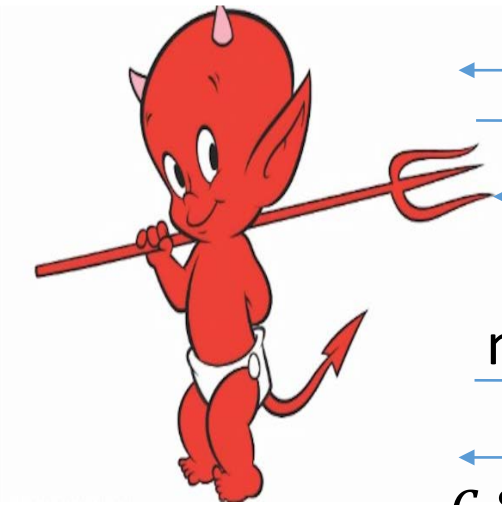
- Authenticated Encryption
- Build Authenticated Encryption Scheme with CCA-Security

# Authenticated Encryption

**Encryption:** Hides a message from the attacker

**Message Authentication Codes**: Prevents attacker from tampering with message

# Unforgeable Encryption Experiment ($\mathrm{Encforge}_{A,\Pi}(n)$)



$\mathrm{m}_1$

**$\mathrm{c}_1 = \mathrm{Enc}_K(\mathrm{m}_1)$**

$\mathrm{m}_2$

**$\mathrm{c}_2 = \mathrm{Enc}_K(\mathrm{m}_2)$**

⋮

$\mathrm{m}_q$

**$\mathrm{c}_q = \mathrm{Enc}_K(\mathrm{m}_q)$**

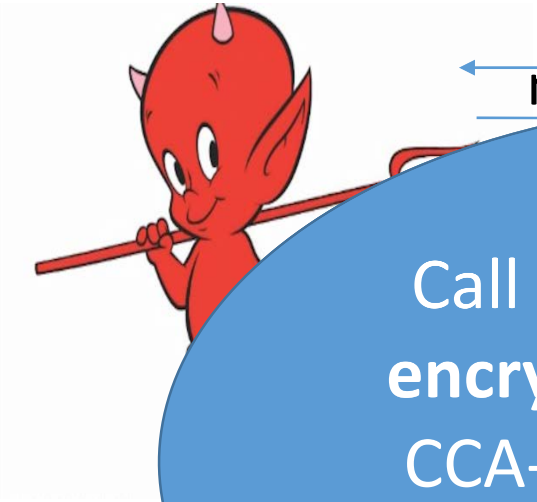
$c$ s.t $c \notin \{c_1, \ldots, \mathrm{c}_q\}$

$\mathrm{Encforge}_{A,\Pi}(n) = 1$ if $\mathrm{Deck}(c) \neq \perp$

**K = Gen(.)**

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

# Unforgeable Encryption Experiment ($\mathrm{Encforge}_{A,\Pi}(n)$)



$m_1$

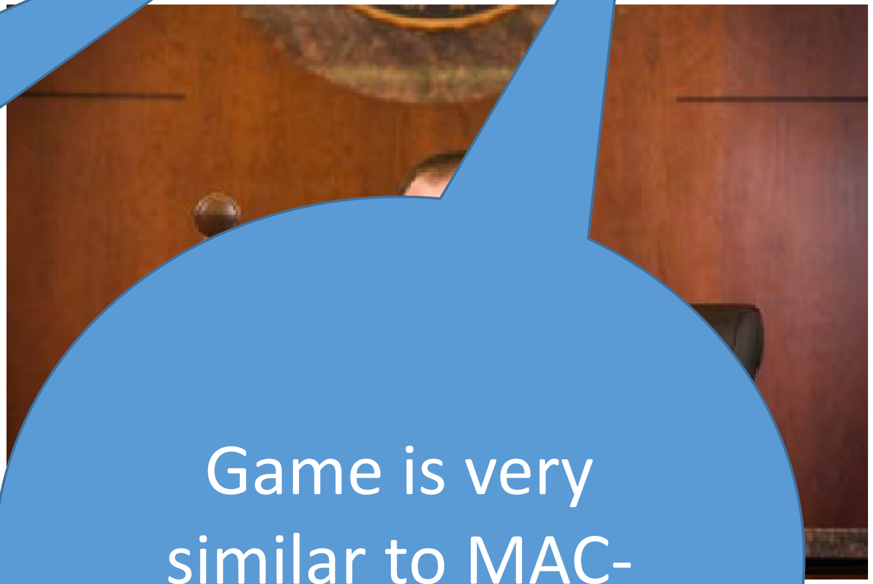
$c_1 = \mathrm{Enc}_K(m$

$m_2$

$_2)$

$(m_q)$

$\neq \perp$



Call $\Pi$ an **authenticated encryption scheme** if it is CCA-secure **and** any PPT attacker wins Encforge with negligible probability

Game is very similar to MAC-Forge game

(negligible) s.t

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

# Building Authenticated Encryption

**Attempt 1:** Let $\mathrm{Enc}'_K(m)$ be a CPA-Secure encryption scheme and let $\mathrm{Mac}'_K(m)$ be a secure MAC

$$Enc_K(m) = \langle \mathrm{Enc}'_K(m), \mathrm{Mac}'_K(m) \rangle$$

Any problems?

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

# Building Authenticated Encryption

**Attempt 1:**

$$Enc_K(m) = \langle r, F_k(r) \oplus m, F_k(m) \rangle$$

CPA-Attack:

- Intercept ciphertext c

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

- Ask to encrypt r

$$c_r = Enc_K(r) = \langle r', F_k(r') \oplus r, F_k(r) \rangle$$

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

# Building Authenticated Encryption

**Attempt 1:** Let $\mathrm{Enc}'_K(m)$ be a CPA-Secure encryption scheme and let $\mathrm{Mac}'_K(m)$ be a secure MAC

$$Enc_K(m) = \langle \mathrm{Enc}'_K(m), \mathrm{Mac}'_K(m) \rangle$$

Attack exploited fact that same secret key used for MAC'/Enc'

# Independent Key Principle

“different instances of cryptographic primitives should always use independent keys”

# Building Authenticated Encryption

**Attempt 2:** (Encrypt-and-Authenticate) Let $\mathrm{Enc}'_{K_E}(m)$ be a CPA-Secure encryption scheme and let $\mathrm{Mac}'_{K_M}(m)$ be a secure MAC. Let $K = (K_E, K_M)$ then

$$Enc_K(m) = \left\langle \mathrm{Enc}'_{K_E}(m), \mathrm{Mac}'_{K_M}(m) \right\rangle$$

Any problems?

$$\mathrm{Enc}'_{K_E}(m) = \left\langle r, F_{K_E}(r) \oplus m \right\rangle$$
$$\mathrm{Mac}'_{K_M}(m) = F_{K_M}(m)$$

# Building Authenticated Encryption

**Attempt 2:**

$$Enc_K(m) = \langle r, F_{K_E}(r) \oplus m, F_{K_M}(m) \rangle$$

CPA-Attack:

- Select $m_0$,$m_1$
- Obtain ciphertext c

$$c = \langle r, F_{K_E}(r) \oplus m_b, F_{K_M}(m_b) \rangle$$

- Ask to encrypt $m_0$

$$c_r = \langle r', F_{K_E}(r') \oplus m_0, F_{K_M}(m_0) \rangle$$

$$F_{K_M}(m_0) =? \, F_{K_M}(m_b)$$

# Building Authenticated Encryption

**Attempt 2:**

$$Enc_K(m) = \langle r, F_{K_E}(r) \oplus m, F_{K_M}(m) \rangle$$

CPA-Attack:

- Select $m_0, m_1$
- Obtain ciphertext c

$$c = \langle r, F_{K_E}(r) \oplus m_b, F_{K_M}$$

- Ask to encrypt $m_0$

$$c_r = \langle r', F_{K_E}(r') \oplus m_0, F_{K_M}(m_0$$

$$F_{K_M}(m_0) =? F_{K_M}(m_b)$$

Encrypt and Authenticate Paradigm does not work in general

# Building Authenticated Encryption

**Attempt 2:** (Encrypt-and-Authenticate) Let $\mathrm{Enc}'_{K_E}(m)$ be a CPA-Secure encryption scheme and let $\mathrm{Mac}'_{K_M}(m)$ be a secure MAC. Let $K = (K_E, K_M)$ then

$$Enc_K(m) = \langle \mathrm{Enc}'_{K_E}(m), \mathrm{Mac}'_{K_M}(m) \rangle$$

**Problem:** MAC security definition doesn't promise to hide m!

**This is what SSH does** ☹

# Building Authenticated Encryption

**Attempt 3:** (Authenticate-then-encrypt) Let $\mathrm{Enc}'_{K_E}(m)$ be a CPA-Secure encryption scheme and let $\mathrm{Mac}'_{K_M}(m)$ be a secure MAC. Let $K = (K_E, K_M)$ then

$$Enc_K(m) = \langle \mathrm{Enc}'_{K_E}(m \parallel t) \rangle \text{ where } \mathrm{t} = \mathrm{Mac}'_{K_M}(m)$$

- Used in SSL/TLS
- Not generically secure (Hugo Krawczyk)
- Easy to make mistakes when implementing (e.g., Lucky13 attack on TLS)

The Order of Encryption and Authentication for Protecting Communications (or: How Secure Is SSL?)

# Authenticate-then-Encrypt: A Bad Case

**Attempt 3:** (Authenticate-then-encrypt)

$$Enc_K(m) = \langle \mathrm{Enc}'_{K_E}(m \parallel t) \rangle \text{ where } \ \mathrm{t} = \mathrm{Mac}'_{K_M}(m)$$

**(Contrived? Plausible?) bad case:**

$\mathrm{Enc}'_{K_E}(m) = ECC\left(\langle r, F_{K_E}(r) \oplus m \rangle\right)$

Error Correcting Code

$\mathrm{D}ec'_{K_E}(c)$

$\langle r, s \rangle \coloneqq ECCD(c)$

Return $m = F_{K_E}(r) \oplus s$

# Authenticate-then-Encrypt: A Bad Case

**Attempt 3:** (Authenticate-then-encrypt)

$$Enc_K(m) = \langle \mathrm{Enc}'_{K_E}(m \parallel t) \rangle \text{ where } \ \mathrm{t} = \mathrm{Mac}'_{K_M}(m)$$

**(Contrived? Plausible?) bad case:**

$$\mathrm{Enc}'_{K_E}(m) = ECC\big(\langle r, F_{K_E}(r) \oplus m \rangle\big)$$

Error Correcting Code

$$ECC(101) = 111100001111$$

Ties?

$$ECCD(1100) = 1$$

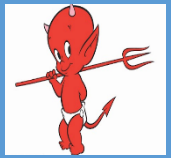
$$ECCD(0011) = 1$$

$Dec'_{K_E}(c)$

$\langle r, s \rangle \coloneqq ECCD(c)$

Return $m = F_{K_E}(r) \oplus s$

# Authenticate-then-Encrypt: A Bad C

Can learn tag and message bit by bit by repeatedly querying decryption oracle! 

ncryp

$t))\rangle$

$\oplus m\rangle)$

Error Correcting Code

$ECC(101) = 111100001111$

$ECCD(1100) = 1$

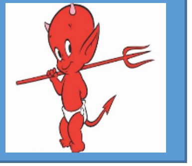
$ECCD(0011) = 1$

$Dec'_{K_E}(c)$

$\langle r, s\rangle := ECCD(c)$

Return $m = F_{K_E}(r) \oplus s$

1. Attacker obtains c $= ECC(\langle r, s = F_{K_E}(r) \oplus (m \parallel t)\rangle)$
2. Attacker asks for decryption of c$' = ECC(\langle r, s\rangle) \oplus (0 \ldots 0 \parallel 0011)$
   - What happens if last bit of s is a zero?
   - Answer: decryption error since $t' = t \oplus (0 \ldots 0 \parallel 1)$!
   - $ECCD(c') = \langle r, s' = s \oplus (0 \ldots 0 \parallel 1)\rangle$
3. What happens if last bit of s is a one?
   - Answer: Valid! $ECCD(c) = ECCD(c')$



# Building Authenticated Encryption

**Attempt 4:** (Encrypt-then-authenticate) Let $\mathrm{Enc}'_{K_E}(m)$ be a CPA-Secure encryption scheme and let $\mathrm{Mac}'_{K_M}(m)$ be a secure MAC. Let $K = (K_E, K_M)$ then

$$Enc_K(m) = \langle \mathrm{c}, \mathrm{Mac}'_{K_M}(\mathrm{c}) \rangle \text{ where } \mathrm{c} = \mathrm{Enc}'_{K_E}(m)$$

Secure?

# Building Authenticated Encryption

**Theorem:** (Encrypt-then-authenticate) Let $\mathrm{Enc}'_{K_E}(m)$ be a CPA-Secure encryption scheme and let $\mathrm{Mac}'_{K_M}(m)$ be a secure MAC. Then the following construction is an authenticated encryption scheme.

$$Enc_K(m) = \left\langle \mathrm{c}, \mathrm{Mac}'_{K_M}(\mathrm{c}) \right\rangle \text{ where } \mathrm{c} = \mathrm{Enc}'_{K_E}(m)$$

Proof?

Two Tasks:

$\mathrm{Encforge}_{A,\Pi}$

CCA-Security

# Building Authenticated Encryption

**Theorem:** (Encrypt-then-authenticate) Let $\mathrm{Enc}'_{K_E}(m)$ be a CPA-Secure encryption scheme and let $\mathrm{Mac}'_{K_M}(m)$ be a secure MAC. Then the following construction is an authenticated encryption scheme.

$$Enc_K(m) = \left\langle \mathrm{c}, \mathrm{Mac}'_{K_M}(\mathrm{c}) \right\rangle \text{ where } \mathrm{c} = \mathrm{Enc}'_{K_E}(m)$$

**Proof Intuition:** Suppose that we have already shown that any PPT attacker wins $\mathrm{Encforge}_{A,\Pi}$ with negligible probability.

Why does CCA-Security now follow from CPA-Security?

CCA-Attacker has decryption oracle, but cannot exploit it! Why?

Always sees ⊥ “invalid ciphertext” when he query with unseen ciphertext

# Proof Sketch

1. Let ValidDecQuery be event that attacker submits new/valid ciphertext to decryption oracle

2. Show Pr[ValidDecQuery] is negl(n) for any PPT attacker
   - Hint: Follows from strong security of MAC since
   $$Enc_K(m) = \langle \mathrm{c}, \mathrm{Mac}'_{K_M}(\mathrm{c}) \rangle$$
   - This also implies unforgeability (even if we gave the attacker $K_E$!).

3. Show that attacker who does not issue valid decryption query wins CCA-security game with probability ½ + negl(n)
   - Hint: otherwise we can use A to break CPA-security
   - Hint 2: simulate decryption oracle by always returning ⊥ when given new ciphertext

# Secure Communication Session

- Solution Protocol? Alice transmits $c_1 = Enc_K(m_1)$ to Bob, who decrypts and sends Alice $c_2 = Enc_K(m_2)$ etc…
- Authenticated Encryption scheme is
  - Stateless
  - For fixed length-messages
- We still need to worry about
  - Re-ordering attacks
    - Alice sends 2n-bit message to Bob as $c_1 = Enc_K(m_1)$, $c_2 = Enc_K(m_2)$
  - Replay Attacks
    - Attacker who intercepts message $c_1 = Enc_K(m_1)$ can replay this message later in the conversation
  - Reflection Attack
    - Attacker intercepts message $c_1 = Enc_K(m_1)$ sent from Alice to Bob and replays to $c_1$ Alice only

# Secure Communication Session

- Defense
  - Counters ($CTR_{A,B}$,$CTR_{B,A}$)
    - Number of messages sent from Alice to Bob ($CTR_{A,B}$) --- initially 0
    - Number of messages sent from Bob to Alice ($CTR_{B,A}$) --- initially 0
    - Protects against Re-ordering and Replay attacks
  - Directionality Bit
    - $b_{A,B} = 0$ and $b_{B,A} = 1$ (e.g., since A < B)

- Alice: To send m to Bob, set $c=Enc_K(b_{A,B} \| CTR_{A,B} \| m)$, send c and increment $CTR_{A,B}$
- Bob: Decrypts c, (if $\perp$ then reject), obtain $b \| CTR \| m$
  - If $CTR \neq CTR_{A,B}$ or $b \neq b_{A,B}$ then reject
  - Otherwise, output m and increment $CTR_{A,B}$

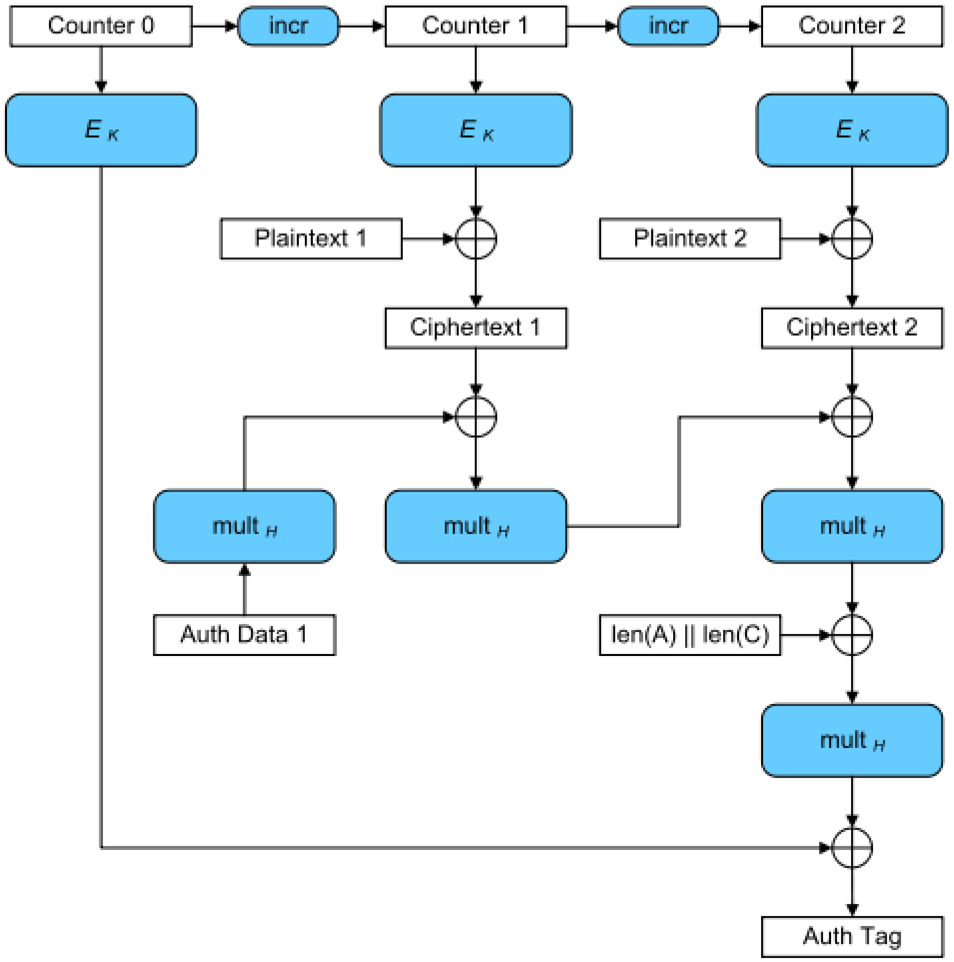
# Authenticated Security vs CCA-Security

- Authenticated Encryption → CCA-Security (by definition)

- CCA-Security does not necessarily imply Authenticate Encryption
  - But most natural CCA-Secure constructions are also Authenticated Encryption Schemes
  - Some constructions are CCA-Secure, but do not provide Authenticated Encryptions, but they are less efficient.
- Conceptual Distinction
  - CCA-Security the goal is secrecy (hide message from active adversary)
  - Authenticated Encryption: the goal is integrity + secrecy

# Galois Counter Mode (GCM)

- AES-GCM is an Authenticated Encryption Scheme
- **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!



# Next Week

- Read Katz and Lindell 5.1-5.6
- Cryptographic Hash Functions
- HMACs
- Generic Attacks on Hash Functions
- Random Oracle Model
- Applications of Hashing
- Homework 2 Assigned