Cryptography
CS 555

Week 4:

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

Readings: Katz and Lindell Chapter 4.1-4.4

Homework 1 Solutions Released

Homework 2 Released: Due Feb 18 @11:59PM on Gradescope

Spring 2021

Recap

- Chosen Plaintext Attacks/Chosen Ciphertext Attacks
    - CPA vs CCA-security

- Blockciphers and Modes of Operation

- Message Authentication Codes
    - ~~Confidentiality~~ vs Integrity
    - Canonical Verification and Timing Side Channel

Current Goal:

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

Week 4: Topic 1: Constructing Message Authentication Codes

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.

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

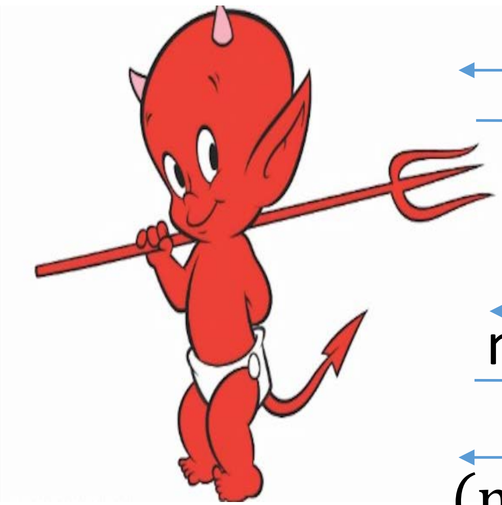
$$\mathrm{Mac}_{\mathrm{k}}(m) = \mathrm{F}_{\mathrm{K}}(m)$$

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

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

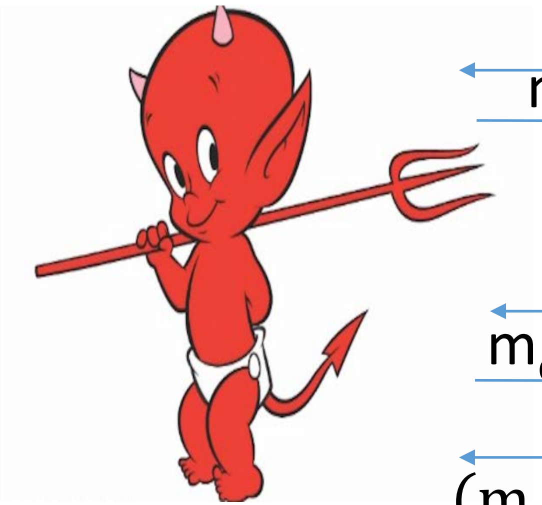
(m, t) s.t $(m, t) \notin \{(m_1, t_1), \ldots, (m_q, t_q)\}$

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

K = Gen(.)

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

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



m_1

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

m_2

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

⋮

m_q

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

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

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

K = Gen(.)

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

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

Strong MAC Construction (Fixed Length)

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

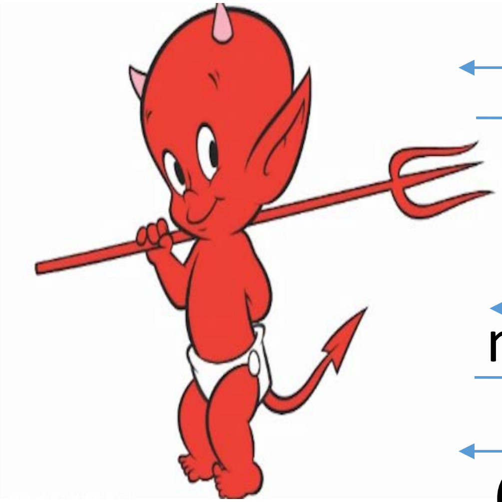
$$\mathrm{Vrfy}_{\mathrm{k}}(m, t) = \begin{cases} 1 & \text{if } t = \mathrm{F}_{\mathrm{K}}(m) \\ 0 & \text{otherwise} \end{cases}$$

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



m_1

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

m_2

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

$\vdots$

m_q

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

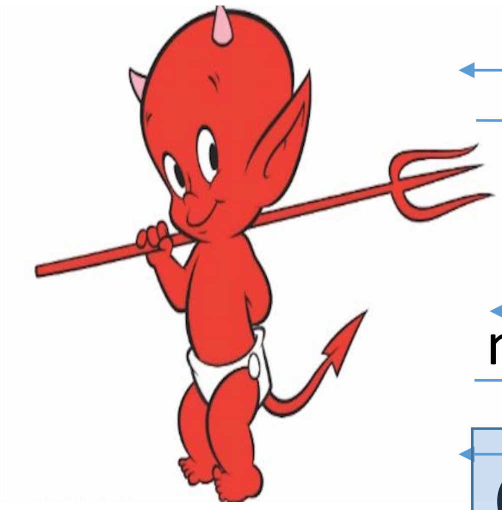
(m, t) s.t $m \notin \{m_1, \dots, m_q\}$

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

K = Gen(.)

$$\exists PPT\ A\ and\ g(.)\ \text{(positive/non negligible) s.t}$$
$$\Pr[\mathrm{Macforge}_{A,\Pi}(n) = 1] > g(n)$$

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



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 $m \notin \{m_1, \ldots, m_q\}$

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

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

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

- If O=F_K then

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

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)

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

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

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

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 $\left(2^n, 2^{n/2}, 2^{-n}\right)$-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)

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

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

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 Π'=(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 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 = $m_1,\ldots,m_d$ 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 Π'=(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 ℓ as $\frac{n}{4}$ bit strings)
- **Output** $\langle r, t_1, \ldots, t_d \rangle$

MACs for Arbitrary Length Messages

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

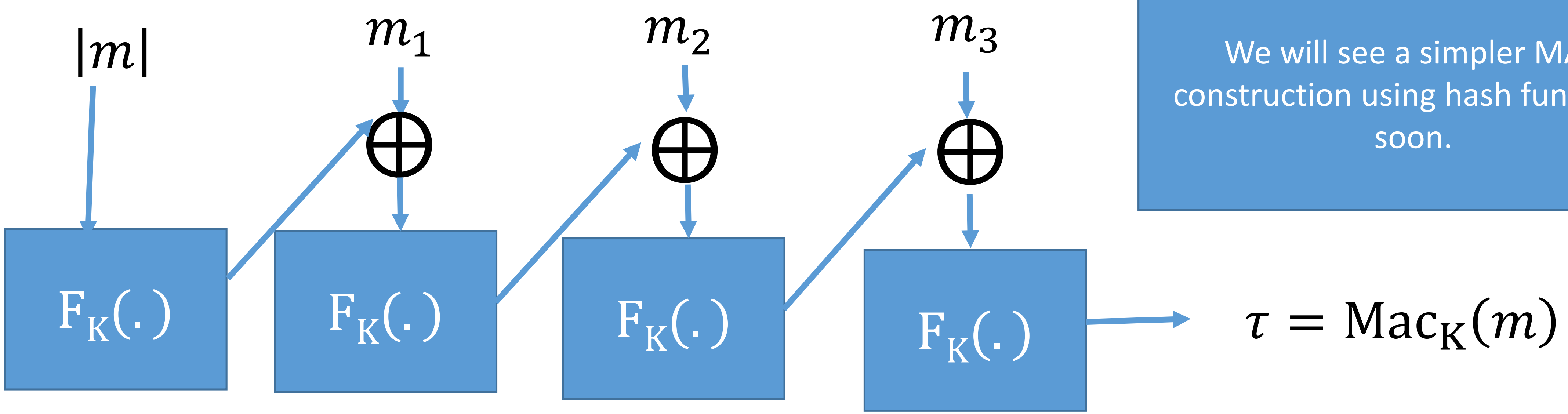
- Select random n/4 bit string r
- Let $t_i = \text{Mac}'_K(r \parallel \ell \parallel i \parallel m_i)$ for i=1,…,d
  - (Note: encode i and ℓ as n/4 bit strings)
- **Output** $\langle r, t_1, \ldots, t_d \rangle$

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.

CBC-MAC

Caveat: Tricky Padding Issues arise if |m| is not a multiple of the block-length. See textbook.

We will see a simpler MAC construction using hash functions soon.



Advantages over Previous Solution

- Both MACs are secure
- Works for unbounded length messages
- Canonical Verification
- Short Authentication tag
- ~~Parallelizable~~

for i=1,…,d

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

(encode i and ℓ as n/4 bit strings)

Output $\langle r, t_1, \ldots, t_d \rangle$

Coming Soon

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

Week 4

Topics 2&3: Authenticated Encryption + CCA-Security

Recap

- Message Authentication Codes
- Secrecy vs Confidentiality

Today's Goals:

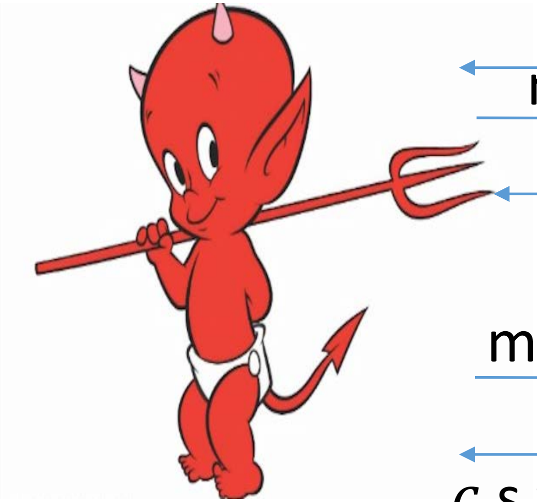
- Authenticated Encryption
- Build Authenticated Encryption Scheme with CCA-Security

Authenticated Encryption

Encryption: Hides a message from the attacker

Message Authentication Codes: Prevents attacker from tampering with message

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



m_1

$c_1 = \text{Enc}_K(m_1)$

m_2

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

$\vdots$

m_q

$c_q = \text{Enc}_K(m_q)$

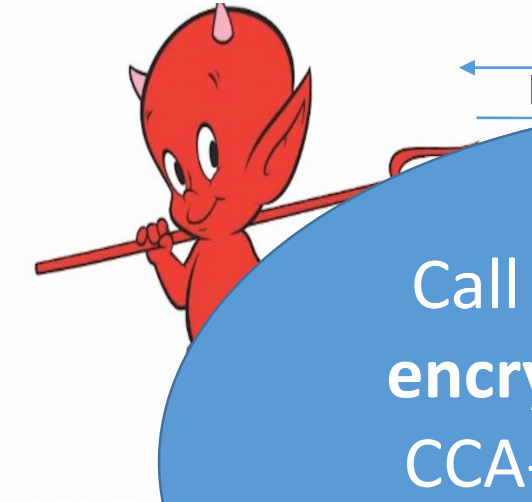
c s.t $c \notin \{c_1, \ldots, c_q\}$

$\text{Encforge}_{A,\Pi}(n) = 1 \text{ if } \text{Dec}_K(c) \neq \perp$

K = Gen(.)

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

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



m_1

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

m_2

$_2)$

(m_q)

$\neq\perp$

Call Π 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: (Encrypt-and-Authenticate)

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

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: (Encrypt-**and**-Authenticate)

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

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 SSL 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:

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

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:

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

Error Correcting Code

$$ECC(101) = 111100001111$$

Ties?

$$ECCD(1100) = 1$$

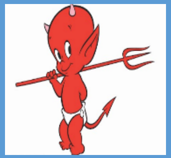
$$ECCD(0011) = 1$$

$Dec'_{K_E}(c)$

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

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))

⊕m⟩)

Error Correcting Code

$$ECC(101) = 111100001111$$

$$ECCD(1100) = 1$$

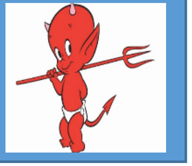
$$ECCD(0011) = 1$$

$$Dec'_{K_E}(c)$$

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

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

1. Attacker obtains c $= ECC\left(\langle r, s = F_{K_E}(r) \oplus (m \parallel t)\rangle\right)$
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 was 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 strongly 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?

Recap

- MACs for Unbounded Length Messages
  - Reordering/Truncation/Block Swapping Attacks
  - Nonce Based Construction
  - CBC MAC
- Authenticated Encryption = CCA-Secure + Unforgeable Encryptions
  - Independent Key Principle
  - Encrypt and Authenticate
    - Not generically secure
  - Authenticate then Encrypt
    - Not generically secure
  - Encrypt then Authenticate
    - Always secure given CPA-Secure encryption + strongly secure MAC

Announcement: Quiz 2 Released today due by Saturday at 11:30PM on Brightspace

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 **strongly** secure MAC. Then the following construction is an authenticated encryption scheme.

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

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 **strongly** secure MAC. Then the following construction is an authenticated encryption scheme.

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

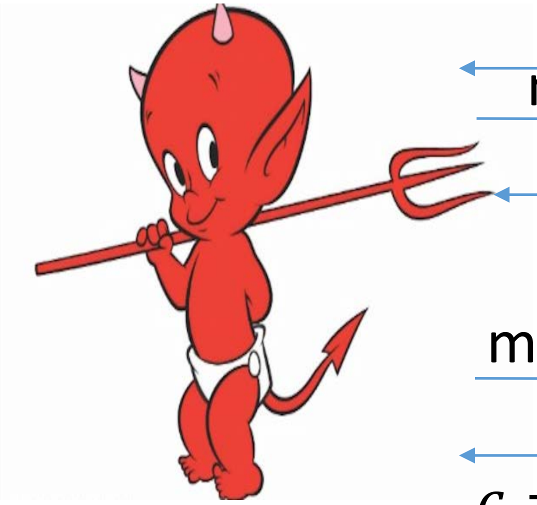
Proof Intuition: Suppose that we have already shown that any PPT attacker wins $\mathrm{Encforge}_{A,\Pi}$ with negligible probability.

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

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

Always sees $\perp$ “invalid ciphertext” when he query with unseen ciphertext

Encryption Forgery Attacker ($\mathrm{Encforge}_{A,\Pi}(n)$)



m_1

$\mathbf{c}_1 = \langle \mathbf{c}_1{}', \mathrm{Mac}'_{K_M}(\mathbf{c}_1{}') \rangle$

m_2

$\mathbf{c}_2 = \langle \mathbf{c}_2{}', \mathrm{Mac}'_{K_M}(\mathbf{c}_2{}') \rangle$

$\vdots$

m_q

$\mathbf{c}_\mathrm{q} = \langle \mathbf{c}_\mathrm{q}{}', \mathrm{Mac}'_{K_M}(\mathbf{c}_\mathrm{q}{}') \rangle$

$c = \langle m, t \rangle$ s.t $c \notin \{c_1, \ldots, c_\mathrm{q}\}$

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

K = Gen(.)

iff $\mathrm{Verify}'_{K_M}(m, t) = 1$

$$\langle m, t \rangle \notin \{\langle \mathbf{c}_1{}', \mathbf{t}_1{}' \rangle, \ldots, \langle \mathbf{c}_\mathrm{q}{}', \mathbf{t}_\mathrm{q}{}' \rangle\}$$

$\downarrow$

MAC forgery for the key K_M

Unforgeable Encryptions

Theorem: (Encrypt-then-authenticate) Let $\mathrm{Mac}'_{K_M}(m)$ be a strongly secure MAC. Then the following construction has unforgeable encryptions.

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

Note: unforgeable property holds even if the encryption scheme is not CPA-Secure.

Reduction: MAC Attacker A' picks key $\mathrm{K_E}$ and simulates encryption forgery attacker A (Note: A' plays the role of the challenger in the encryption forgery game).

Whenever A submits query m_i to encryption oracle A' responds by

1. computes $\mathbf{c}_{i'} = \mathrm{Enc}'_{\mathrm{K_E}}(\mathrm{m})$ and
2. sends $\mathbf{c}_{i'}$ to MAC challenger to get $\mathrm{Mac}'_{\mathrm{K_M}}(\mathbf{c}_{i'})$ and
3. sends $\mathbf{c}_i = \langle \mathbf{c}_{i'}, \mathrm{Mac}'_{K_M}(\mathbf{c}_{i'}) \rangle$ back to A.

Whenever A outputs a forged ciphertext $\mathrm{c} = \langle c', t' \rangle$ we output the pair (m=c',t=t') as our MAC forgery

Unforgeable Encryptions

Reduction: MAC Attacker A' picks key K_E and simulates encryption forgery attacker A

(Note: A' plays the role of the challenger in the encryption forgery game).

Whenever A submits query m_i to encryption oracle A' responds by

1. computes $\mathbf{c_{i'}} = \mathrm{Enc}'_{K_E}(\mathrm{m})$ and
2. sends $\mathbf{c_{i'}}$ to MAC challenger to get $\mathrm{Mac}'_{K_M}(\mathbf{c_{i'}})$ and
3. sends $\mathbf{c_i} = \langle \mathbf{c_{i'}}, \mathrm{Mac}'_{K_M}(\mathbf{c_{i'}}) \rangle$ back to A.

Whenever A outputs a forged ciphertext $\mathrm{c} = \langle c', t' \rangle$ we output the pair (m=c',t=t') as our MAC forgery.

Fact: A' wins the MAC forgery game if and only if A wins the encryption forgery game.

Proof Sketch (CCA-Security)

1. Let ValidDecQuery be event that attacker submits new/valid ciphertext to decryption oracle at any point in time

2. Show Pr[ValidDecQuery] is negl(n) for any PPT CCA attacker A
   - If not then we could win encryption forgery game with probability at least Pr[ValidDecQuery]/q where q is the number of queries to the decryption oracle
   - **Reduction Challenge:** *a priori* don't know which query i* to decryption oracle yields encryption forgery
   - **Solution:** Guess index i of query $\Pr[\mathrm{i}=\mathrm{i}^*] \geq \frac{1}{q}$
   - We win the encryption forgery game if the event ValidDecQuery occurs and we guessed correctly $\mathrm{i}=\mathrm{i}^*$
     - $\Pr[\text{Win Enc Forgery}] \geq \Pr[\text{ValidDecQuery} \wedge \mathrm{i}=\mathrm{i}^*] \geq \frac{\Pr[\text{ValidDecQuery}]}{q}$
     - If $\Pr[\text{ValidDecQuery}]$ is non-negligible so is $\Pr[\text{Win Enc Forgery}]$

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
   - 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)
   - **Key Idea:** Given attacker A breaking CCA-Security we can build A' which breaks CPA-security of $\mathrm{Enc}'_{\mathrm{K_E}}$

Proof Sketch

3. Show that attacker who does not issue valid decryption query wins CCA-security game with probability ½ + negl(n)
   - **Key Idea:** Given attacker A breaking CCA-Security we can build A' which breaks CPA-security of $\mathrm{Enc}'_{\mathrm{K_E}}$

Reduction: CPA attacker A' picks MAC key $\mathrm{K_M}$ and simulates CCA-Attacker A (A' plays role of CCA challenger)

Whenever A queries encryption oracle on message m

A' forwards encryption oracle to CPA challenger to get $\mathrm{c}' = \mathrm{Enc}'_{\mathrm{K_E}}(m)$

A' computes $\mathrm{t} = \mathrm{MAC}_{\mathrm{K_M}}(\mathrm{c}')$ and responds with $\mathrm{c} = (\mathrm{c}, \mathrm{t})$

Whenever A queries the decryption oracle on a ciphertext c

If c is the fresh ciphertext then respond with ⊥ (failure)

(If c was produced in response to a query then simply respond with the original message m)

Finally, A' outputs the same guess b' as A.

Claim: $\Pr[\mathrm{PrivK}^{cpa}_{A',\Pi'}(n)] \geq \Pr[\mathrm{PrivK}^{cca}_{A,\Pi}(n) | \overline{\mathrm{ValidDecQuery}}]\, \Pr[\overline{\mathrm{ValidDecQuery}}]$

➔ If $\Pr[\mathrm{PrivK}^{cca}_{A,\Pi}(n)]$ is non-negligible then so is $\Pr[\mathrm{PrivK}^{cpa}_{A',\Pi'}(n)]$

If A breaks CCA-security of our construction Π then A' breaks CPA-security of Π'
(Contradiction! $\mathrm{Enc}'_{\mathrm{K_E}}$ is assumed to be CPA-secure)

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 (or Truncation)
    - Alice sends three n-bit message to Bob as $c_1 = Enc_K(m_1)$, $c_2 = Enc_K(m_2)$, $c_3 = Enc_K(m_3)$. Mallory can reorder
    - $m_1 =$ *"I love you"*, $m_2 =$ "I will never say that", $m_3 =$ "you are stupid"
  - Replay Attacks
    - Mallory intercepts ciphertext $c_3 = Enc_K(m_3)$ and can now replay the message m_3 later in the conversation
  - Reflection Attack
    - Attacker intercepts message $c_1 = Enc_K(m_1)$ sent from Alice to Bob and Mallory reply's to Alice with c_1

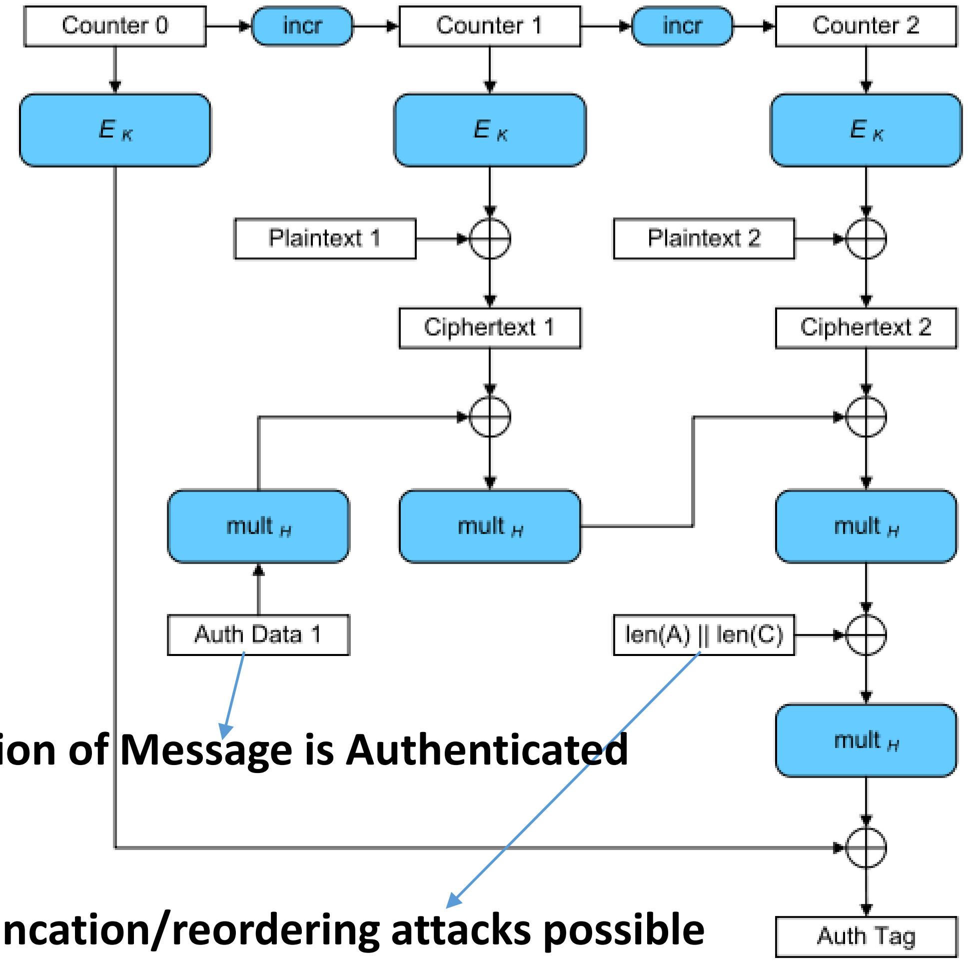
Secure Communication Session

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

Galois Counter Mode (GCM)

- AES-GCM is an Authenticated Encryption Scheme
  - Encrypt then Authenticate
  - Only uses one symmetric key, but still secure ☺
- **Bonus:** Authentication Encryption with Associated Data
  - Associated Data incorporated into MAC
  - Ensures attacker cannot tamper with associated packet data
    - Source IP
    - Destination IP
    - Why can't these values be encrypted?
- Encryption is largely parallelizable!



Direction of Message is Authenticated

No truncation/reordering attacks possible amongst the blocks of this message

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

Week 4: Topic 4: Cryptographic Hash Functions

Hash Functions

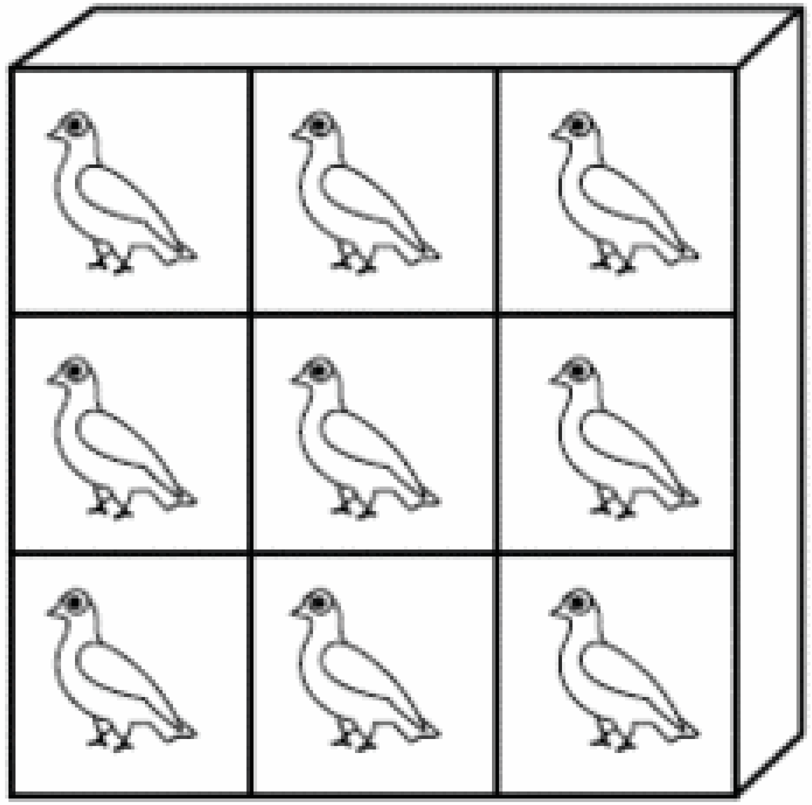
$$H(x)=y$$

Long Input: $\mathbf{x}$

Short Output: $\mathbf{y}$ s.t.
$|\mathbf{y}| \ll |\mathbf{x}|$

Pigeonhole Principle

"You cannot fit 10 pigeons into 9 pigeonholes"

Hash Collisions

By Pigeonhole Principle there must exist x and y s.t.

$$H(x) = H(y)$$

Classical Hash Function Applications

- Hash Tables
  - O(1) lookup*
- “Good hash function” should yield “few collisions”

* Certain terms and conditions apply

Collision-Resistant Hash Function

Intuition: Hard for computationally bounded attacker to find *any pair* x, x' s.t.

$$H(x) = H(x')$$

How to formalize this intuition?

- **Attempt 1:** For all PPT A,

$$\Pr[A(1^n) = (x, x')\ s.t\ H(x) = H(x')] \leq negl(n)$$

- **The Problem:** Let x, x' be given s.t. $H(x) = H(x')$

$$A_{x,x'}(1^n) = (x, x')$$

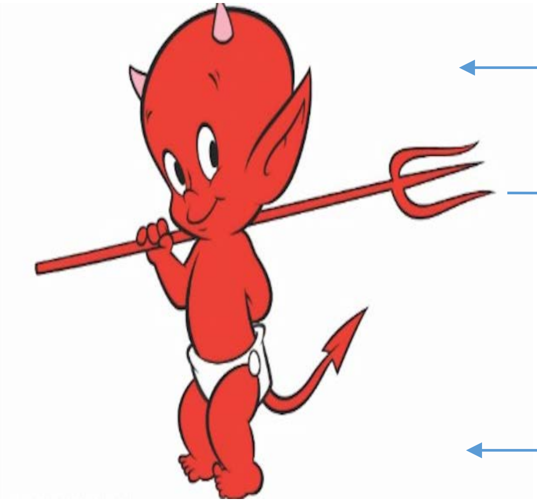
- We are assuming that |x| > |H(x)|. Why?
  - H(x)=x is perfectly collision resistant! (but with no compression)

Keyed Hash Function Syntax

- Two Algorithms
  - $\mathrm{Gen}(1^n; R)$ (Key-generation algorithm)
    - **Input:** Random Bits R
    - **Output:** ~~Secret~~ key s
  - $H^s(m)$ (Hashing Algorithm)
    - **Input:** key s and message $\mathrm{m} \in \{0,1\}^*$ (unbounded length)
    - **Output:** hash value $H^s(m) \in \{0,1\}^{\ell(n)}$

- Fixed length hash function
  - $m \in \{0,1\}^{\ell'(n)}$ with $\ell'(n) > \ell(n)$

Collision Experiment ($HashColl_{A,\Pi}(n)$)



s

x_1, x_2

$$HashColl_{A,\Pi}(n) = \begin{cases} 1 & if\ H^s(x_1) = H^s(x_2) \\ 0 & otherwise \end{cases}$$

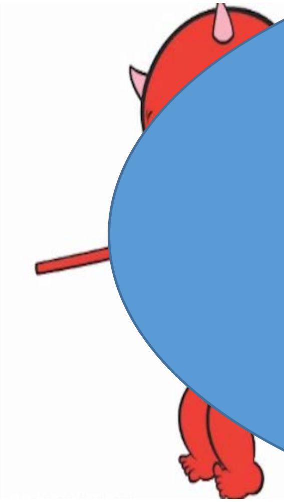
$s = \mathrm{Gen}(1^n; R)$

Definition: (Gen,H) is a collision resistant hash function if

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

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

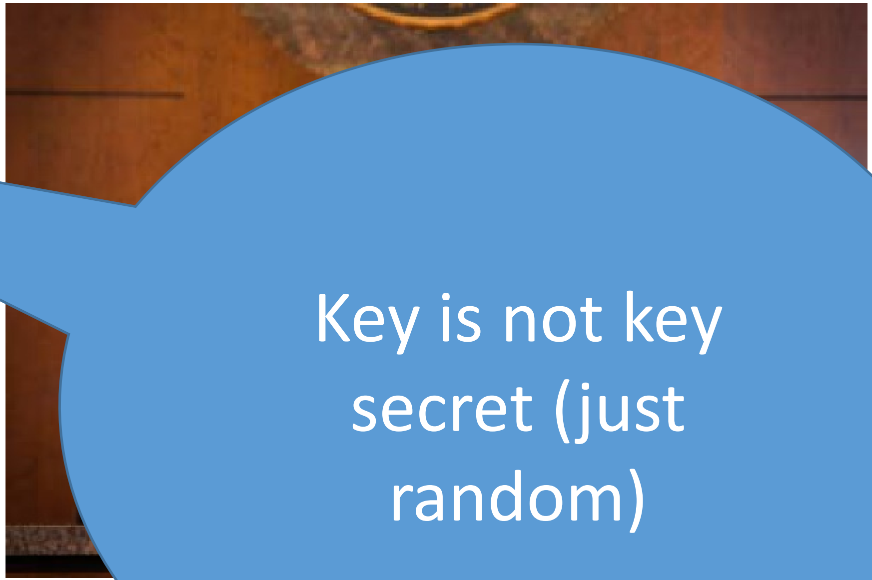
Collision Experiment ($HashColl_{A,\Pi}(n)$)



For simplicity we will sometimes just say that H (or H^s) is a collision resistant hash function

$= H^s(x_2)$

ise



Key is not key secret (just random)

Definition: (Gen,H) is a collision resistant hash function if

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

$$\Pr\left[HashColl_{A,\Pi}(n)=1\right] \le \mu(n)$$

Theory vs Practice

- Most cryptographic hash functions used in practice are un-keyed
  - Examples: MD5, SHA1, SHA2, SHA3
- Tricky to formally define collision resistance for keyless hash function
  - There is a PPT algorithm to find collisions
  - We just usually can't find this algorithm ☺

Formalizing Human Ignorance:
Collision-Resistant Hashing without the Keys

Phillip Rogaway

Department of Computer Science, University of California,
Davis, California 95616, USA, and
Department of Computer Science, Faculty of Science,
Chiang Mai University, Chiang Mai 50200, Thailand
rogaway@cs.ucdavis.edu

31 January 2007

Abstract. There is a foundational problem involving collision-resistant hash-functions: com-