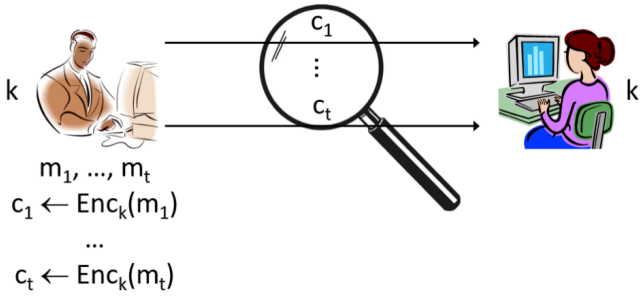
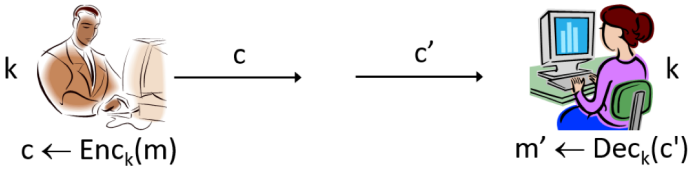
Passive & active attack

- So far, we have been assuming only a *passive, eavesdropping* attacker
  - Even if it can carry out *chosen-plaintext attacks* (CPA)



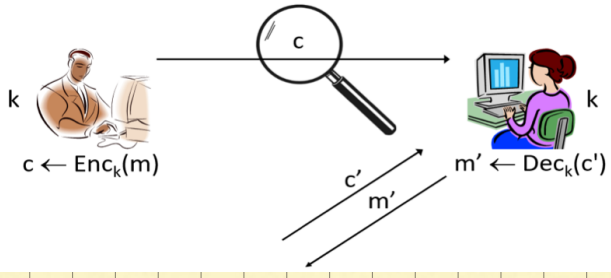
CPA只查看密文，但并不篡改信道上的密文.

- What if the attacker can be *active*?
  - E.g., interfering with the communication channel



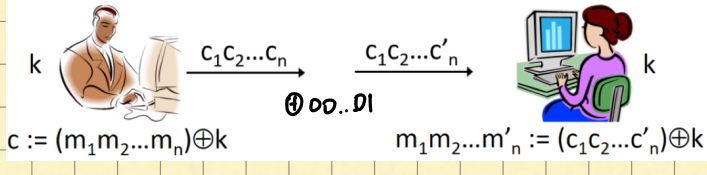
主动攻击者：
会篡改信道上的密文

  - E.g., "impersonating" the sender; injecting communication on the channel



Malleability 可延展性

- (Informal): A scheme is *malleable* if it is possible to modify a ciphertext and thereby cause a *predictable* change to the plaintext
- *Malleability* can be dangerous!
  - E.g., encrypted bank transactions
- All the schemes we have seen so far are *malleable*
  - E.g., the one-time pad ...



可以修改密文且因此对明文的改变是可预知的，即为可延展的

窃听信道中传输的ciphertext，并⊕0...01（即只修改最低位）后再传，可以预知receiver信息的变化

Chosen-ciphertext attacks

- Models settings in which the attacker can influence what gets *decrypted*, and observe the effects
  - How to model?
- Allow attackers to submit ciphertexts of its choice (with one restriction) to the receiver, and learn the corresponding plaintext
  - In addition to being able to carry out a *chosen-plaintext attack*

CCA-security

- Define a randomized experiment $PrivCCA_{A,\Pi}(n)$:
  1. $k \leftarrow Gen(1^n)$
  2. $A(1^n)$ interacts with an *encryption oracle* $Enc_k(\cdot)$, and a *decryption oracle* $Dec_k(\cdot)$, and then outputs m_0, m_1 of the same length
  3. $b \leftarrow \{0,1\}$, $c \leftarrow Enc_k(m_b)$, give c to A
  4. A can continue to interact with $Enc_k(\cdot)$, $Dec_k(\cdot)$, but may not request decryption of c
  5. A outputs b'; A succeeds if $b = b'$, and experiment evaluates to 1 in this case

Definition 6.1 Π is *secure against chosen-ciphertext attacks (CCA-secure)* if for all PPT attackers A, there is a *negligible* function ϵ such that

$$\Pr[PrivCCA_{A,\Pi}(n) = 1] \leq 1/2 + \epsilon(n)$$

能任选明文让 $Enc_k(\cdot)$ 加密
任选密文让 $Dec_k(\cdot)$ 解密
(但不能解密需要猜的密文)

Chosen-ciphertext attacks and malleability

- If a scheme is *malleable*, then it cannot be *CCA-secure*
  - Modify c, submit the modified ciphertext c' to the decryption oracle and determine original message based on the result
- *CCA-security* implies *non-malleability*

$Gen(1^n)$: choose a uniform key $k \in \{0,1\}^n$
$Enc_k(m)$, for $|m| = |k|$
- Choose uniform $r \in \{0,1\}^n$ (*nonce/ initialization vector*)
- Output ciphertext $\langle r, F_k(r) \oplus m \rangle$

$Dec_k(c_1, c_2)$: output $c_2 \oplus F_k(c_1)$

Theorem 5.1 If F is a pseudorandom function, then this scheme is *CPA-secure*.

若scheme是malleable的，则其不可能CCA安全.

需要加上MAC才能是CCA安全
(message authentication code)

- In the definition of *CCA-security*, the attacker can obtain the decryption of any ciphertext of its choice (besides the challenge ciphertext)
  - Is this realistic?
- We show a scenario where:
  - *One bit* about decrypted ciphertexts is leaked
  - The scenario occurs in the real world!
  - This can be exploited to learn the entire plaintext

只靠泄露出的1-bit信息还原出全部明文.

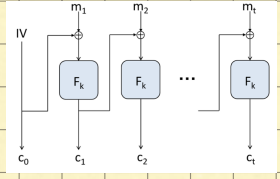
Arbitrary-length messages

- Message → encoded data → ciphertext
- PKCS #5 encoding:
  - Assume message is an integeral # of bytes
  - Let L be the block length (in bytes) of the cipher
  - Let $b \geq 1$ be # of bytes that need to be appended to the message to get length a multiple of L
  - $1 \leq b \leq L$; note $b \neq 0$
  - Append b (encoded in 1 byte), b times
  - I.e., if 3 bytes of padding are needed, append 0x030303

message需先补足为L的倍数.

不能b=0而能b=L是因为b=0时会有歧义
设L=6，若有message为0x0604…05时，将不确定末尾的0x05是由于取b=L-1填补上的还是b=0时message就有的信息.

- To Decrypt:
  - Use CBC-mode decryption to obtain encoded data
  - Say, the final byte of encoded data has value b
    - If $b = 0$ or $b > L$, return "error"
    - If final b bytes of encoded data are not all equal to b, return "error"
    - Otherwise, strip off the final b bytes of the encoded data, and output what remains as the message



Example ($L=8$)

msg: | AB | 01 | 4F | 21 | 00 | 7C |

encrypt

补0x0202 ($b=0x02$)

| AB | 01 | 4F | 21 | 00 | 7C | 02 | 02 |

| AB | 01 | 4F | 21 | 00 | 7C | 02 | 02 |

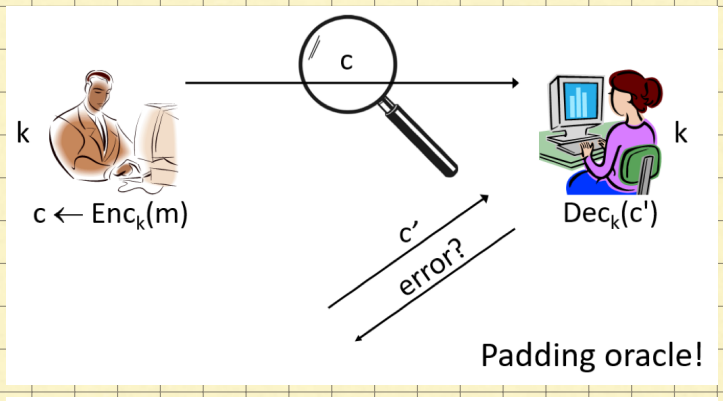
从最后一位知 $b=0x02$

decrypt

删去补充的0x0202

| AB | 01 | 4F | 21 | 00 | 7C |

Padding oracles



- Padding oracles are frequently present in, e.g., web applications
- Even if an error is not explicitly returned, an attacker might be able to detect differences in timing, behavior, etc.
- Main idea of the attack
  - Consider a two-block ciphertext IV, c
  - Encoded data $= F_k^{-1}(c) \oplus IV$
  - Main observation: If an attacker modifies the ith byte of IV, this causes a predictable change (only) to the ith byte of the encoded data

一般情况：$\begin{cases} C_0 = IV \\ C_1 = F_k(m_1 \oplus C_0) \end{cases}$ $m_1 = F_k^{-1}(C_1) \oplus C_0$

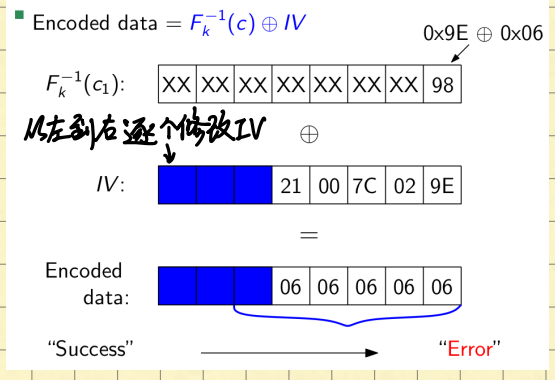
篡改后：$C_0' = C_0 \oplus \Delta$ $m_1' = F_k^{-1}(C_1') \oplus C_0' = m_1 \oplus \Delta$

因此对密文的修改明文变化是可预测的.

不断重复试不同的Δ，直到不报 error 则可知 i-th byte

① find b

- Encoded data $= F_k^{-1}(c) \oplus IV$



从左往右逐个改变IV的字节

当第一次发生error时，该字节即为b.

② find Δi

改动若知道明文以0x00...0xb结尾.

$\Delta i \overset{def}{=}$ 0x00 0x00 ... 0xi 0x(b+1) ... 0x(b+1) = 0x00 0x00 ... 0xi 0xb⊕(b+1) ... 0xb⊕(b+1)

⊕ 0x00 0x00 ... 0x00 0xb ... 0xb

然后提交 IV, C₁⊕Δi, C₂, ..., Cₙ 去解密. 得到的结果的最后(b+1)字节是 0x(M⊕i) 0x(b+1) ... 0x(b+1)，且仅当 0x(M⊕i) = 0x(b+1) 时不报错. 此时可还原出 M = 0x(b+1)⊕0xi

然后不断重复这步，每次0xb都+1. 可以从后向前逐步还原明文内容.

- Encoded data $= F_k^{-1}(c) \oplus IV$

$F_k^{-1}(c_1)$: | XX | XX | XX | XX | XX | XX | XX | 98 |

$\oplus$ 0x02 $\oplus$ 0x06 $\oplus$ 0x07

IV: | AB | 01 | 4F | 21 | 00 | 7C | 03 | 9F |

$=$

Encoded data: | XX | XX | 06 | 06 | 06 | 06 | 06 | 07 |

- Encoded data $= F_k^{-1}(c) \oplus IV$

$F_k^{-1}(c_1)$: | XX | XX | XX | XX | XX | XX | XX | 98 |

$\oplus$

IV: | AB | 41 | 4E | 20 | 01 | 7D | 03 | 9F |

$=$

Encoded data: | XX | 07 | 07 | 07 | 07 | 07 | 07 | 07 |

"Success"

$XX \oplus 0x41 = 0x07$

$\Rightarrow XX = 0x41 \oplus 0x07$

$\Rightarrow$ plaintext byte $= XX \oplus 0x01 = 0x47$

Attack complexity

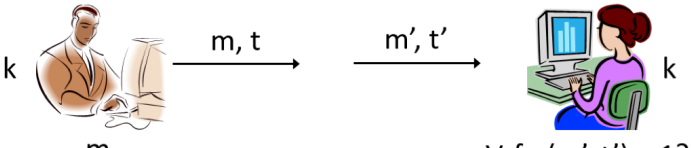
- $\leq L$ tries to learn the # of padding bytes (b)
- $\leq 2^8 = 256$ tries to learn each plaintext byte

CCA-security : Summary

- *Chosen-ciphertext attacks* represent a significant, real-world threat
- Modern encryption schemes are designed to be *CCA-secure*
- None of the schemes we have seen so far are *CCA-secure*

Secrecy vs. Integrity

- So far we have been concerned with ensuring *secrecy* of communication
- What about *integrity*?
  - I.e., ensuring that a received message originated from the intended party, and was not modified
    - Even if an attacker controls the channel!
  - Standard error-correction techniques not enough!
    - The right tool is a *message authentication code*



Security: 窃听者无法得知 message 内容.
Integrity: 窃听者无法篡改 ciphertext.

- *Secrecy* and *integrity* are *orthogonal* concerns
  - Possible to have either one without the other
  - Sometimes you might want one without the other
  - Most often, both are needed

- Encryption does not (in general) provide any integrity
  - Integrity is even stronger than *non-malleability*
  - None of the schemes we have seen so far provide any integrity

加密并不提供 integrity
integrity 比 non-malleability 更强

Message authentication code (MAC)

- A *message authentication code* is defined by three PPT algorithms (*Gen*, *Mac*, *Vrfy*):
  - *Gen*: take as input 1^n; outputs k. (Assume $|k| \geq n$.)
  - *Mac*: take as input key k and message $m \in \{0,1\}^*$; outputs *tag* t: $t := Mac_k(m)$
  - *Vrfy*: takes key k, message m, and tag t as input; outputs 1 ("accept") or 0 ("reject")

  For all m and all k output by *Gen*,
  $$Vrfy_k(m, Mac_k(m)) = 1$$

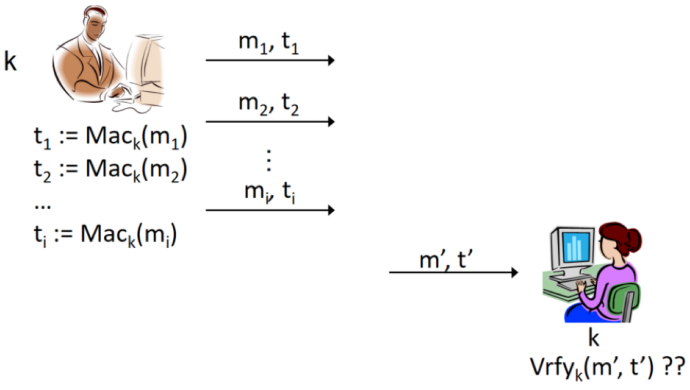
增加一个 tag

Security

- Threat model
  - "Adaptive chosen-message attack"
  - Assume the attacker can induce the sender to authenticate *messages of the attacker's choice*

- Security goal
  - "Existential unforgeability"
  - Attacker should be unable to forge a valid tag on *any* message not previously authenticated by the sender

chosen-message attack 是针对 integrity 的攻击



Formal definition

- Fix A, Π. Define a randomized experiment $Forge_{A,\Pi}(n)$:
  1. $k \leftarrow Gen(1^n)$
  2. $A(1^n)$ interacts with an *oracle* $Mac_k(\cdot)$; let M be the set of messages submitted to this oracle
  3. A outputs (m, t)
  4. A *succeeds*, and the experiment evaluates to 1, if $Vrfy_k(m, t) = 1$ and $m \notin M$

  Definition 6.2 Π is *secure* if for all PPT attackers A, there is a *negligible* function ϵ such that
  $$\Pr[Forge_{A,\Pi}(n) = 1] \leq \epsilon(n)$$

  Note: A CPA-secure encryption scheme is not a secure MAC.

能与 authentication oracle 交互

m 需要有效且不曾被 A 向 oracle 询问过

CPA 的加密机制不能做为安全的 MAC.
(即用 CPA 的密文作为 tag)

$(m, \langle r, F(r) \oplus m \rangle)$

- Is the definition too strong?
  - We don't want to make any assumptions about what the sender might authenticate
  - We don't want to make any assumptions about what forgeries are "meaningful"

- A *MAC* satisfying this definition can be used anywhere integrity is needed

Replay attacks 重放攻击

- Replay attacks are not prevented
  - No *stateless* mechanism can prevent them
- Replay attacks are often a significant real-world concern
- Need to protect against replay attacks at a higher level
  - Decision about what to do with a replayed message is application-dependent

block-reordering attack:

观察到 $\langle t_1, m_1\rangle$, $\langle t_2, m_2\rangle$

可伪造 $\langle t_1, m_2\rangle$, $\langle t_2, m_1\rangle$

防止：$i \| m_i$ (消息前加上序号)

A fixed-length MAC

- Intuition: we need a keyed function *Mac* such that:
  - Given $Mac_k(m_1)$, $Mac_k(m_2)$, ...,
  - It is infeasible to predict the value $Mac_k(m)$ for any $m \notin \{m_1, m_2, \ldots\}$
- Let *Mac* be a PRF!

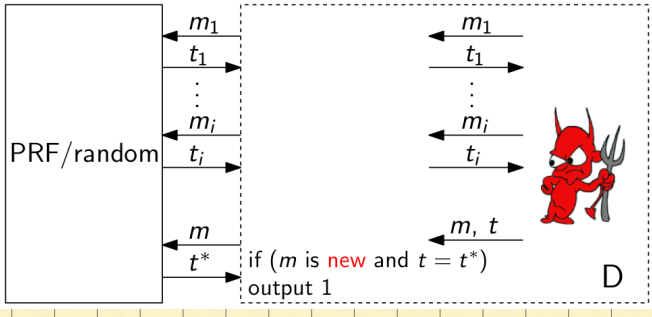
Construction

- Let F be a length-preserving PRF (aka block cipher)
- Construct the following *MAC* Π:
  - *Gen*: choose a uniform key k for F
  - $Mac_k(m)$: output $F_k(m)$
  - $Vrfy_k(m, t)$: output 1 iff $F_k(m) = t$
- **Theorem 6.3** Π is a *secure* MAC

Theorem 4.6 in Textbook

Proof by reduction

- **Theorem 6.3** Π is a *secure* MAC



t_i 可能来自是PRF也可能是random function，看完多个pair后 attacker 能伪造一个pair $\langle m, t\rangle$ (m未曾问过) 能pass 验证

Analysis

- When D interacts with F_k for uniform k, the view of the adversary is *identical* to its view in the real MAC experiment
  - $\Pr[D^{F_k} \text{ outputs } 1] = \Pr[Forge_{Adv,\Pi}(n) = 1]$
- When D interacts with uniform f, then seeing $f(m_1), \ldots, f(m_i)$ does not help predict $f(m)$ for any $m \notin \{m_1, \ldots, m_i\}$
  - $\Pr[D^f \text{ outputs } 1] \leq 2^{-n}$
- Since F is a *PRF*, $|\Pr[D^{F_k} \text{ outputs } 1] - \Pr[D^f \text{ outputs } 1]| < negl(n)$
  $\Rightarrow \Pr[Forge_{Adv,\Pi}(n) = 1] = \Pr[D^{F_k} \text{ outputs } 1] \leq 2^{-n} + negl(n)$

Drawbacks

- This only works for *fixed-length* messages
- This only works for *short* messages
  - E.g., AES has a 128-bit block size (shorter than a tweet!)
- So, the previous construction is limited to authenticating *short, fixed-length* messages
- One natural idea:
  - $Mac'_k(m_1, \ldots, m_\ell) = Mac_k(m_1), \ldots, Mac_k(m_\ell)$
  - $Vrfy'_k(m_1, \ldots, m_\ell, t_1, \ldots, t_\ell) = 1$ iff $Vrfy_k(m_i, t_i) = 1$ for all i
  - Is this secure?

只能对有限长的短信息

即相当于ECB加密模式，并不安全

Construction

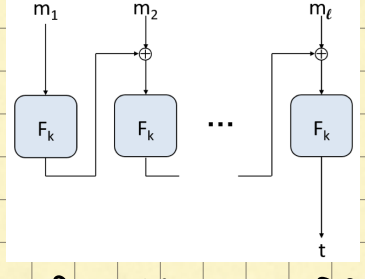
- Need to prevent (at least)
  - Block reordering
  - Truncation
  - "Mixing-and-matching" blocks from multiple messages
- One solution:
  - $Mac'_k(m_1, \ldots, m_\ell) = r, Mac_k(r||\ell||1||m_1), \ldots, Mac_k(r||\ell||\ell||m_\ell)$
  - See Construction 4.7 & Theorem 4.8
  - Not very efficient. Can we do better?

"随机数" ↑ →顺序编号 ↓ 长度

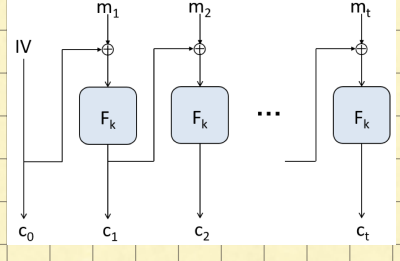
收到 (m_1, m_2, t_1, t_2) (m'_1, m'_2, t'_1, t'_2)
Block reordering : (m_2, m_1, t_2, t_1) (加入顺序信息可避免)
Truncation : (m_1, t_1) (加入长度信息)
Mixing-and-matching : (m_1, m'_2, t_1, t'_2) (加入随机数)

此时安全了，但是效率不高

(Basic) CBC-MAC



CBC mode



CBC-MAC vs. CBC-mode

- CBC-MAC is *deterministic* (no IV)
  - MACs do not need to be randomized to be secure
  - Verification is done by re-computing the result
- In CBC-MAC, *only the final value* is output
- Both are essential for security

MAC无需随机性来保证安全

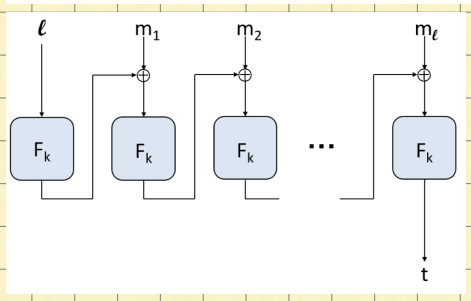
CBC-MAC只要最终的值作为输出

Security of (Basic) CBC-MAC

- If F is a PRF with block length n, then for any fixed ℓ basic CBC-MAC is a secure MAC for messages of length $\ell \cdot n$
- The sender and receiver must agree on the length parameters ℓ in advance
  - Basic CBC-MAC is not secure if this is not done! (Attacks?)
- Several ways to handle variable-length messages
  - One of the simplest: *prepend* the message length before applying (basic) CBC-MAC

若ℓ变化，则不安全。如
Attack可问谕言机 $MAC(\cdot)$：m_1 → $F_k(m_1)$；$F_k(m_1)\oplus m_2$ → $F_k(F_k(m_1)\oplus m_2)$
那么可获得 (m_1, m_2) 的tag：$F_k(F_k(m_1)\oplus m_2)$

后接是不安全的



Authenticated encryption (secrecy + integrity)

- We have shown primitives for achieving *secrecy* and *integrity* in the private-key setting
  What if we want to achieve both?
- Secrecy notion: CCA-security
- Integrity notion: unforgeability
  - Adversary cannot generate ciphertext that decrypts to a previously unencrypted message

认证加密需保证 CCA-security 和 unforgeability

Constructions

- There are three natural generic constructions:
  - Encrypt and Authenticate (E&A): Compute $c = Enc_{k_1}(m)$ and $t = Mac_{k_2}(m)$ and send (c, t) (SSH style)
  - Authenticate and then Encrypt (AtE): Compute $t = Mac_{k_2}(m)$ and then $Enc_{k_1}(t)$ (SSL style)
  - Encrypt and then Authentication (EtA): Compute $c = Enc_{k_1}(m)$ and $t = Mac_{k_1}(c)$ and send (c, t) (IPSec style)

  Note: In all these methods, we use independent keys (k_1, k_2) for encryption and authentication

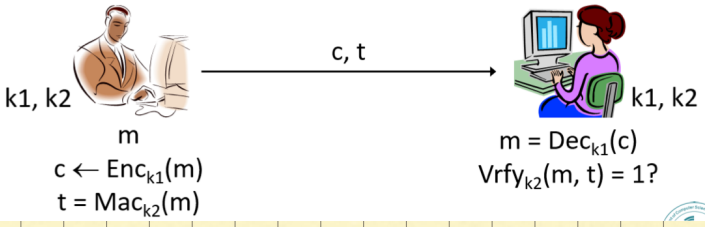
} 存在不安全

安全

encryption 和 authentication 用的 key 不一样。
(如果一样也不安全)

Generic constructions

- Generically combine an *encryption scheme* and a *MAC*
- **Goal**: the combination should be an authenticated encryption scheme when instantiated with any *CPA-secure* encryption scheme and any *secure* MAC
- Encrypt and authenticate (E&A)



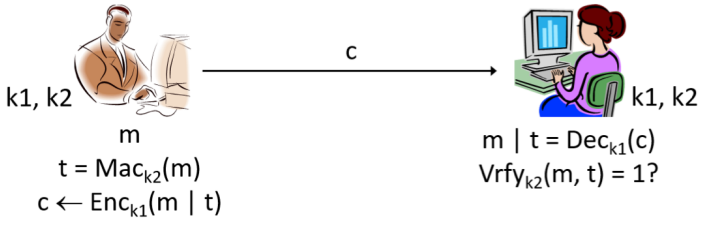
Problems

- The *tag* t might leak information about m!
  - Nothing in the definition of security for a MAC implies that it hides information about m
  - So, the combination may not even be *EAV-secure*
- If the MAC is deterministic (as is CBC-MAC), then the tag leaks whether the same message is encrypted twice
  - I.e., the combination will not be *CPA-secure*

tag可能会泄露明文信息

若MAC是确定性的，tag会泄露同一明文是否被加密了两次.

Authenticated then encrypt (AtE)



- Problems
  - Padding-oracle attack still works
  - Other counterexamples are also possible
  - The combination may not be *CCA-secure*

AtE is not secure in general

- Idea: consider the *CPA-secure* entryption scheme in **Theorem 5.1**, if combined with every secure MAC in the form of AtE, by proving it is malleable, we show that it is not *CCA-secure*.

Gen(1^n): choose a uniform key $k \in \{0,1\}^n$
Enc$_k(m)$, for $|m| = |k|$
  - Choose uniform $r \in \{0,1\}^n$ (*nonce/ initialization vector*)
  - Output ciphertext $\langle r, F_k(r) \oplus m \rangle$

Dec$_k(c_1, c_2)$: output $c_2 \oplus F_k(c_1)$

Theorem 5.1 If F is a pseudorandom function, then this scheme is *CPA-secure*.

- Given such a CPA-secure encryption schemed (Gen, Enc, Dec), we build a new encryption scheme (Gen', Enc', Dec'):

  For an n-bit plaintext m, first apply an encoding of m into a $2n$-bit string m' by representing each bit m_i, $i = 1, \ldots, n$, in m with two bits in m':
  1. if bit $m_i = 0$, then $(m'_{2i-1}, m'_{2i}) = (0,0)$
  2. if bit $m_i = 1$, then $(m'_{2i-1}, m'_{2i}) = (0,1)$ or $(1,0)$

  The encryption Enc is then applied to m'.

  For decrypting $c = Enc'_k(m')$, one first applies the decryption Dec to obtain m', which is then decoded into m by mapping
  $(0,0) \mapsto 0$, $(0,1)$ or $(1,0) \mapsto 1$
  If m' contains a pair $(m'_{2i-1}, m'_{2i}) = (1,1)$, the decoding outputs the invalidity sign $\perp$.

- We consider an active attack:
  When an attacker Eve sees a transmitted ciphertext $c = Enc'_k(m)$, she can learn the first bit m_1 of m as follows:
  She intercepts c, flips the first two bits (c_1, c_2) of c, and sends the modified ciphertext c' to its destination. If she can obtain the information of whether the decryption output a valid or invalid plaintext then Eve learns the first bit of m.
  This is so since the modified c' is valid if and only if $m_1 = 1$.
  Recall that AtE means compute $t = Mac_{k_2}(m)$ and then send $Enc_{k_1}(m||t)$. The MAC is applied to the data before encoding and encryption.
  If the original bit is 1, the change in ciphertext will result in the same decrypted plaintext and then the MAC check will succeed.
  In a sense, the MAC just makes things worse since a failure of authentication is easier to be learnt by the attacker.

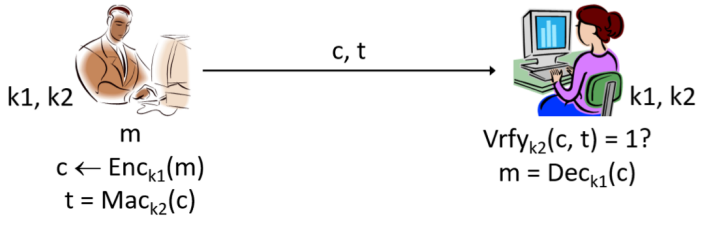
- Instead of this CPA-secure encryption scheme, even we use a perfect secure one-time pad, AtE is not generally secure.

  Note:
  1. The application of an encoding to a plaintext before encryption is used commonly for padding and other purposes.
  2. Encoding of this type can be motivated by stronger security requirements: e.g., to prevent an attacker from learning the exact length of transmitted messages or other traffic analysis information.

  If one wants to **claim** the security of AtE, one needs to analyze the combination as a whole or use stronger or specific properties of the encryption function.

  Read the paper [Krawczyk 2001] & the textbook
  Note: This does not mean that SSL is not secure, but does mean that it is not *generically secure*.

Encrypt then authenticate (EtA)



- Security
  - If the encryption scheme is *CPA-secure* and the MAC is *secure* (with unique tags), then this is an *authenticated encryption scheme*
  - It achieves something even stronger: Given ciphertexts corresponding to (chosen) plaintexts $m_1, \ldots, m_k$, it is infeasible for an attacker to generate any new, valid ciphertext!

CPA + MAC = CCA (实际上是更强的AE)

攻击者无法生成任何新的有效密文.

令 private-key encryption scheme $\Pi_E=(Enc, Dec)$, message authentication code $\Pi_M=(Mac, Vrfy)$
key 生成是独立的：$k_E, k_M \leftarrow_R \{0,1\}^n$. 定义新 scheme $\Pi=(Gen', Enc', Dec')$

- Gen'：输入 1^n，独立、均等选择：$k_E, k_M \leftarrow_R \{0,1\}^n$. 输出 (k_E, k_M)
- Enc'：输入 key (k_E, k_M) 及明文 m，计算 $c \leftarrow Enc_{k_E}(m)$ 和 $t \leftarrow Mac_{k_M}(c)$，输出密文 $\langle c,t\rangle$.
- Dec'：输入 key (k_E, k_M) 及密文 $\langle c,t\rangle$. 若 $Vrfy(c,t)=1$，输出 $Dec_{k_E}(c)$；否则输出 $\perp$ (error)

以上这个 scheme 是 CCA-secure 的（或更强的 AE）

证明：设 PPT 攻击者 A 攻击 CCA scheme. 令事件 ValidQuery 为 A 向其 decryption oracle 提交新的（即 A 事先未从 encryption oracle 或 challenge ciphertext 中获得）、有效的 ciphertext.

① $Pr[ValidQuery] \le negl(n)$

事件 ValidQuery 发生 $\Rightarrow$ 在 MAC 实验中攻击者成功伪造了新的有效 (c,t).

令 $q(\cdot)$ 为 A 访问 decryption oracle 次数的多项式上界.

对 Π_M 构造 A_M，并使 A 为其 subroutine：

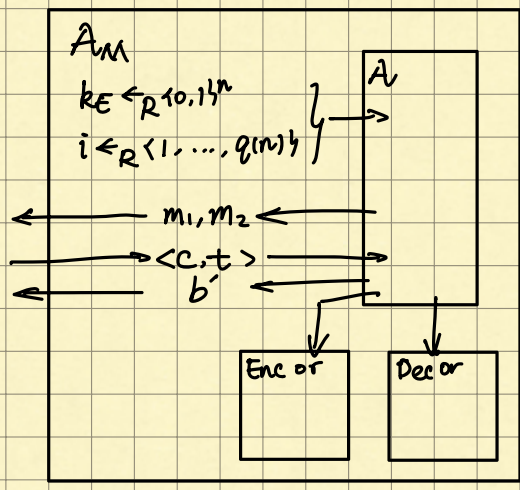
A_M 接收输入 1^n. 能访问 MAC oracle $Mac_{k_M}(\cdot)$.

a. $k_E \leftarrow_R \{0,1\}^n$, $i \leftarrow_R \{1, \dots, q(n)\}$

b. 输入 1^n 至 A. 若

A 访问 encryption oracle（对明文 m）：i) $c \leftarrow Enc_{k_E}(m)$
ii) 将 c 给 A_M 并访问 MAC oracle $t \leftarrow Mac_{k_M}(c)$，返还 $\langle c,t\rangle$ 给 A.

A 访问 decryption oracle（对密文 $\langle c,t\rangle$）：若 $\langle c,t\rangle$ 是第 i 次 decryption oracle 访问，则输出 (c,t) 并停止. (*)
否则：若 $\langle c,t\rangle$ 的明文 m 曾在 A 访问 encryption oracle 时出现，则返回 m
否则返回 $\perp$ (error)



注(*)：因为 A_M 猜测 A 第 i 次访问 decryption oracle 是第一次新的有效的访问.

上述实验中的 $Pr[A \text{ runs as subroutine of } A_M] = Pr[PrivK^{cca}_{A,\Pi}(n)=1 \wedge \overline{ValidQuery}]$

A 的 encryption oracle 访问由 A 的 MAC oracle 最终返回

decryption oracle 访问中的 ii) 返回 $\perp$ 是因为此时的 $\langle c,t\rangle$ 已经是新的了，而 A 的 decryption oracle 实际上只能解 c，解不了 $\langle c,t\rangle$.

如果 A_M 恰好猜中 i（即 A 第 i 次访问 decryption oracle 恰是 ValidQuery 事件第一次发生），则 A_M 会输出 (c,t)，而不返回给 A，该 pair 的输出也表示 A_M 赢得了实验 $Mac\text{-}forge_{A_M,\Pi_M}(n)$.

$$Pr[Mac\text{-}forge_{A_M,\Pi_M}(n)=1] \ge Pr[ValidQuery] \cdot Pr[A_M \text{ 成功猜中 } i] = Pr[ValidQuery]/q(n)$$

因为 Π_M 是安全 MAC 且 $q(n)$ 是多项式，故 $Pr[ValidQuery] \le negl(n)$

② Π 是 CCA-secure

设 A 为 Π 的攻击者.

$$Pr[PrivK^{cca}_{A,\Pi}(n)=1] = Pr[PrivK^{cca}_{A,\Pi}(n)=1 \mid ValidQuery]\,Pr[ValidQuery] + Pr[PrivK^{cca}_{A,\Pi}(n)=1 \wedge \overline{ValidQuery}]$$
$$\le Pr[ValidQuery] + Pr[PrivK^{cca}_{A,\Pi}(n)=1 \wedge \overline{ValidQuery}]$$

因为 $Pr[ValidQuery]$ 是 negligible 的，仅需证 $Pr[PrivK^{cca}_{A,\Pi}(n)=1 \wedge \overline{ValidQuery}] \le \frac{1}{2} + negl(n)$

对 Π_E 构造 A_E，并使 A 为其 subroutine：

A_E 接收输入 1^n，能访问 $Enc_{k_E}(\cdot)$。（不需要 $Dec_{k_E}(\cdot)$）

a. $k_E \leftarrow_R \{0,1\}^n$

b. 输入 1^n 至 A，若

A 访问 encryption oracle（对明文 m）：i) 访问 A_E 的 $Enc_{k_E}(\cdot)$ 并得到密文 C。

ii) $t \leftarrow Mac_{k_M}(c)$，返还 $\langle c,t\rangle$ 给 A。

A 访问 decryption oracle（对密文 $\langle c,t\rangle$）：若 $\langle c,t\rangle$ 的明文 m 曾在 A 访问 encryption oracle 时出现，则返回 m

否则返回 $\perp$ (error)（不需要 $Dec_{k_E}(\cdot)$ 的原因）

c. A 输出明文对 (m_0,m_1) 时，A_E 直接转发给 challenger。

challenger 返还密文 C 后，计算 $t \leftarrow Mac_{k_M}(c)$ 并返还 $\langle c,t\rangle$ 给 A

A 可以继续访问两个 oracle。

d. A_E 输出 A 输出的 b'。

上述实验中的 $Pr[A \text{ runs as subroutine of } A_E] = Pr[PrivK^{cca}_{A,\Pi}(n)=1 \wedge \overline{ValidQuery}]$

所以有：

$$Pr[PrivK^{cpa}_{A_E,\Pi_E}(n)=1] \geq Pr[PrivK^{cpa}_{A_E,\Pi_E}(n)=1 \wedge \overline{ValidQuery}]$$

$$= Pr[PrivK^{cca}_{A,\Pi}(n)=1 \wedge \overline{ValidQuery}]$$

因为 Π_E 是 CPA-secure 的，故 $Pr[PrivK^{cpa}_{A_E,\Pi_E}(n)=1] \leq \frac{1}{2} + negl(n)$

所以 $Pr[PrivK^{cca}_{A,\Pi}(n)=1 \wedge \overline{ValidQuery}] \leq \frac{1}{2} + negl(n)$

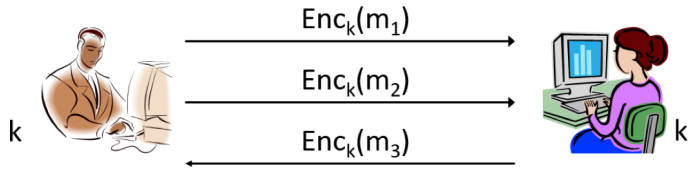
综上，Π 是 CCA-secure 的。

- Encrypt-then-authenticate (with independent keys) is the recommended generic approach for constructing *authenticated encryption*
- Other, more efficient constructions have been proposed and are an active area of research and standardization

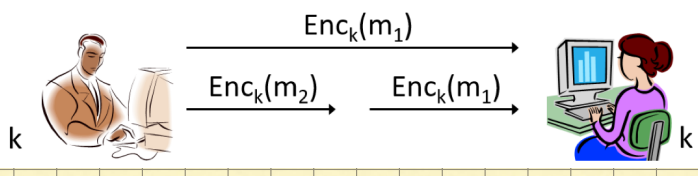
  https://competitions.cr.yp.to/caesar.html

Secure sessions

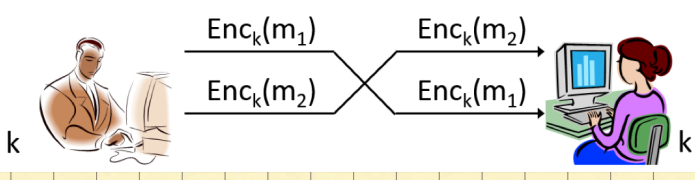
- Consider parties who wish to communicate securely over the course of a session
  - "*Securely*" = secrecy and integrity
  - "*Session*" = period of time over which the parties are willing to maintain state
- Can use *authenticated encryption*



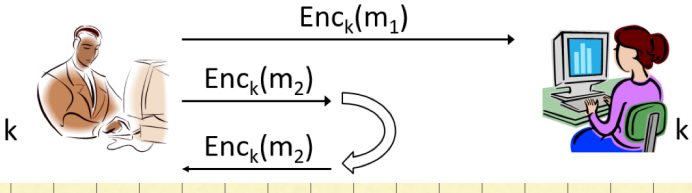
Replay attack



Re-ordering attack



Reflection attack



Secure sessions

- These attacks (and many others) can be prevented using *counters/sequence numbers* and *identifiers*
  - Can also use a *directionality bit* in place of identifiers