CS 6501 Week 2: Symmetric Cryptography

Goal of secure communication:

Alice $\xleftrightarrow{\text{(untrusted network)}}$ Bob

$\uparrow$

Eve

How do Alice and Bob communicate over an <u>untrusted</u> network?

Properties we might care about:
- Message confidentiality: Eve should not learn messages
- Message integrity: Eve should not be able to modify messages

<u>This week</u>: focus on <u>symmetric</u> cryptography (e.g., assume Alice and Bob have a <u>shared</u> key)

(also called a <u>cipher</u>) — we hide the (implicit) dependence on the security parameter

<u>Definition</u>. A symmetric encryption scheme defined over a key-space $\mathcal{K}$, message space $\mathcal{M}$ and ciphertext space $\mathcal{C}$ is a tuple of efficient algorithms (KeyGen, Encrypt, Decrypt) with the following properties:
- $\text{KeyGen}(1^\lambda) \to k$: On input the security parameter λ, outputs a key $k \in \mathcal{K}$
- $\text{Encrypt}(k, m) \to c$: On input a key $k \in \mathcal{K}$ and a message $m \in \mathcal{M}$, output a ciphertext $c \in \mathcal{C}$

} can be <u>randomized</u> algorithms

- $\text{Decrypt}(k, c) \to m$: On input a key $k \in \mathcal{K}$ and a ciphertext $c \in \mathcal{C}$, output a message $m \in \mathcal{M}$

} typically a <u>deterministic</u> algorithm

<u>Correctness</u>. for all $\lambda \in \mathbb{N}$ and $m \in \mathcal{M}$:

$$\Pr[k \leftarrow \text{KeyGen}(1^\lambda) : \text{Decrypt}(k, \text{Encrypt}(k, m)) = m] = 1.$$

"Decryption recovers the message."

can allow for $\text{negl}(\lambda)$ failure probability (i.e., this relation holds with prob $1 - \text{negl}(\lambda)$)

↳ called "statistical correctness" instead of "perfect correctness"

"one-wayness" is not enough: leaking partial information can be problematic

<u>How to define security</u>? What is the property that we want? Eve learns <u>nothing</u> about the message from its ciphertext.

<u>Perfect security</u>. for all $\lambda \in \mathbb{N}$ and $m_0, m_1 \in \mathcal{M}$ and $c \in \mathcal{C}$:

$$\Pr[k \leftarrow \text{KeyGen}(1^\lambda) : \text{Encrypt}(k, m_0) = c] = \Pr[k \leftarrow \text{KeyGen}(1^\lambda) : \text{Encrypt}(k, m_1) = c]$$

given c, adversary learns <u>nothing</u> about underlying message (no matter how powerful the adversary is!)

A cipher with perfect security: the one-time pad (OTP):

$\mathcal{K} = \{0,1\}^\lambda$	$\text{KeyGen}(1^\lambda)$: output $k \xleftarrow{R} \mathcal{K}$
$\mathcal{M} = \{0,1\}^\lambda$	$\text{Encrypt}(k, m)$: output $k \oplus m$
$\mathcal{C} = \{0,1\}^\lambda$	$\text{Decrypt}(k, c)$: output $k \oplus c$

bitwise xor operation (addition modulo 2)

<u>Correctness.</u> Take any $k \in \{0,1\}^\lambda$. Then for any $m \in \{0,1\}^\lambda$:

$$\text{Decrypt}(k, \text{Encrypt}(k, m)) = k \oplus (k \oplus m) = m$$

<u>Perfect Secrecy.</u> Take any $m \in \{0,1\}^\lambda$ and $c \in \{0,1\}^\lambda$. Then,

$$\Pr[k \leftarrow \text{KeyGen}(1^\lambda) : \text{Encrypt}(k, m) = c] = \Pr[k \xleftarrow{R} \{0,1\}^\lambda : k \oplus m = c]$$
$$= \Pr[k \xleftarrow{R} \{0,1\}^\lambda : k = m \oplus c] = \frac{1}{2^\lambda}.$$

OTP is very simple (just computing xors) and provides <u>perfect secrecy</u>, so are we done?

NO! The keys in a OTP are as long as the message itself.

though key can be shared in advance

↳ If we had a mechanism to share the key securely, can just share the message instead.

Theorem (Shannon). If an encryption scheme with key-space $\mathcal{K}$ and message space $\mathcal{M}$ satisfies perfect secrecy, then $|\mathcal{K}| \geq |\mathcal{M}|$

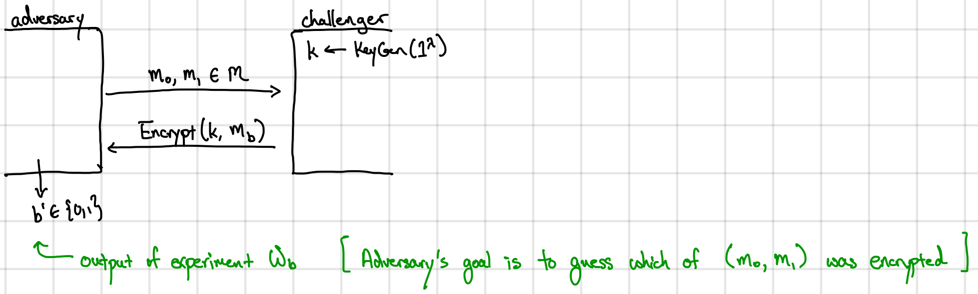
Proof (Sketch). Follows by a "counting argument". Suppose $|\mathcal{K}| < |\mathcal{M}|$. Let $c = \text{Encrypt}(k, m)$ for some $k \in \mathcal{K}$ and $m \in \mathcal{M}$. Ciphertext c can decrypt to at most $|\mathcal{K}| < |\mathcal{M}|$ possible messages, so there exist $m' \in \mathcal{M}$ such that $\text{Encrypt}(k, m') \neq c$ for all $k \in \mathcal{K}$ (by correctness).

What if we want short keys? Have to settle for weaker security. Compromise: require security against computationally-bounded adversaries.

Semantic security. An encryption scheme (KeyGen, Encrypt, Decrypt) is semantically secure if for all efficient adversaries $\mathcal{A}$,

$$\text{SSAdv}[\mathcal{A}] = \left|\Pr[W_0 = 1] - \Pr[W_1 = 1]\right| = \text{negl}(\lambda)$$

where we define W_b (for $b \in \{0,1\}$) to be the output of the semantic security game:



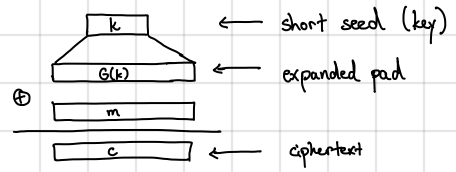
Stream ciphers: We can construct semantically-secure encryption schemes from PRGs. This is often called a "stream cipher."

Let $G : \{0,1\}^\lambda \to \{0,1\}^\ell$ be a PRG where $\ell \gg \lambda$. We define the stream cipher as follows:

$\mathcal{K} = \{0,1\}^\lambda$ — $\text{KeyGen}(1^\lambda)$: output $k \xleftarrow{R} \{0,1\}^\lambda$

$\mathcal{M} = \{0,1\}^\ell$ — $\text{Encrypt}(k, m)$: output $c \leftarrow G(k) \oplus m$

$\mathcal{C} = \{0,1\}^\ell$ — $\text{Decrypt}(k, c)$: output $m \leftarrow G(k) \oplus c$



Semantic security: (Hybrid argument).

$$\begin{aligned}
\{k \xleftarrow{R} \{0,1\}^\lambda : G(k) \oplus m_0\} &\overset{c}{\approx} \{r \xleftarrow{R} \{0,1\}^\ell : r \oplus m_0\} && \text{PRG security} \\
&\equiv \{r \xleftarrow{R} \{0,1\}^\ell : r \oplus m_1\} && \rangle\ \text{OTP} \\
&\overset{c}{\approx} \{k \xleftarrow{R} \{0,1\}^\lambda : G(k) \oplus m_1\} && \rangle\ \text{PRG security}
\end{aligned}$$

Summary: A PRG can be used to "implement" (or "compress") a OTP in the setting of computationally-bounded adversaries

Why "one-time" pad? What happens if we reuse the same key (pad) to encrypt multiple messages?

Suppose
$$c_0 = G(k) \oplus m_0, \quad c_1 = G(k) \oplus m_1 \implies c_0 \oplus c_1 = [G(k) \oplus m_0] \oplus [G(k) \oplus m_1] = m_0 \oplus m_1 \Leftarrow$$
leaks information about the underlying message!

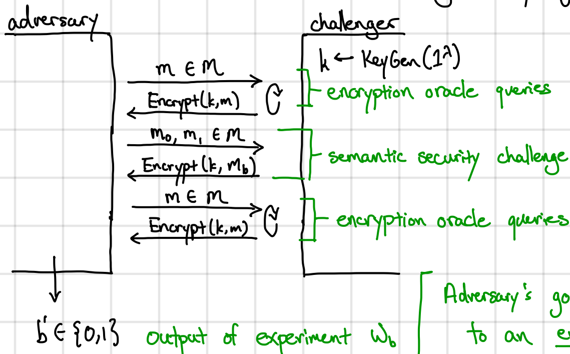
The "two-time pad" is completely insecure! Never use a stream cipher to encrypt multiple messages!

Notice: our security model does not capture this attack. Adversary only gets to see encryption of a single message. To capture reusability, we need a stronger security definition that allows the adversary to see multiple ciphertexts.

CPA security: An encryption scheme (KeyGen, Encrypt, Decrypt) is secure against chosen plaintext attacks (CPA-secure) if for all efficient adversaries $\mathcal{A}$,

$$\text{CPAAdv}[\mathcal{A}] = \left| \Pr[W_0 = 1] - \Pr[W_1 = 1] \right| = \text{negl}(\lambda)$$

where W_b ($b \in \{0,1\}$) is the output of the following security game



[Adversary's goal is to guess which of m_0 or m_1 was encrypted, given access to an encryption oracle (i.e., adversary gets to see encryptions of messages of its choice.]

CPA security captures a conservative notion of "multi-message" security: even if adversary gets to choose the message that is encrypted, it cannot break semantic security.

→ standard notion of message confidentiality

Implication: CPA-secure encryption schemes must be randomized! OTP and stream ciphers are not CPA-secure.

ciphertexts must be longer than plaintexts.

CPA-secure encryption from PRFs: Let $F: \{0,1\}^\lambda \times \{0,1\}^\lambda \to \{0,1\}^\ell$ be a PRF. We define the cipher as follows:

$\mathcal{K} = \{0,1\}^\lambda$

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

$\mathcal{C} = \{0,1\}^{\lambda+\ell}$

$\text{KeyGen}(1^\lambda)$: output $k \xleftarrow{R} \{0,1\}^\lambda$

$\text{Encrypt}(k, m)$: sample $r \xleftarrow{R} \{0,1\}^\lambda$

compute $z \leftarrow F(k, r) \oplus m$ ← use PRF to derive a (different) random pad for each encryption query

output $c = (r, z)$

$\text{Decrypt}(k, c)$: parse $c = (r, z)$ and output $m \leftarrow F(k, r) \oplus z$

CPA-security: Proceed via hybrid argument:

Hyb_0: Real game where challenger encrypts m_0

Hyb_1: Hyb_0 except challenger uses a truly random function $f(\cdot)$ in place of $F(k, \cdot)$

Hyb_2: Hyb_1 except challenger encrypts m_1

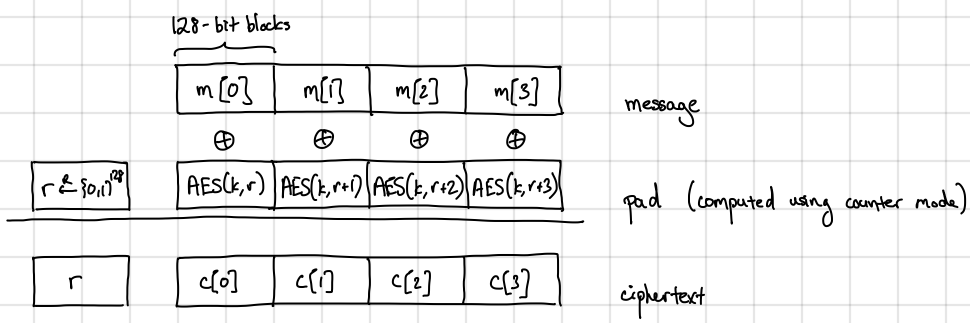
Hyb_3: Real game where challenger encrypts m_1

} PRF security

} Statistically indistinguishable as long as r^* used to encrypt challenge ciphertext never appears in encryption query (w.p. $\frac{\text{poly}(\lambda)}{2^\lambda} = \text{negl}$)

} PRF security

In practice, we have block ciphers (PRPs) with fixed block sizes. For example, AES: $\{0,1\}^{128} \times \{0,1\}^{128} \to \{0,1\}^{128}$ is a commonly used block cipher with 128-bit blocks. To encrypt messages longer than 128-bits, we use "randomized counter mode":



Security: As long as no "collision" (repeated block), secure assuming AES is a PRP [thus, can use same key to encrypt $\sim 2^{64}$ blocks]

__Message integrity__: Confidentiality alone not sufficient, also need __message integrity__. Otherwise adversary can tamper with the message (e.g., "Send \$100 to Bob" $\rightarrow$ "Send \$100 to Eve")

__Idea__: Append a "tag" (also called a "signature") to the message to prove integrity (property we want is tags should be hard to forge)

__Message authentication codes (MACs)__: A message authentication code with key-space $\mathcal{K}$, message space $\mathcal{M}$, and tag space $\mathcal{T}$ is a tuple of three algorithms (KeyGen, Sign, Verify) with the following properties:

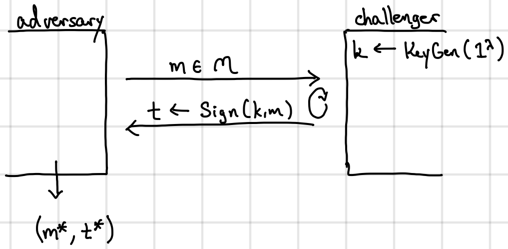
- $\mathrm{KeyGen}(1^\lambda) \rightarrow k$: On input the security parameter λ, outputs a key $k \in \mathcal{K}$
- $\mathrm{Sign}(k, m) \rightarrow t$: On input a key $k \in \mathcal{K}$ and a message $m \in \mathcal{M}$, outputs a tag $t \in \mathcal{T}$
- $\mathrm{Verify}(k, m, t) \rightarrow \{0,1\}$: On input a key $k \in \mathcal{K}$, a message $m \in \mathcal{M}$, and a tag $t \in \mathcal{T}$, output $b \in \{0,1\}$

__Correctness__: for all $\lambda \in \mathbb{N}$ and all messages $m \in \mathcal{M}$,

$$\Pr[k \leftarrow \mathrm{KeyGen}(1^\lambda) : \mathrm{Verify}(k, m, \mathrm{Sign}(k,m)) = 1] = 1.$$

adversary gets to choose messages to be signed

__Unforgeability__ : A MAC (KeyGen, Sign, Verify) satisfies existential unforgeability against $\underbrace{\text{chosen message}}$ attacks (EUF-CMA) if for all efficient adversaries $\mathcal{A}$, $\mathrm{MACAdv}[\mathcal{A}] = \Pr[W = 1] = \mathrm{negl}(\lambda)$ where W is the output of the following security game:



Let $m_1, \ldots, m_Q$ be the signing queries the adversary submits to the challenger, and let $t_i \leftarrow \mathrm{Sign}(k, m_i)$ be the challenger's responses. Then, $W = 1$ if and only if:

$$\mathrm{Verify}(k, m^*, t^*) = 1 \quad \text{and} \quad (m^*, t^*) \notin \{(m_1, t_1), \ldots, (m_Q, t_Q)\}$$

MAC security notion says that adversary cannot produce a __new__ tag on any message even if it gets to obtain tags on messages of its choosing

__MACs from PRFs__: Let $F : \mathcal{K} \times \mathcal{M} \rightarrow \mathcal{T}$ be a PRF. We construct a MAC as follows:

$\mathrm{KeyGen}(1^\lambda)$: output $k \xleftarrow{R} \mathcal{K}$

$\mathrm{Sign}(k, m)$: output $t \leftarrow F(k, m)$

$\mathrm{Verify}(k, m, t)$: output 1 if $t = F(k,m)$ and 0 otherwise

__Unforgeability__ : Use a hybrid argument:

Hyb_0 : Real EUF-CMA game

Hyb_1 : Hyb_0 except the challenger computes $f(\cdot)$ in place of $F(k, \cdot)$ where $f \xleftarrow{R} \mathrm{Funs}[\mathcal{M}, \mathcal{T}]$

In particular, we can show that for any efficient algorithm $\mathcal{A}$ that breaks security of the MAC, there exists an efficient adversary $\mathcal{B}$ for the PRF such that

$$\mathrm{MACAdv}[\mathcal{A}] \leq \mathrm{PRFAdv}[\mathcal{B}, F] + \frac{1}{|\mathcal{T}|}$$

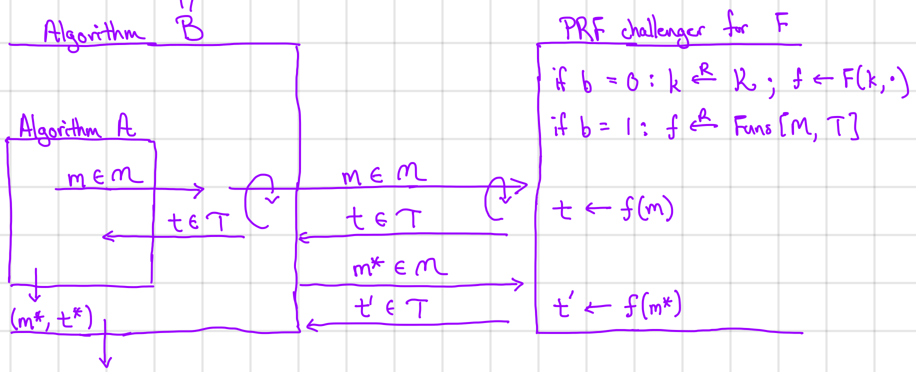
Thus, if F is a secure PRF and $1/|\mathcal{T}|$ is negligible (i.e., the size of the tag space is superpolynomial in the security parameter), this construction is a secure MAC. We give a formal proof for reference on the next page.

Proof of Unforgeability. For an adversary A, let $\text{Hyb}_i(A)$ denote the output of A interacting according to the specification of Hyb_i. By construction, $\text{MACAdv}(A) = \Pr[\text{Hyb}_0(A) = 1]$. We now show the following:

Lemma. For all efficient adversaries A, there exists an efficient adversary B such that

$$\left|\Pr[\text{Hyb}_0(A) = 1] - \Pr[\text{Hyb}_1(A) = 1]\right| = \text{PRFAdv}[B, F]$$

Proof. Let A be an adversary for the MAC security game. We use A to construct an adversary B for the PRF security game. Algorithm B will use a copy of A as follows:



Output 1 if (m^*, t^*) did not appear in any signing query and $\underbrace{t^* = t'}$

the overall criterion is the one used in Hyb_0 or Hyb_1

this is precisely checking $\text{Verify}(k, x, t^*) = 1$ in Hyb_0 and Hyb_1

Suppose $b = 0$. Then B perfectly simulates experiment Hyb_0 for A. In this case, $\Pr[B \text{ outputs } 1 \mid b = 0] = \Pr[\text{Hyb}_0(A) = 1]$.
Suppose $b = 1$. Then B perfectly simulates experiment Hyb_1 for A. In this case, $\Pr[B \text{ outputs } 1 \mid b = 1] = \Pr[\text{Hyb}_1(A) = 1]$.
Thus, we have that

$$\begin{aligned} \text{PRFAdv}[B, F] &= \left|\Pr[B \text{ outputs } 1 \mid b = 0] - \Pr[B \text{ outputs } 1 \mid b = 1]\right| \\ &= \left|\Pr[\text{Hyb}_0(A) = 1] - \Pr[\text{Hyb}_1(A) = 1]\right| \quad \blacksquare \end{aligned}$$

Lemma. For all adversaries A, $\Pr[\text{Hyb}_1(A) = 1] \le \frac{1}{|T|}$.

Proof. Take any adversary A and consider an execution of $\text{Hyb}_1(A)$. Let $f: M \to T$ be the function sampled by the challenger and (m^*, t^*) be the adversary's challenge. We bound the probability that $\text{Hyb}_1(A) = 1$. We consider two cases:

<u>Case 1:</u> Suppose the adversary previously made a signing query on m^*. Then, $\Pr[\text{Hyb}_1(A) = 1] = 0$ since the first requirement says that $t^* \neq f(m^*)$, in which case $\text{Verify}(k, m^*, t^*) = 0$.

<u>Case 2:</u> Suppose the adversary never makes a signing query on m^*. Then, its view is completely independent of $f(m^*)$ since f is a truly random function. In this case,

$$\begin{aligned} \Pr[\text{Hyb}_1(A) = 1] &= \Pr[f \xleftarrow{R} \text{Funs}[M, T] : f(m^*) = t^*] \\ &= \Pr[t' \xleftarrow{R} T : t' = t^*] = \frac{1}{|T|}. \end{aligned}$$

We conclude that $\Pr[\text{Hyb}_1(A) = 1] \le \frac{1}{|T|}$. $\blacksquare$

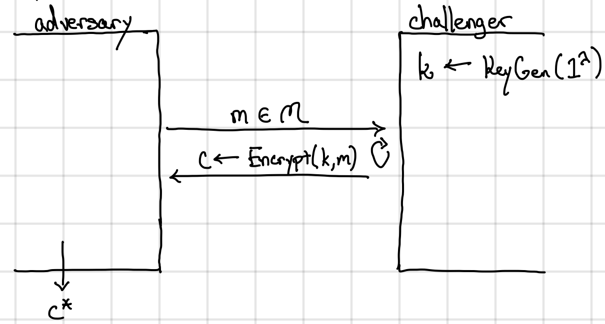
By the lemmas above, we have that for every efficient MAC adversary A, there exists an efficient PRF adversary B such that

$$\begin{aligned} \text{MACAdv}[A] = \Pr[\text{Hyb}_0(A) = 1] &\le \text{PRFAdv}[B, F] + \Pr[\text{Hyb}_1(A) = 1] \\ &\le \text{PRFAdv}[B, F] + \frac{1}{|T|} \quad \blacksquare \end{aligned}$$

Combining confidentiality and integrity: when we use an encryption scheme, we usually want both confidentiality and integrity. This is provided by an authenticated encryption scheme.

Authenticated encryption: An encryption scheme (KeyGen, Encrypt, Decrypt) is an authenticated encryption scheme if it satisfies the following properties:

- CPA security
- Ciphertext integrity: For all efficient adversaries A, $\Pr[W=1] = \mathrm{negl}(\lambda)$ where W is the output of the following experiment:



Let $c_1, \ldots, c_Q$ be the ciphertexts the challenger computes in response to encryption queries. The output $W=1$ if and only if

$$\mathrm{Decrypt}(k, c^*) \neq \perp \quad \text{and} \quad c^* \notin \{c_1, \ldots, c_Q\}$$

Namely, an encryption scheme provides ciphertext integrity if no efficient adversary can come up with a "valid" ciphertext (i.e., a ciphertext that does not decrypt to $\perp$).

Takeaway: Authenticated encryption schemes provide both confidentiality (CPA-security) and integrity (ciphertext integrity).
↳ This is what you should use for symmetric encryption!

Constructing authenticated encryption: "encrypt-then-MAC": Let $(\mathrm{KeyGen}_{SE}, \mathrm{Encrypt}, \mathrm{Decrypt})$ be a CPA-secure encryption scheme with key-space K_{SE}, message space $\mathcal{M}$, and ciphertext space $\mathcal{C}$. Let $(\mathrm{KeyGen}_{MAC}, \mathrm{Sign}, \mathrm{Verify})$ be a MAC with key-space K_{MAC}, message space $\mathcal{C}$ and tag space $\mathcal{T}$.

Authenticated encryption scheme:

$K_{AE} = K_{SE} \times K_{MAC}$

$\mathcal{M}_{AE} = \mathcal{M}$

$\mathcal{C}_{AE} = \mathcal{C} \times \mathcal{T}$

$\mathrm{KeyGen}_{AE}(1^\lambda)$: compute $k_{SE} \leftarrow \mathrm{KeyGen}_{SE}(1^\lambda)$
$k_{MAC} \leftarrow \mathrm{KeyGen}_{MAC}(1^\lambda)$ and
output (k_{SE}, k_{MAC})

$\mathrm{Encrypt}(k, m)$: parse $k = (k_{SE}, k_{MAC})$ and compute
$ct' \leftarrow \mathrm{Encrypt}(k_{SE}, m)$
$t \leftarrow \mathrm{Sign}(k_{MAC}, ct')$ and
output $ct = (ct', t)$

$\mathrm{Decrypt}(k, ct)$: parse $k = (k_{SE}, k_{MAC})$ and $ct = (ct', t)$
if $\mathrm{Verify}(k_{MAC}, ct', t) \neq 1$, output $\perp$
else, output $\mathrm{Decrypt}(k_{SE}, ct')$

Theorem. If (KeyGen_{SE}, Encrypt, Decrypt) is CPA secure and (KeyGen_{MAC}, Sign, Verify) is EUF-CMA secure, then "encrypt-then-MAC" is an authenticated encryption scheme.

block cipher (PRP)

Corollary. PRFs (and thus also, OWFs) $\Longrightarrow$ authenticated encryption (in practice: AES-GCM mode)