Theorem. If Encrypt, Decrypt) is CPA-secure and (Sign, Verify) is a secure MAC, then (Encrypt, Verify) is an authenticated encryption scheme.

Proof (Sketch). CPA-security follows by CPA-security of (Encrypt, Decrypt). Specifically, the MAC is computed on ciphertexts and not the messages. MAC key is independent of encryption key so cannot compromise CPA-security.

Ciphertext integrity follows directly from MAC security (i.e., any valid ciphertext must contain a new tag on some ciphertext that was not given to the adversary by the challenger).

Important notes: Encryption + MAC keys must be independent. Above proof required this (in the formal reduction, need to be able to simulate ciphertexts/MACs - only possible if reduction can change its own key).

- Can also give explicit constructions that are completely broken if same key is used (i.e., both properties fail to hold)
- In general, never reuse cryptographic keys in different schemes; instead, sample fresh, independent keys!

MAC-then-Encrypt: Let (Encrypt, Verify) be a CPA-secure encryption scheme and (Sign, Verify) be a secure MAC. We define MAC-then-Encrypt to be the following scheme:

\[
\begin{align*}
\text{Encrypt}'((k_E, k_M), m) & : \ t \leftarrow \text{Sign}(k_M, m) \\
\text{c} & \leftarrow \text{Encrypt}(k_E, \langle m, t \rangle) \\
\text{output } c
\end{align*}
\]

\[
\begin{align*}
\text{Decrypt}'((k_E, k_M), \langle c, t \rangle) & : \ \text{compute } \langle m, t \rangle \leftarrow \text{Decrypt}(k_E, c) \\
\text{if } \text{Verify}(k_M, m, t) & = 1, \ \text{output } m, \ \text{else, output } \bot
\end{align*}
\]

Not generally secure! SSL 3.0 (precursor to TLS) used randomized CBC + secure MAC

- Simple CCA attack on scheme (by exploiting padding in CBC encryption)

\[
\text{POODLE attack on SSL 3.0 can decrypt all encrypted traffic using a CCA attack}
\]

Padding is a common source of problems with MAC-then-Encrypt systems [see HUSK for an example.]

In the past, libraries provided separate encryption + MAC interfaces — common source of errors

- Good library design for crypt should minimize ways for users to make errors, not provide more flexibility

Today, there are standard block cipher modes of operation that provide authenticated encryption

- One of the most widely used is GCM (Galois counter mode) — standardized by NIST in 2007

GCM mode: follows encrypt-then-MAC paradigm

- CPA-secure encryption + nonce-based counter mode
- MAC is a Carter-Wegman MAC

\[
\text{(AES-GCM provides authenticated encryption)}
\]
Construction based on a Carter-Wegman MAC (built on a one-time MAC)

One-time MAC: analog of one-time pad for integrity
- Information-theoretic security against adversaries that only see one message-tag pair
- Does not need cryptography - can be much faster than cryptographic constructions

- Basic construction: let \( p \) be a large prime (messages will be elements of \( \mathbb{Z}_p \) - integers modulo \( p \))
  KeyGen: sample \( \alpha, \beta \sim \mathbb{Z}_p \) (two random integers in \( \{0, \ldots, p-1\} \)). Key is \( k = (\alpha, \beta) \)
  \( \text{Sign}(k, m) \): output \( \tau = \alpha m + \beta \mod p \)
  \( \text{Verify}(k, m, \tau) \): accept if \( \tau = \alpha m + \beta \mod p \) and reject otherwise

- Security: given \( m, \tau = \alpha m + \beta \), adversary wins only if it outputs \( m' \neq m \) and \( \tau' = \alpha m' + \beta \).
  Since \( \alpha, \beta \sim \mathbb{Z}_p \), we can show that for any choice of \( m \neq m' \) and \( \tau, \tau' \in \mathbb{Z}_p \):

\[
\Pr_{\alpha, \beta \in \mathbb{Z}_p} \left[ \alpha m + \beta = \tau, \alpha m' + \beta = \tau' \right] = \frac{1}{p^2}
\]

\[
\Pr_{\alpha, \beta \in \mathbb{Z}_p} \left[ \alpha m' + \beta = \tau' \mid \alpha m + \beta = \tau \right] = \frac{1}{p} = \text{negl}(\lambda)
\]

in particular, for all \( m \neq m' \) and \( \tau, \tau' \in \mathbb{Z}_p \):

\[
\Pr_{\alpha, \beta \in \mathbb{Z}_p} \left[ \alpha m' + \beta = \tau' \mid \alpha m + \beta = \tau \right] = \frac{1}{p} = \text{negl}(\lambda)
\]

Regardless of adversary's running time.

- For longer messages \( m = m_1 \ldots m_l \) where each \( m_i \in \mathbb{Z}_p \), define the MAC to be:

\[
\tau = m_1 \alpha^l + m_2 \alpha^{l-1} + \ldots + m_l \alpha + \beta \mod p
\]

- Still provides information-theoretic security:
  - for any \( m \neq m' \) of length up to \( l \) and \( \tau, \tau' \in \mathbb{Z}_p \):

\[
\tau = \beta + \sum_{i=1}^{l} m_i \alpha^{i-1} \Rightarrow \sum_{i=1}^{l} (m_i - m'_i) \alpha^{i-1} + (\tau - \tau') = 0
\]

  a polynomial of degree at most \( l \)

\[
\Pr_{\alpha \in \mathbb{Z}_p} \left[ \alpha \text{ is a root of this poly} \right] = \frac{1}{p}
\]

- In practice, the MAC is defined as follows:

\[
\text{Verify}(k, \tau, (r, t)) \text{ output } 1 \text{ if } \text{Verify}(k, F(k_f, r) \oplus t)
\]

- The Carter-Wegman MAC is defined as follows:

\[
\text{Sign}(k, m) \Rightarrow (r \sim \text{Rand}, t \leftarrow \text{Sign}(k, m) \oplus F(k_f, r))
\]

Carter-Wegman MAC ("encrypted MAC") : very lightweight, randomized MAC from the one-time MAC:
- Let (\text{Sign}, \text{Verify}) be a one-time MAC (or more generally, a universal hash function)
- Let \( F: k_f \times R \rightarrow \{0, 1\}^l \) be a PRF

The Carter-Wegman MAC is defined as follows:

\[
\text{Sign}(k, m) \Rightarrow (r \sim \text{Rand}, t \leftarrow \text{Sign}(k, m) \oplus F(k_f, r))
\]

\[
\text{Verify}(k, r, t) \text{ output } 1 \text{ if } \text{Verify}(k, F(k_f, r) \oplus t)
\]

Advantage : Use a fast one-time MAC (no cryptography!) on long message
- Apply cryptographic operation to short output (slower)
- Paradigm used in GCM mode of operation

PolyB05
GCM encryption: encrypt message with AES in counter mode.

- Compute Carter-Wegman MAC on resulting message using GHASH as the underlying hash function and the block cipher as underlying PRF.

Typically, use AES-GCM for authenticated encryption.

- GHASH operates on blocks of 128 bits.
- Operations can be expressed as operations over $GF(q^{128}) -$ Galois field with $q^{128}$ elements implemented in hardware - very fast!

$\mathbb{GF}(q^{128})$ is defined by the polynomial $g(x) = x^{128} + x^7 + x^2 + x + 1$.

- Elements are polynomials over $\mathbb{F}_2$ with degree less than 128, e.g. $x^{27} + x^{51} + x + 1$.
- Can be represented by 128-bit string: each bit is coefficient of polynomial.
- Can add elements (XOR) and multiply them (as polynomials) - implemented in hardware.

GHASH $(K, m)$ := $m[1] k + m[2] k^{-1} + \ldots + m[l] k$ [values $m[1], \ldots, m[l]$ give coefficients of polynomial, evaluate at point $K$]

- Same as one-time MAC from above.

Oftentimes, only part of the payload needs to be hidden, but still needs to be authenticated.

- E.g., sending packets over a network: desire confidentiality for packet body, but only integrity for packet headers (otherwise, cannot route!)

AEAD: authenticated encryption with associated data.

- August encryption scheme with additional plaintext input; resulting ciphertext ensures integrity for associated data, but not confidentiality.

AEAD: authenticated encryption with associated data.

- Can construct directly via “encrypt-then-MAC”: namely, encrypt payload and MAC the ciphertext + associated data.

AES-GCM is an AEAD scheme.