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 chose 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 needs to be computed over the entire ciphertext
  - Early version of ISO 19772 for AE did not MAC IV (CBC used for CPA-secure encryption)
  - RNCryptor in Apple iOS (for data encryption) also problematic (HMAC not applied to encryption IV)

  ] means first block (i.e, "header") is malleable

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:

Encrypt'$((k_E, k_M), m)$: $t \leftarrow \text{Sign}(k_M, m)$
$c \leftarrow \text{Encrypt}(k_E, (m,t))$
output c

Decrypt'$((k_E, k_M), (c,t))$: compute $(m,t) \leftarrow \text{Decrypt}(k_E, c)$
if $\text{Verify}(k_M, m, t) = 1$, output m, else, output $\perp$

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)
[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 HW2 for an example]

In the past, libraries provided separate encryption + MAC interfaces — common source of errors
↳ Good library design for crypto 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 is nonce-based counter mode
- MAC is a Carter-Wegman MAC

} Most commonly used in conjuction with AES (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) $1/p = \text{negl}(\lambda)$
  KeyGen: sample $\alpha, \beta \xleftarrow{R} \mathbb{Z}_p$ (two random integers in $\{0, \ldots, p-1\}$). Key is $k = (\alpha, \beta)$
  Sign(k, m): output $\tau = \alpha m + \beta \pmod p$
  Verify(k, m, τ): accept if $\tau = \alpha m + \beta \pmod p$ 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 $\alpha, \beta \xleftarrow{R} \mathbb{Z}_p$, we can show that for any choice of $m \neq m'$ and $\tau, \tau' \in \mathbb{Z}_p$:

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

$$\hookrightarrow \underbrace{\begin{bmatrix} m & 1 \\ m' & 1 \end{bmatrix}}_{\substack{\text{linearly independent} \\ \text{if } m \neq m'}} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} \tau \\ \tau' \end{bmatrix} \implies \underbrace{\begin{bmatrix} \alpha \\ \beta \end{bmatrix}}_{\text{uniform}} = \begin{bmatrix} m & 1 \\ m' & 1 \end{bmatrix}^{-1} \begin{bmatrix} \tau \\ \tau' \end{bmatrix}$$

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

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

regardless of adversary's running time!

- For longer messages $m = m_1 m_2 \cdots m_\ell$ where each $m_i \in \mathbb{Z}_p$, define the MAC to be

$$\tau = m_1 \alpha^\ell + m_2 \alpha^{\ell-1} + \cdots + m_\ell \alpha + \beta \pmod p$$

very fast to evaluate: to process each message block: multiply by α and add current block

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

$$\begin{aligned} \tau &= \beta + \textstyle\sum_{i \in [\ell]} m_i \alpha^{\ell-i+1} \\ \tau' &= \beta + \textstyle\sum_{i \in [\ell]} m_i' \alpha^{\ell-i+1} \end{aligned} \implies \underbrace{\sum_{i \in [\ell]} (m_i - m_i') \alpha^{\ell-i+1} - (\tau - \tau')}_{\text{polynomial of degree at most } \ell} = 0$$

$$\Pr_{\alpha \xleftarrow{R} \mathbb{Z}_p}[\alpha \text{ is a root of this poly}] = \frac{\ell}{p}$$

the tag τ adversary sees perfectly hides α since $\beta \xleftarrow{R} \mathbb{Z}_p$, so m, m', τ, τ' is independent of α

Carter-Wegman MAC ("encrypted MAC"): very lightweight, randomized MAC from the one-time MAC:

- Let (Sign, Verify) be a one-time MAC (or more generally, a universal hash function)
- Let $F: \mathcal{K}_F \times \mathcal{R} \to \{0,1\}^n$ be a PRF

The Carter-Wegman MAC is defined as follows:

Sign$((k_M, k_F), m)$: $r \xleftarrow{R} \mathcal{R}$
$t \leftarrow \text{Sign}(k_M, m) \oplus F(k_F, r)$
output (r, t)

key for one-time MAC

Verify$((k_M, k_F), (r, t))$: output 1 if Verify$(k_M, F(k_F, r) \oplus t)$ and 0 otherwise

Very simple construction!
but tags are longer (need both a nonce and a PRF output)

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 and in Poly1305

Polynomial evaluation over $\mathbb{Z}_p$ ($p = 2^{130} - 5$)

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

Galois Hash

key derived from PRF evaluation at 0^n

GHASH operates on blocks of 128-bits

operations can be expressed as operations over $GF(2^{128})$ — Galois field with 2^{128} elements

implemented in hardware — very fast!

Typically, use AES-GCM for authenticated encryption

$GF(2^{128})$ 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. $x^{127} + x^{52} + x^2 + x + 1$]

(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

(also used for evaluating the AES round function)

↳ $\mathrm{GHASH}(k, \underbrace{m}_{(m[1], m[1], \ldots, m[\ell])}) := m[1]k^{\ell} + m[2]k^{\ell-1} + \cdots + m[\ell]k$ [values $m[1], \ldots, m[\ell]$ 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

↳ augment encryption scheme with additional plaintext input; resulting ciphertext ensures integrity for associated data, but not confidentiality (will not define formally here but follows straightforwardly from AE definitions)

↳ can construct directly via "encrypt-then-MAC": namely, encrypt payload and MAC the ciphertext + associated data

↳ AES-GCM is an AEAD scheme