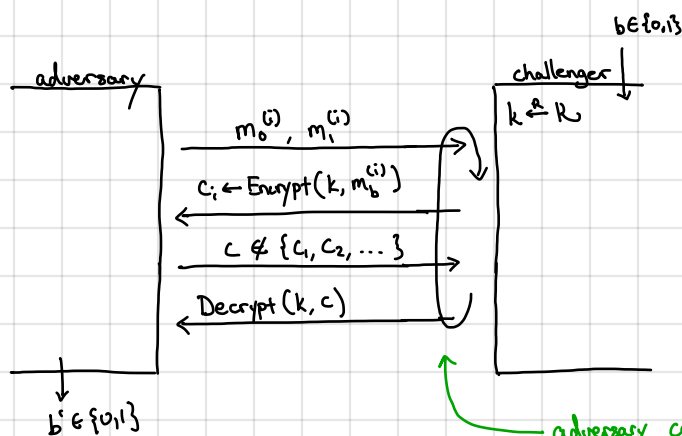


Definition. An encryption scheme $\Pi_{SE}(\text{Encrypt}, \text{Decrypt})$ is secure against chosen-ciphertext attacks (CCA-secure) if for all efficient adversaries A , $\text{CCAAdv}[A, \Pi_{SE}] = \text{negl.}$ where we define $\text{CCAAdv}[A, \Pi_{SE}]$ as follows:



adversary can make arbitrary encryption and decryption queries, but cannot decrypt any ciphertexts it received from the challenger (otherwise, adversary can trivially break security)
 $\rightarrow$ called an "admissibility" criterion

$$\text{CCAAdv}[A, \Pi_{SE}] = |\Pr[b' = 1 \mid b = 0] - \Pr[b' = 1 \mid b = 1]|$$

CCA-security captures above attack scenario where adversary can tamper with ciphertexts

- $\rightarrow$ Rules out possibility of transforming encryption of $x \parallel z$ to encryption of $y \parallel z$
- $\rightarrow$ Necessary for security against active adversaries (CPA-security is for security against passive adversaries)
- $\rightarrow$ We will see an example of a real CCA attack in HW1

Theorem. If an encryption scheme Π_{SE} provide authenticated encryption, then it is CCA-secure.

Proof (Idea). Consider an adversary A in the CCA-security game. Since Π_{SE} provides ciphertext integrity, the challenger's response to the adversary's decryption query will be $\perp$ with all but negligible probability. This means we can implement the decryption oracle with the "output $\perp$ " function. But then this is equivalent to the CPA-security game.

[Formalize using a hybrid argument]

simple counter-example: concatenate unused bits to end of ciphertext in a CCA-secure scheme (stripped away during decryption)

Note: Converse of the above is not true since CCA-security $\nRightarrow$ ciphertext integrity.

$\rightarrow$ However, CCA-security + plaintext integrity $\Rightarrow$ authenticated encryption

Take-away: Authenticated encryption captures meaningful confidentiality + integrity properties; provides active security

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

$\text{Encrypt}'((k_E, k_M), m): c \leftarrow \text{Encrypt}(k_E, m)$

$\uparrow \quad \uparrow$
independent keys

$t \leftarrow \text{Sign}(k_M, c)$

output (c, t)

$\text{Decrypt}'((k_E, k_M), (c, t)): \text{if } \text{Verify}(k_M, c, t) = 0, \text{ output } \perp$
 else, output $\text{Decrypt}(k_E, c)$

Theorem. If $(\text{Encrypt}, \text{Decrypt})$ is CPA-secure and $(\text{Sign}, \text{Verify})$ is a secure MAC, then $(\text{Encrypt}', \text{Verify}')$ is an authenticated encryption scheme.

Proof (Sketch). CPA-security follows by CPA-security of $(\text{Encrypt}, \text{Decrypt})$. 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 choose 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 $(\text{Encrypt}, \text{Verify})$ be a CPA-secure encryption scheme and $(\text{Sign}, \text{Verify})$ be a secure MAC. We define MAC-then-Encrypt to be the following scheme:

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

$c \leftarrow \text{Encrypt}(k_E, (m, t))$

output c

$\text{Decrypt}'((k_E, k_M), (c, t)): \text{compute } (m, t) \leftarrow \text{Decrypt}(k_E, c)$

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 conjunction 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)
 KeyGen: sample $\alpha, \beta \xleftarrow{R} \mathbb{Z}_p$ (two random integers in $\{0, \dots, p-1\}$). 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 $\tau, \tau' \in \mathbb{Z}_p$:

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

$$\hookrightarrow \begin{bmatrix} m & 1 \\ m' & 1 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} \tau \\ \tau' \end{bmatrix} \Rightarrow \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} m & 1 \\ m' & 1 \end{bmatrix}^{-1} \begin{bmatrix} \tau \\ \tau' \end{bmatrix}$$

$\uparrow$ linearly independent if $m \neq m'$ $\uparrow$ uniform

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

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

regardless of adversary's running time!

- For longer messages $m = m_1 m_2 \dots m_\ell$ where each $m_i \in \mathbb{Z}_p$, define the MAC to be
 $\tau = m_1 \alpha^\ell + m_2 \alpha^{\ell-1} + \dots + m_\ell \alpha + \beta \pmod{p}$

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 $\tau, \tau' \in \mathbb{Z}_p$:

$$\tau = \beta + \sum_{i \in [\ell]} m_i \alpha^{\ell-i+1}$$

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

polynomial of degree at most ℓ at $\alpha \in \mathbb{Z}_p$ $\left\{ \Pr[\alpha \text{ is a root of this poly}] = \frac{\ell}{p} \right\}$

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 $F: K_F \times R \rightarrow \{0,1\}^n$ be a PRF

The Carter-Wegman MAC is defined as follows:

$$\text{Sign}((k_m, k_f), m): r \xleftarrow{R} R$$

$\uparrow$
key for one-time MAC

$$t \leftarrow \text{Sign}(k_m, m) \oplus F(k_f, r)$$

output (r, t)

$$\text{Verify}((k_m, k_f), (r, t)): \text{output } 1 \text{ if } \text{Verify}(k_m, F(k_f, r) \oplus t) \text{ and } 0 \text{ otherwise}$$

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 PolyBOB

Polynomial evaluation over $\mathbb{Z}_p$ ($p=2^{30}-5$)