# CCA-Secure and Authentication Encryption

Yu Zhang

HIT/CST/NIS

Cryptography, Spring, 2012

# Outline

# Content

# Recall Security Against CCA

The CCA indistinguishability experiment $\mathsf{PrivK}^{\mathsf{cca}}_{\mathcal{A},\Pi}(n)$:

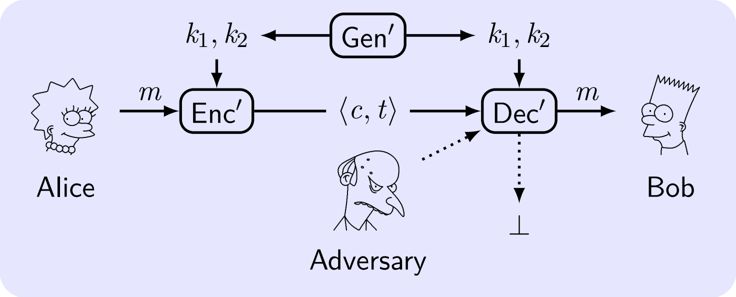
1. $k \leftarrow \mathsf{Gen}(1^n)$.
2. $\mathcal{A}$ is given input $1^n$ and oracle access $\mathcal{A}^{\mathsf{Enc}_k(\cdot)}$ and $\mathcal{A}^{\mathsf{Dec}_k(\cdot)}$, outputs $m_0, m_1$ of the same length.
3. a random bit $b \leftarrow \{0,1\}$ is chosen. Then $c \leftarrow \mathsf{Enc}_k(m_b)$ is given to $\mathcal{A}$.
4. $\mathcal{A}$ continues to have oracle access **except for** $c$, outputs $b'$.
5. If $b' = b$, $\mathcal{A}$ succeeded $\mathsf{PrivK}^{\mathsf{cca}}_{\mathcal{A},\Pi} = 1$, otherwise 0.

**Definition 1**

$\Pi$ has **indistinguishable encryptions under a CCA (CCA-secure)** if $\forall$ PPT $\mathcal{A}$, $\exists$ negl such that

$$\Pr\left[\mathsf{PrivK}^{\mathsf{cca}}_{\mathcal{A},\Pi}(n) = 1\right] \leq \frac{1}{2} + \mathsf{negl}(n).$$

# Constructing CCA-Secure Encryption Schemes




## Construction 2

$\Pi_E = (\mathsf{Gen}_E, \mathsf{Enc}, \mathsf{Dec})$, $\Pi_M = (\mathsf{Gen}_M, \mathsf{Mac}, \mathsf{Vrfy})$. $\Pi'$:

- $\mathsf{Gen}'(1^n)$: $k_1 \leftarrow \mathsf{Gen}_E(1^n)$ *and* $k_2 \leftarrow \mathsf{Gen}_M(1^n)$.
- $\mathsf{Enc}'_{k_1,k_2}(m)$: $c \leftarrow \mathsf{Enc}_{k_1}(m)$, $t \leftarrow \mathsf{Mac}_{k_2}(c)$ *and output* $\langle c, t\rangle$.
- $\mathsf{Dec}'_{k_1,k_2}(\langle c, t\rangle)$: *If* $\mathsf{Vrfy}_{k_2}(c, t) \stackrel{?}{=} 1$, *output* $\mathsf{Dec}_{k_1}(c)$; *otherwise output "failure"* $\perp$.

# Proof of CCA-Secure Encryption Schemes

### Theorem 3

*If $\Pi_E$ is a CPA-secure private-key encryption scheme and $\Pi_M$ is a secure MAC with unique tags, then Construction $\Pi'$ is CCA-secure.*

**Idea**: The decryption oracle is useless. CCA = CPA-then-MAC.

### Proof.

VQ: $\mathcal{A}$ submits a "new" query to oracle $\mathsf{Dec}'$ and $\mathsf{Vrfy} = 1$.

$$\Pr[\mathsf{PrivK}^{\mathsf{cca}}_{\mathcal{A},\Pi'}(n) = 1] \leq \Pr[\mathsf{VQ}] + \Pr[\mathsf{PrivK}^{\mathsf{cca}}_{\mathcal{A},\Pi'}(n) = 1 \wedge \overline{\mathsf{VQ}}]$$

We need to prove the following claims.

1. $\Pr[\mathsf{VQ}]$ is negligible.
2. $\Pr[\mathsf{PrivK}^{\mathsf{cca}}_{\mathcal{A},\Pi'}(n) = 1 \wedge \overline{\mathsf{VQ}}] \leq \frac{1}{2} + \mathsf{negl}(n)$.

$\square$

# Proof of "$\Pr[\text{VQ}]$ is negligible"

**Idea**: Reduce $\mathcal{A}_M$ (attacking $\Pi_M$ with an oracle $\mathsf{Mac}_{k_2}(\cdot)$) to $\mathcal{A}$.

**Proof.**

- $\mathcal{A}_M$ chooses $i \leftarrow \{1, \ldots, q(n)\}$ *u.a.r.*
- Run $\mathcal{A}$ with the encryption/decryption oracles.
- If the $i$th decryption oracle query from $\mathcal{A}$ uses a "new" $c$, output $(c, t)$ and stop.
- $\mathsf{Macforge}_{\mathcal{A}_M,\Pi_M}(n) = 1$ only if VQ occurs.
- $\mathcal{A}_M$ correctly guesses $i$ with probability $1/q(n)$.

$$\Pr[\mathsf{Macforge}_{\mathcal{A}_M,\Pi_M}(n) = 1] \geq \Pr[\mathsf{VQ}]/q(n).$$

$\square$

# Proof of "$\Pr[\mathsf{PrivK}^{\mathsf{cca}}_{\mathcal{A},\Pi'}(n) = 1 \wedge \overline{\mathsf{VQ}}] \leq \frac{1}{2} + \mathsf{negl}(n)$"

**Idea**: Reduce $\mathcal{A}_E$ (attacking $\Pi_E$ with an oracle $\mathsf{Enc}_{k_1}(\cdot)$) to $\mathcal{A}$.

**Proof.**

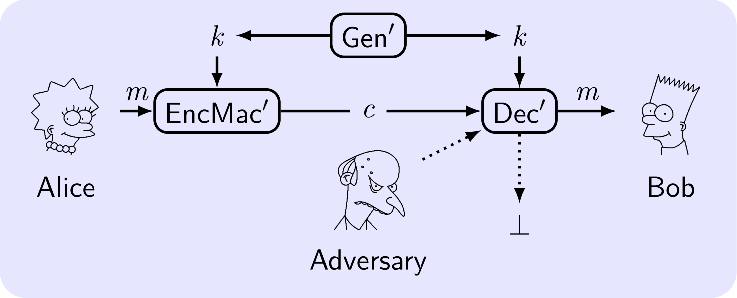
- Run $\mathcal{A}$ with the encryption/decryption oracles.
- Run $\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A}_E,\Pi_E}$ as $\mathsf{PrivK}^{\mathsf{cca}}_{\mathcal{A},\Pi'}$.
- $\mathcal{A}_E$ outputs the same $b'$ that is output by $\mathcal{A}$.
- $\Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A}_E,\Pi_E}(n) = 1 \wedge \overline{\mathsf{VQ}}] = \Pr[\mathsf{PrivK}^{\mathsf{cca}}_{\mathcal{A},\Pi'}(n) = 1 \wedge \overline{\mathsf{VQ}}]$ unless VQ occurs.

$$\Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A}_E,\Pi_E}(n) = 1] \geq \Pr[\mathsf{PrivK}^{\mathsf{cca}}_{\mathcal{A},\Pi'}(n) = 1 \wedge \overline{\mathsf{VQ}}].$$

$\square$

# Content

# Message Transmission Scheme




- **Key-generation** algorithm outputs $k \leftarrow \mathsf{Gen}'(1^n)$.
  $k = (k_1, k_2)$. $k_1 \leftarrow \mathsf{Gen}_E(1^n)$, $k_2 \leftarrow \mathsf{Gen}_M(1^n)$.
- **Message transmission** algorithm is derived from $\mathsf{Enc}_{k_1}(\cdot)$ and $\mathsf{Mac}_{k_2}(\cdot)$, outputs $c \leftarrow \mathsf{EncMac}'_{k_1,k_2}(m)$.
- **Decryption** algorithm is derived from $\mathsf{Dec}_{k_1}(\cdot)$ and $\mathsf{Vrfy}_{k_2}(\cdot)$, outputs $m \leftarrow \mathsf{Dec}'_{k_1,k_2}(c)$ or $\perp$.
- **Correctness requirement**: $\mathsf{Dec}'_{k_1,k_2}(\mathsf{EncMac}'_{k_1,k_2}(m)) = m$.

# Defining Secure Message Transmission

The secure message transmission experiment $\mathsf{Auth}_{\mathcal{A},\Pi'}(n)$:

1. $k = (k_1, k_2) \leftarrow \mathsf{Gen}'(1^n)$.
2. $\mathcal{A}$ is given input $1^n$ and oracle access to $\mathsf{EncMac}'_k$, and outputs $c \leftarrow \mathsf{EncMac}'_k(m)$.
3. $m := \mathsf{Dec}'_k(c)$. $\mathsf{Auth}_{\mathcal{A},\Pi'}(n) = 1 \iff m \neq \perp \wedge\ m \notin \mathcal{Q}$.

### Definition 4

$\Pi'$ achieves **authenticated communication** if $\forall$ PPT $\mathcal{A}$, $\exists$ negl such that

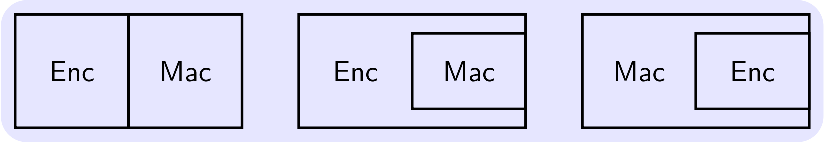
$$\Pr[\mathsf{Auth}_{\mathcal{A},\Pi'}(n) = 1] \leq \mathsf{negl}(n).$$

### Definition 5

$\Pi'$ is **secure (authenticated encryption)** if it is both CCA-secure and also achieves authenticated communication.[1]

[1] CPA security and integrity imply CCA security.

# Combining Encryption and Authentication




- **Encrypt-and-authenticate** (e.g., SSH):

$$c \leftarrow \mathsf{Enc}_{k_1}(m),\ t \leftarrow \mathsf{Mac}_{k_2}(m).$$

- **Authenticate-then-encrypt** (e.g, SSL):

$$t \leftarrow \mathsf{Mac}_{k_2}(m),\ c \leftarrow \mathsf{Enc}_{k_1}(m\|t).$$

- **Encrypt-then-authenticate** (e.g, IPsec):

$$c \leftarrow \mathsf{Enc}_{k_1}(m),\ t \leftarrow \mathsf{Mac}_{k_2}(c).$$

# Analyzing Security of Combinations

**All-or-nothing**: Reject any combination for which there exists even a single counterexample is insecure.

- **Encrypt-and-authenticate**: $\mathsf{Mac}'_k(m) = (m, \mathsf{Mac}_k(m))$.
- **Authenticate-then-encrypt**:
  - $\mathsf{Trans} : 0 \to 00; 1 \to 10/01$; $\mathsf{Enc}'$ uses CRT mode; $c = \mathsf{Enc}'(\mathsf{Trans}(m\|\mathsf{Mac}(m)))$.
  - Flip the first two bits of $c$ and verify whether the ciphertext is valid. $10/01 \to 01/10 \to 1$, $00 \to 11 \to \perp$.
  - If valid, the first bit of message is 1; otherwise 0.
  - For any MAC, this is not CCA-secure.
- **Encrypt-then-authenticate**:
  Decryption: If $\mathsf{Vrfy}(\cdot) = 1$, then $\mathsf{Dec}(\cdot)$; otherwise output $\perp$.

# Authenticated Encryption Theory and Practice

**Theorem 6**

*$\Pi_E$ is CPA-secure and $\Pi_E$ is a secure MAC with unique tages, $\Pi'$ deriving from encrypt-then-authenticate approach is secure.*

**GCM(Galois/Counter Mode)**: CTR encryption then Galois MAC. (RFC4106/4543/5647/5288 on IPsec/SSH/TLS)
**EAX**: CTR encryption then CMAC.

**Proposition 7**

*Encrypt-then-authenticate approach is secure if $\Pi_E$ is rand-CTR mode or rand-CBC mode.*

**CCM (Counter with CBC-MAC)**: CBC-MAC then CTR encryption. (802.11i, RFC3610)
**OCB (Offset Codebook Mode)**: integrating MAC into ENC. (two times fast as CCM, EAX)
**All support AEAD (A.E. with associated data):** part of message is in clear, and all is authenticated.

# Remarks on Secure Message Transmission

- Authentication may leak the message.
- Secure message transmission implies CCA-security. The opposite direction is not necessarily true.
- Different security goals should always use different keys.
  - otherwise, the message may be leaked if $\mathsf{Mac}_k(c) = \mathsf{Dec}_k(c)$.
- Implementation may destroy the security proved by theory.
  - **Attack with padding oracle** (in TLS 1.0):
    **Dec** return two types of error: padding error, MAC error.
    **Adv.** learns last bytes if no padding error with guessed bytes.
  - **Attack non-atomic dec.** (in SSH Binary Packet Protocol):
    **Dec** (1)decrypt length field; (2)read packets as specified by the length; (3)check MAC.
    **Adv.** (1)send $c$; (2)send $l$ packets until "MAC error" occurs; (3)learn $l = \mathsf{Dec}(c)$.

# Content

# Key Derivation Function (KDF)

**Key Derivation Function (KDF)** generates many keys from a secret source key $sk$.

**For uniformly random** $sk$: $F$ is PRF, $ctx$ is a unique string identifying application,

$$\mathsf{KDF}(sk, ctx, l) = \langle F_{sk}(ctx\|0), F_{sk}(ctx\|1) \cdots, F_{sk}(ctx\|l)\rangle .$$

**For not-uniform** $sk$: extract-then-expand paradigm.

**extract:** HKDF $k \leftarrow \mathsf{HMAC}(salt, sk)$. $salt$ is a random number.

**expand:** as the above.

# Password-Based KDF (PBKDF)

**Key stretching** increases the time of testing key (with slow hash function).

**Key strengthening** increases the length/randomness of key (with salt).

**PKCS#5 (PBKDF1)**: $H^{(c)}(pwd \| salt)$, iterate hash function $c$ times.

**Attack**: either try the enhanced key (larger key space), or else try the initial key (longer time per key).

# IV, Nonce, Counter and Salt

**IV** an input to a cryptographic primitive, providing randomness.

**nonce** a number used only once to sign a communication.

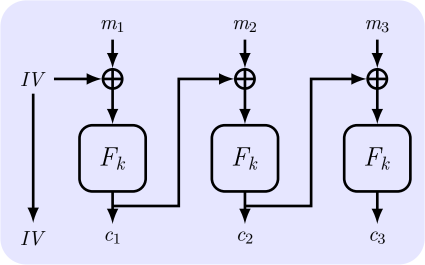
**counter** a sequence number used as nonce.

**salt** consists of random bits, creating the input to a function.

# Content

# Deterministic CPA Security

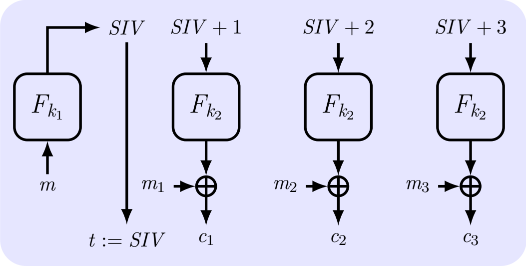
- **Deterministic encryption**: the same message is encrypted to the same ciphertext under the same key.
- **Application**: encrypted database index, disk encryption.
- **But no deterministic encryption is CPA-secure.**
- **Deterministic CPA Security**: CPA-secure if *never encrypt same message twice* using same key. The pair $\langle k, m \rangle$ is unique.
- **Common Mistake**: CBC/CTR with **fixed** $IV$.




Adversary can query $(m_{q1}, m_{q2})$ and get $(c_{q1}, c_{q2})$; then output PT: $IV \oplus c_{q1} \oplus m_{q2}$ and expect CT: $c_{q2}$.

# Synthetic $IV$ (SIV) for Det. Encryption

- **Synthetic $IV$ (SIV)**:
  If $F$ is PRF, $\Pi : (\mathsf{Enc}_k(r, m), \mathsf{Dec}_k(r, s))$ is CPA-secure, then $\Pi_{\mathsf{det}}$ is det. CPA-secure scheme:
  $(k_1, k_2) \leftarrow \mathsf{Gen}$; $SIV \leftarrow F_{k_1}(m)$; $s \leftarrow \mathsf{Enc}_{k_2}(SIV, m)$;
  output $c = \langle SIV, s \rangle$.
- **Deterministic Authenticated Encryption (DAE)**: det. CPA-security and integrity.
- **DAE for free with SIV-CTR**: Tag $t := SIV \leftarrow F_{k_1}(m)$ then $CTR_{k_2}$ encryption.

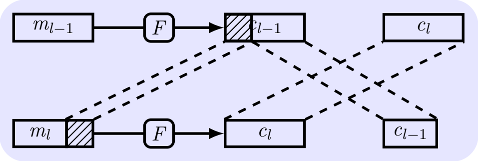


# Wide Block PRP for Det. Encryption

- **Just PRP**: If $F$ is PRP, then $F$ is also det. CPA-secure.
- **PRP-based DAE**: $\mathsf{Enc}_k(m\|0^\ell)$. In Dec, if $\neq 0^\ell$, output $\perp$.
- **Narrow block** may leak info. as some blocks are the same.
- **Wide block PRP**: PRP with longer block length (e.g. a sector on disk) from PRP with short block length (e.g. AES).
- **Standards**: CBC-mask-CBC (CMC) and ECB-mask-ECB (EME) in IEEE P1619.2.
- **Cost**: 2x slower than SIV due to two-pass encryption.

# Tweakable Encryption

- **Encryption without expansion**: $\mathcal{M} = \mathcal{C}$ implies det. encryption without integrity (e.g., disk encryption).
- **Trivial solution**: $k_t = F_k(t), t = 1, \ldots, \ell$.
- **Tweakable block ciphers**: many PRPs from one key $\mathcal{K} \times \mathcal{T} \times \mathcal{X} \to \mathcal{X}$, $\mathcal{T}$ is the set of tweaks.
- **XTS**: XEX(Xor-Encrypt-Xor)-based tweaked-codebook mode with ciphertext stealing. (XTS-AES, NIST SP 800-38E)
- **XEX**: To encrypt block $j$ in sector $I$, $c = F_k(m \oplus x) \oplus x$, where $x = F_k(I) \otimes 2^j$ in Galois field, $(I, j)$ is tweak.
- **Ciphertext stealing (CTS)**: no padding, **no expansion**.



# Summary

- CCA-secure, AE, det. enc., det. CPA-secure, DAE.
- Enc-then-auth, KDF, SIV, wide block cipher, tweakable encryption.
- SIV-CTR, PBKDF, salt, enc. w/o expansion, CTS.