Tutorial 3 - Symmetric Encryption

CS3235 - Spring 2022

Never write your own crypto!

Use the standard libraries Luke!

One-time pad

Message Space (M) = Ciphertext Space (C) = Key Space (K) = $\{0,1\}^n$

Key is chosen randomly.

Encryption Algorithm $E(k,m) = k \oplus m$

Decryption Algorithm $D(k,m) = k \oplus c$

where $k \in K$, $m \in M$, $c \in C$

OTP demonstrates “perfect secrecy”

Perfect Secrecy

A cipher (E,D) has perfect secrecy if for any 2 messages m_0, m_1 of the same length, the probability that a ciphertext c is an encryption of message m_0 with key k is the same as the probability that c is an encryption of message m_1 with key k.

Which means

Given a ciphertext, every message in the message space is exactly as likely to be the underlying plaintext.

$$Pr(M = m_1 \mid C = c) = Pr(M = m_2 \mid C = c)$$

Lemma: OTP has perfect secrecy

$Pr(M = m_1 \mid C = c) = Pr(K = k \mid m_1 \oplus k = c)$

$= Pr(K = k \mid m_1 \oplus k \oplus m_1 = c \oplus m_1)$

$= Pr(K = k \mid k = c \oplus m_1)$

$= \frac{1}{|K|}$

Similarly,

$Pr(M = m_2 \mid C = c) = \frac{1}{|K|}$

Hence Proved.

Problems with perfect secrecy

Shannon's theorem:

For any perfectly secure cipher |K| >= |M|

Need to create and share a new key which is the same length as of the message to be encrypted.

Imagine coming up with keys to encrypt 10 movies that are 5 GB long each. => Impractical

Redefine Security: Semantic Security

For a computationally bound attacker,

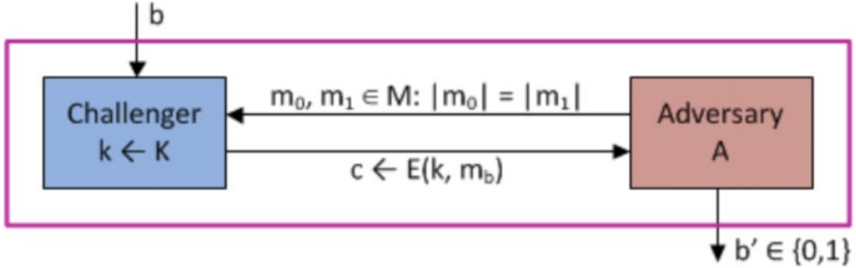


Figure: Semantic Security, Dan Boneh's course

- E is semantically secure if for all efficient adversaries,

  $Adv_{SS}[A,E]$ = | Pr(Exp(0) = 1) - Pr(Exp(1) = 1) | is negligible

- The adversary cannot distinguish the ciphertext belonging to m_0 and m_1

Pseudorandom Generators (PRG)

Idea: replace the random key with a pseudorandom key.

From a small key deterministically generate a much larger key that appears to be random

G: $\{0,1\}^s \rightarrow \{0,1\}^l$, $l >> s$

A PRG is secure if observing up to n bits of the output an attacker cannot predict the next bit with probability better than 0.5 (without knowing the seed s)

Pseudorandom Generators (PRG)

Is XOR(G(k)) = 1 a PRG?

No, given n bits, can predict the n+1 bit such that XOR = 1

Stream Ciphers

Encryption Algorithm $E(k,m) = G(k) \oplus m$

Decryption Algorithm $D(k,m) = G(k) \oplus c$

where $k \in K$, $m \in M$, $c \in C$

If a PRG is secure, then the stream cipher built with it is also secure

Proof of reduction:

Reducing the problem of breaking stream ciphers to that of breaking PRGs

Block ciphers

Function that encrypts fixed-size blocks of data.

Deterministic function $F : K \times X \to Y$

where K is ℓ-bit key space $X, Y = \{0,1\}^b$

- F(k,m) should be efficient to compute
- Given k, $F^{-1}(k, .)$ should exists
- Pseudorandom permutation: For any chosen key k, $F(k,\cdot)$ is computationally infeasible to distinguish from a permutation chosen arbitrarily at random from the set of all permutation functions from $\{0,1\}^b$ to $\{0,1\}^b$

Block chaining Modes

Electronic Code Book

(ECB)

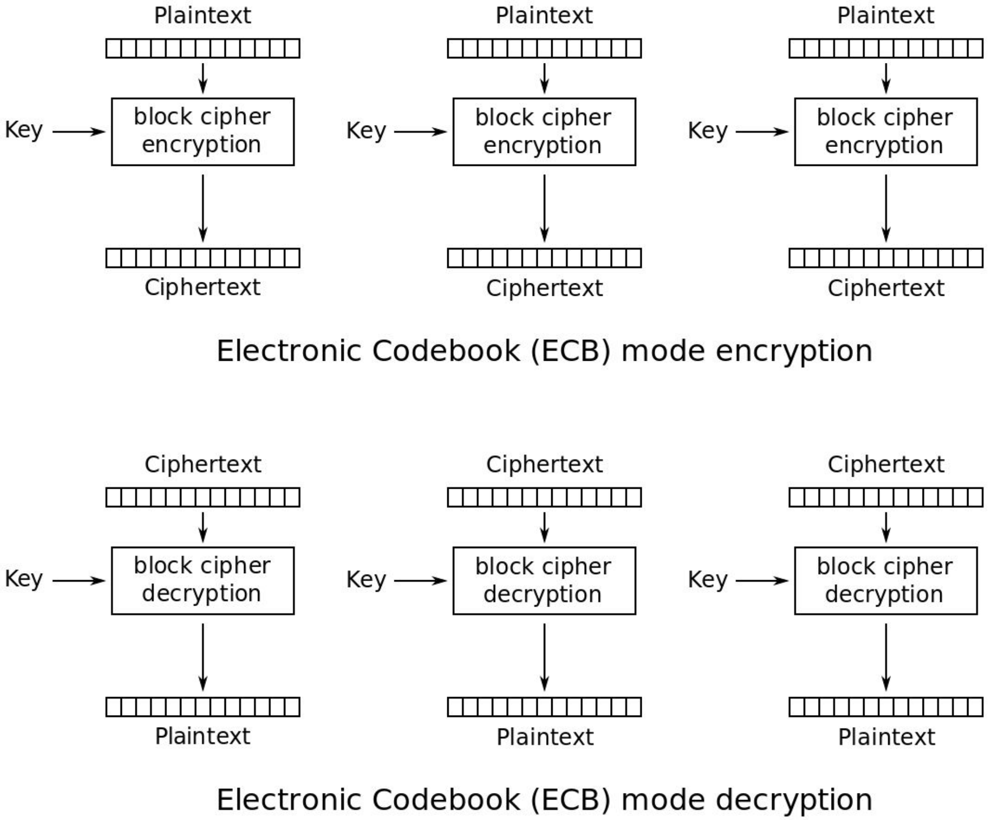
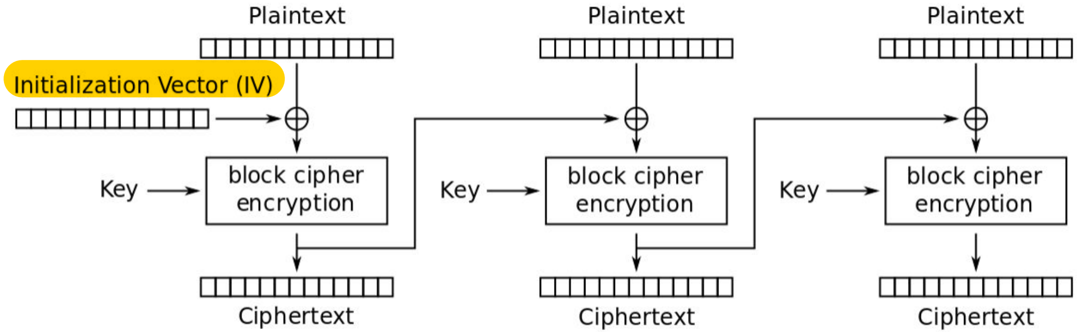


Figure: Wikipedia

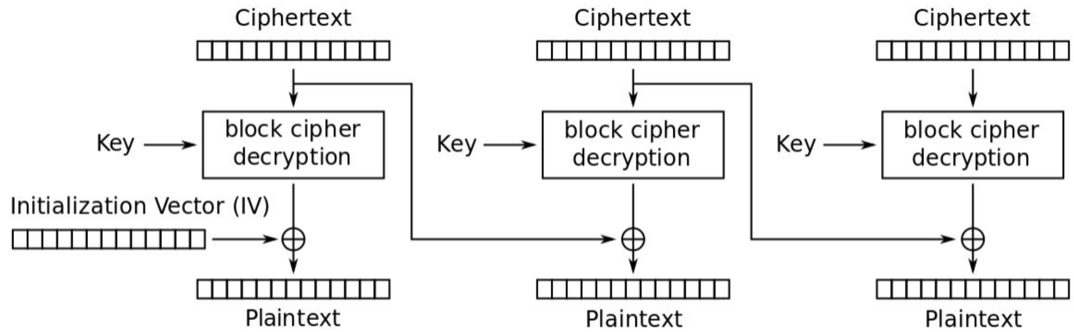
Block chaining Modes

Cipher Block Chaining

(CBC)



Cipher Block Chaining (CBC) mode encryption



Cipher Block Chaining (CBC) mode decryption

Figure: Wikipedia

Block chaining Modes

Counter Mode

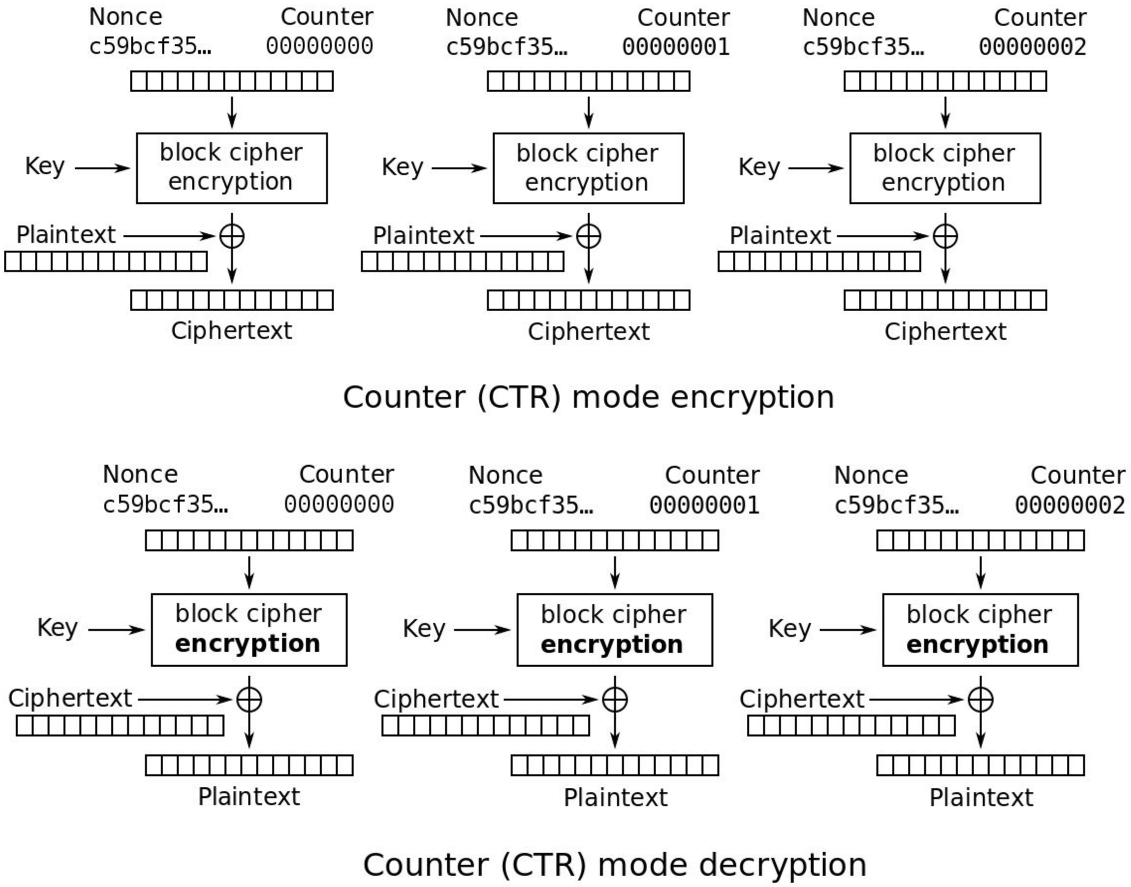
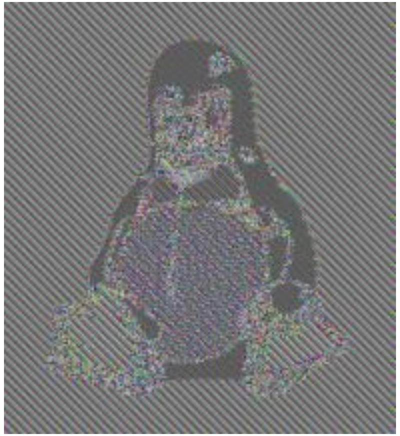
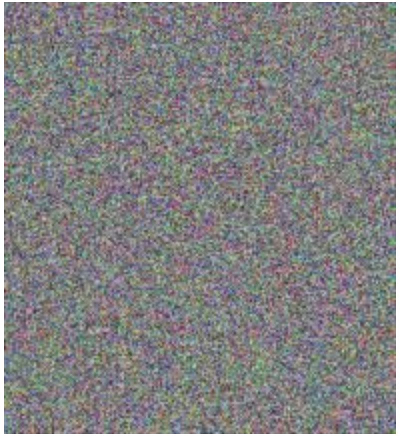


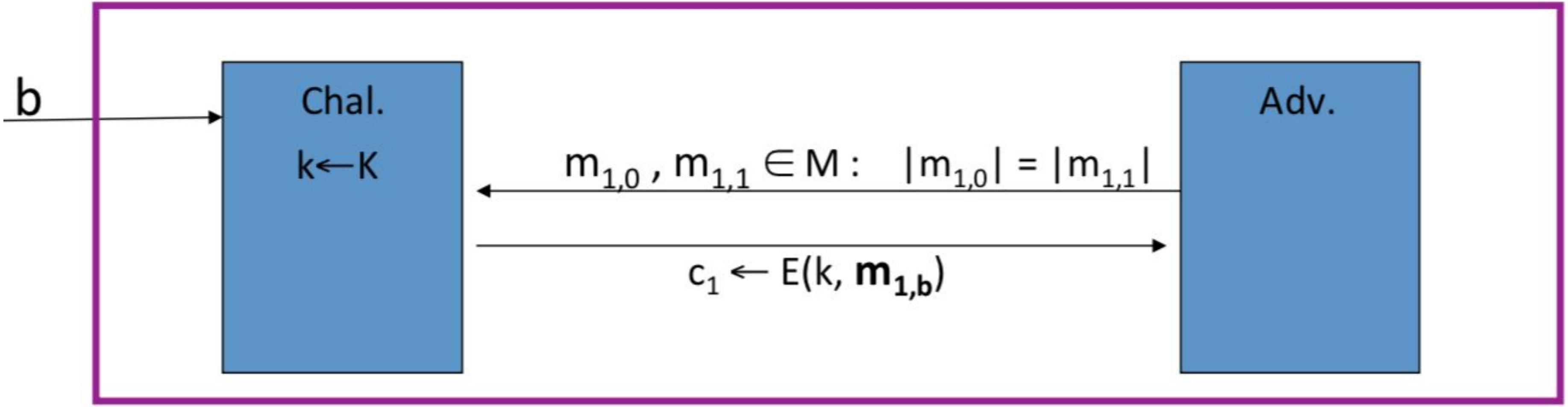
Figure: Wikipedia

Problems with ECB

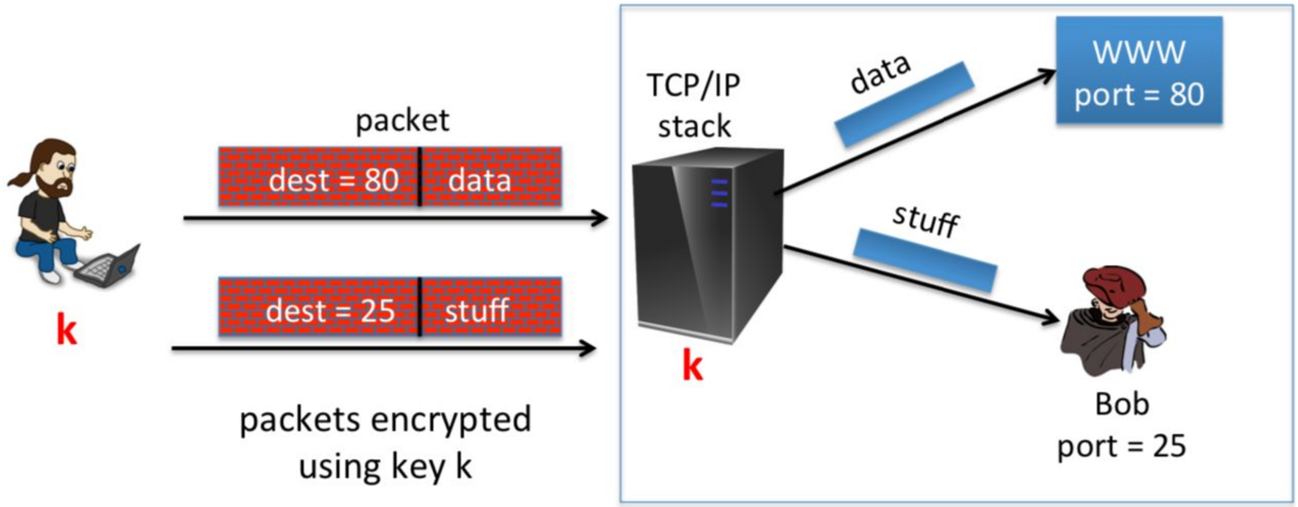




Chosen Plaintext Attack (CPA) Security

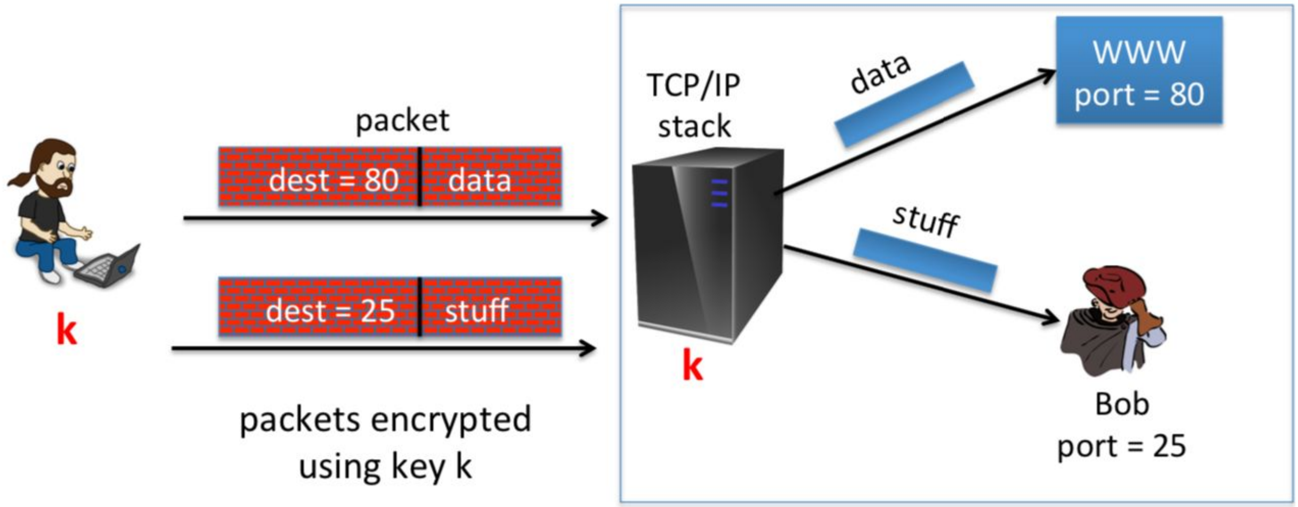


Is CPA security enough?



Is CPA security enough?

No not enough, although the ciphertext is not understood by the attacker - the ciphertext is still malleable



$$IV' = IV \oplus (\dots Port{:}\,80 \dots) \oplus (\dots Port{:}\,25 \dots)$$

$$\begin{aligned} m'[0] &= D(k, c[0]) \oplus IV' \\ &= D(k, c[0]) \oplus IV \oplus (\dots Port{:}\,80 \dots) \oplus (\dots Port{:}\,25 \dots) \\ &= m[0] \oplus (\dots Port{:}\,80 \dots) \oplus (\dots Port{:}\,25 \dots) \end{aligned}$$

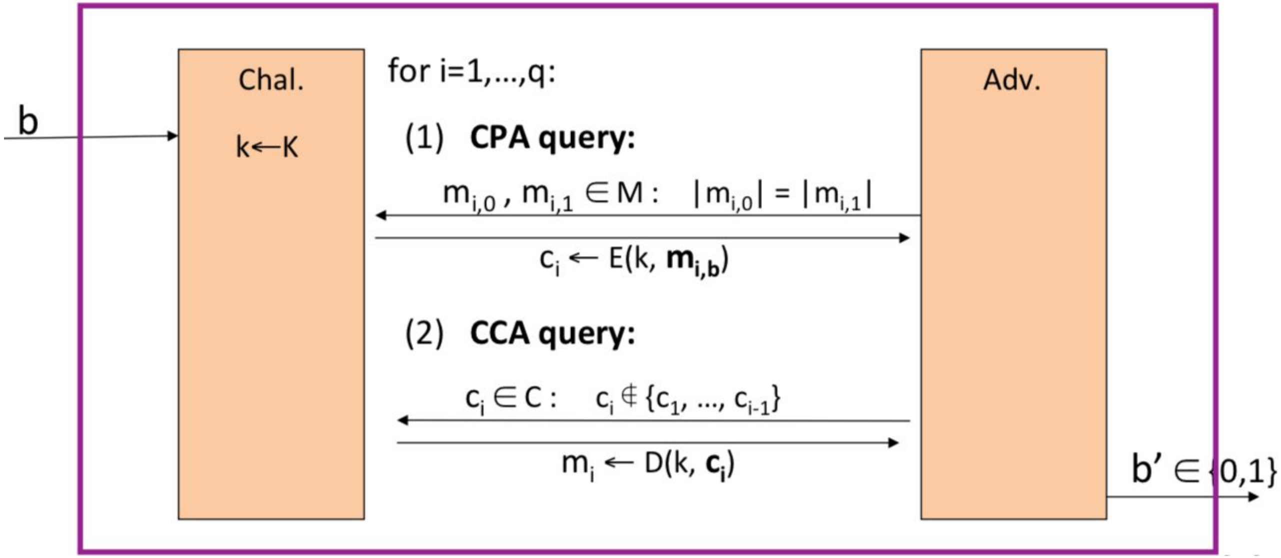
What good is CPA then?

Semantic Security / CPA provides eavesdropping protection

It does ***not*** protect against an active attacker

How can we provide that protection?

Chosen Ciphertext Attack (CCA) Security



Authenticated Encryption

Definition: An **Authenticated Encryption** system (E,D) is a cipher where:

- $E: K \times M \times N \rightarrow C$ (N optional nonce)
- $D: K \times C \times N \rightarrow M \cup \{\perp\}, \perp \notin M$ (bottom is indication that ciphertext rejected)

Security: System must provide:

- Semantic security under a CPA attack, and
- Ciphertext integrity: attacker cannot create new ciphertexts that decrypt correctly

Additional Slides (save for later)

Combining Encryption and Integrity

SSH (Encrypt-and-MAC):

1. Encrypt message, E(ke, m)
2. Append tag = S(ki, m), i.e. compute tag on plaintext
3. Output E(ke, m) || S(ki, m)

SSL (MAC-then-Encrypt):

1. Add tag = S(ki, m) to end of message
2. Encrypt whole thing, E(ke, m || tag)

IPsec (Encrypt-then-MAC):

1. Encrypt message, E(ke, m)
2. Append tag = S(ki, c), i.e. compute tag on ciphertext
3. Output E(ke, m) || S(ki, c)

Encrypt-AND-MAC

NEVER DO THIS!

MAC can leak info about PT - it was never designed for secrecy

Example of why you shouldn't make your own crypto!

https://tonyarcieri.com/all-the-crypto-code-youve-ever-written-is-probably-broken

MAC-then-Encrypt

Usually secure, but…

- Have to decrypt first to see if message is authentic
- Leads to padding oracle attacks, BEAST, others

https://codeinsecurity.wordpress.com/2013/04/05/quick-crypto-lesson-why-mac-then-encrypt-is-bad/

Encrypt-then-MAC

- Ciphertext protected by MAC
- Any tampering gets message immediately rejected
- MAC is applied after encryption, cannot leak any info

When applied correctly yields Authenticated Encryption, but...