How do we combine confidentiality and integrity?

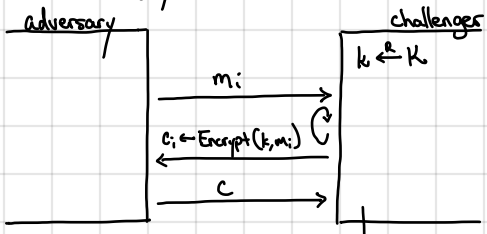
↳ Systems with both guarantees are called authenticated encryption schemes — gold standard for symmetric encryption

Two natural options:

1. Encrypt-then-MAC (TLS 1.2+, IPsec) ← guaranteed to be secure if we instantiate using CPA-secure encryption and a secure MAC
2. MAC-then-encrypt (SSL 3.0 / TLS 1.0, 802.11i) ← as we will see, not always secure

Definition. An encryption scheme $\Pi_{SE} = (\text{Encrypt}, \text{Decrypt})$ is an authenticated encryption scheme if it satisfies the following two properties:

- CPA security [confidentiality]
- ciphertext integrity [integrity]



output 1 if $c \notin \{c_1, c_2, \ldots\}$ and $\text{Decrypt}(k, c) \neq \perp$

special symbol $\perp$ to denote invalid ciphertext

Define $\text{CIAdv}[A, \Pi_{SE}]$ to be the probability that output of above experiment is 1. The scheme Π_{SE} satisfies ciphertext integrity if for all efficient adversaries A,

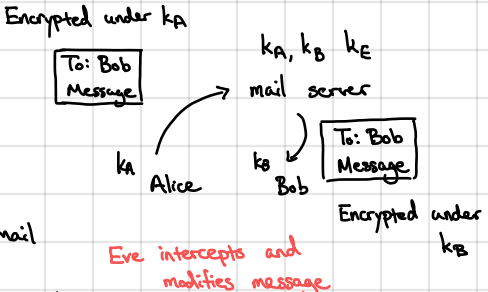
$$\text{CIAdv}[A, \Pi_{SE}] = \text{negl}(\lambda)$$

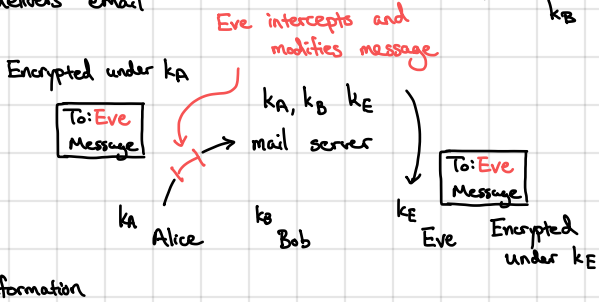
↑ security parameter determines key length

Ciphertext integrity says adversary cannot come up with a new ciphertext: only ciphertexts it can generate are those that are already valid. Why do we want this property?

Consider the following active attack scenario:

- Each user shares a key with a mail server
- To send mail, user encrypts contents and send to mail server
- Mail server decrypts the email, re-encrypts it under recipient's key and delivers email

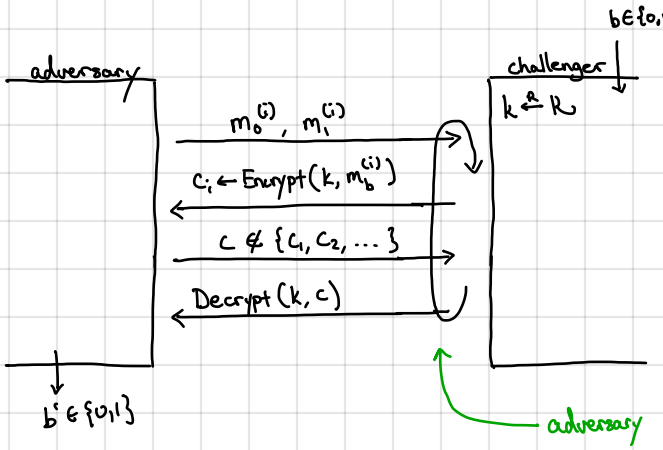




If Eve is able to tamper with the encrypted message, then she is able to learn the encrypted contents (even if the scheme is CPA-secure)

↳ More broadly, an adversary can tamper and inject ciphertexts into a system and observe the user's behavior to learn information about the decrypted values — against active attackers, we need stronger notion of security

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)
↳ called an "admissibility" criterion

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

CCA-security captures above attack scenario where adversary can tamper with ciphertexts
- ↳ Rules out possibility of transforming encryption of $x \| z$ to encryption of $y \| z$
- ↳ Necessary for security against <u>active</u> adversaries [CPA-security is for security against <u>passive</u> adversaries]
- ↳ 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 $\not\Rightarrow$ ciphertext integrity.
- ↳ However, CCA-security + plaintext integrity $\Rightarrow$ authenticated encryption

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

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

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

(independent keys)

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