CSE 484/M584: Computer Security (and Privacy)

Spring 2025

David Kohlbrenner
dkohlbre@cs

UW Instruction Team: David Kohlbrenner, Yoshi Kohno, Franziska Roesner, Nirvan Tyagi. Thanks to Dan Boneh, Dieter Gollmann, Dan Halperin, John Manferdelli, John Mitchell, Vitaly Shmatikov, Bennet Yee, and many others for sample slides and materials

Admin

- Lab 2 (Cryptolab) next Wednesday
- Lab 1a/b Exploits
  - Again, check partner status. Please.
  - Partner status is *per-submission*. You have to do it each time.
  - Grades out.

Person-in-the-Middle Attacks

- Diffie-Hellman protocol (by itself) does not provide integrity (against <u>active</u> attackers)

$sk_A \leftarrow x$
$pk_A \leftarrow g^x$

$sk_B \leftarrow y$
$pk_B \leftarrow g^y$

Public Key Encryption

Alice

pk_B

Bob

pk_B, sk_B

Public Key Encryption

Alice

pk_B

ct

Bob

pk_B, sk_B

$\text{Encrypt}(pk_B , m) \rightarrow ct$

Public Key Encryption

Alice

pk_B

$\xrightarrow{\text{ct}}$

Bob

pk_B, sk_B

$\text{Encrypt}(\text{pk}_B, m) \rightarrow \text{ct}$

$\text{Decrypt}(\text{sk}_B, \text{ct}) \rightarrow m$

Public Key Encryption from Diffie-Hellman

pk_B

Alice

pk_B, sk_B

$sk_B \leftarrow y$
$pk_B \leftarrow g^y$

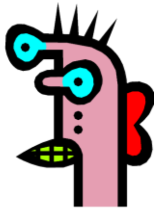
Bob

Public Key Encryption from Diffie-Hellman

pk_B

Alice

pk_B, sk_B



Bob

$sk_B \leftarrow y$
$pk_B \leftarrow g^y$

<u>Sample one-time key:</u>

$sk_A \leftarrow r$
$pk_A \leftarrow g^r$

<u>Compute DH shared secret:</u>

$K = H(g^{ry})$

<u>Encrypt with authenticated symmetric encryption:</u>

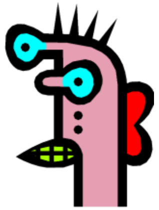
$ct_{SE} = SE.Enc(K, m)$

Public Key Encryption from Diffie-Hellman

pk_B

Alice

pk_B, sk_B



Bob

$sk_B \leftarrow y$
$pk_B \leftarrow g^y$

<u>Sample one-time key:</u>
$sk_A \leftarrow r$
$pk_A \leftarrow g^r$

$ct = (g^r, ct_{SE})$ $\longrightarrow$

<u>Compute DH shared secret:</u>
$K = H(g^{ry})$

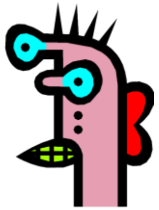
<u>Encrypt with authenticated symmetric encryption:</u>
$ct_{SE} = SE.Enc(K, m)$

Public Key Encryption from Diffie-Hellman

pk_B

Alice

pk_B, sk_B



Bob

$sk_B \leftarrow y$
$pk_B \leftarrow g^y$

<u>Sample one-time key:</u>

$sk_A \leftarrow r$
$pk_A \leftarrow g^r$

$$ct = (g^r, ct_{SE})$$

<u>Compute DH shared secret:</u>

$K = H(g^{ry})$

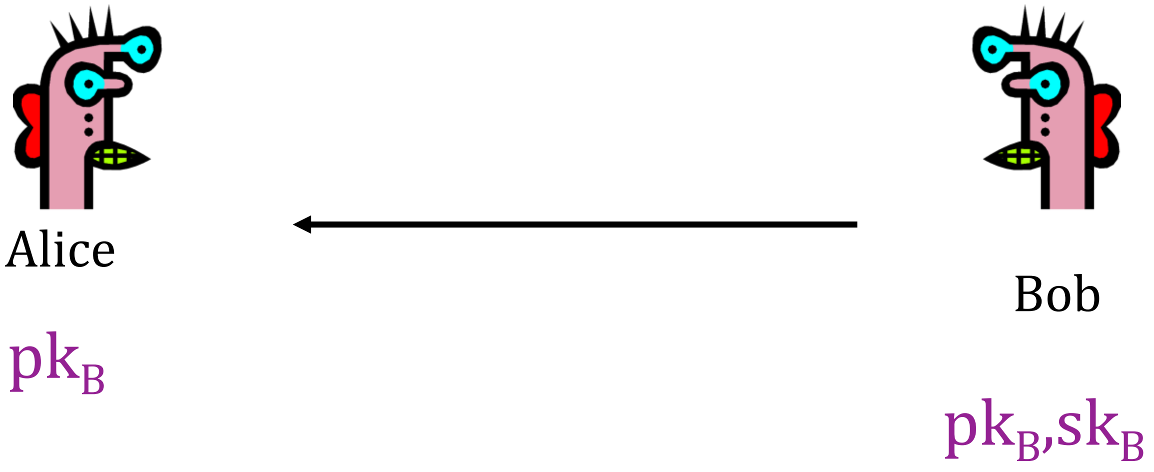
<u>Encrypt with authenticated symmetric encryption:</u>

$ct_{SE} = SE.Enc(K, m)$

$K = H(g^{ry})$
$m = SE.Dec(K, ct_{SE})$

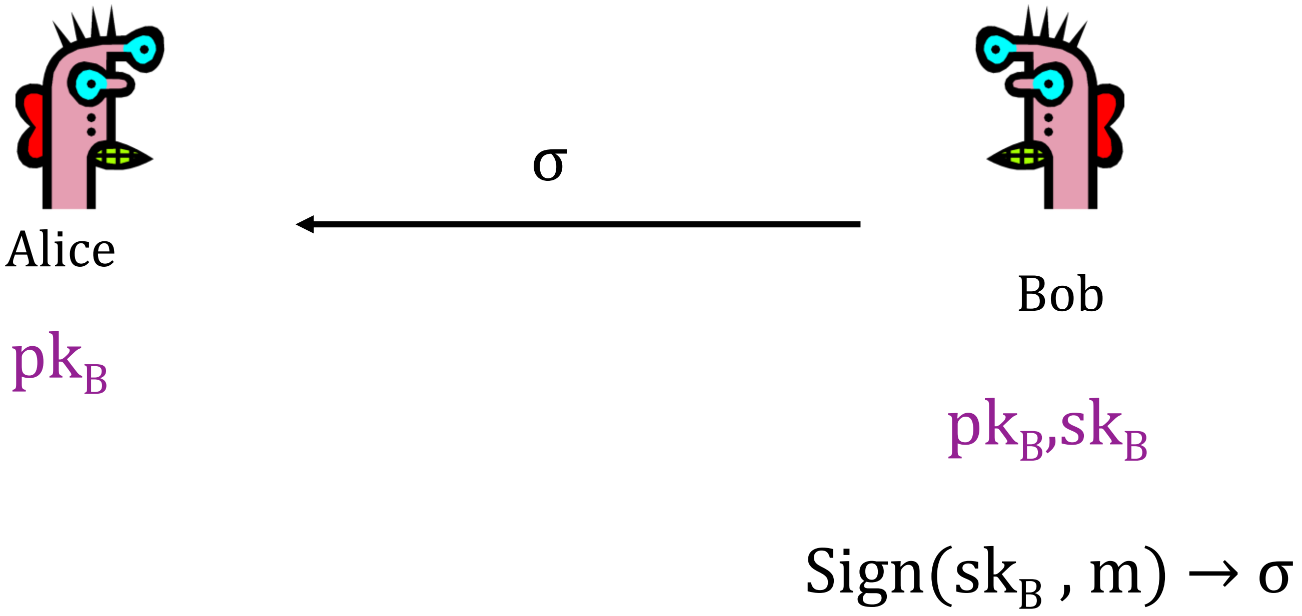
Digital Signatures

- No one should be able to forge signatures from Bob's public key without Bob's secret key



Digital Signatures

- No one should be able to forge signatures from Bob's public key without Bob's secret key



Digital Signatures

- No one should be able to forge signatures from Bob's public key without Bob's secret key

Alice

pk_B

σ

$\longleftarrow$

Bob

pk_B, sk_B

$\mathrm{Verify}(pk_B, m, \sigma) \to 0/1$

$\mathrm{Sign}(sk_B, m) \to \sigma$

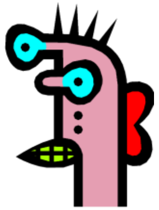
Digital Signatures

- No one should be able to forge signatures from Bob’s public key without Bob’s secret key

Alice

σ

$\longleftarrow$



Bob

pk_B

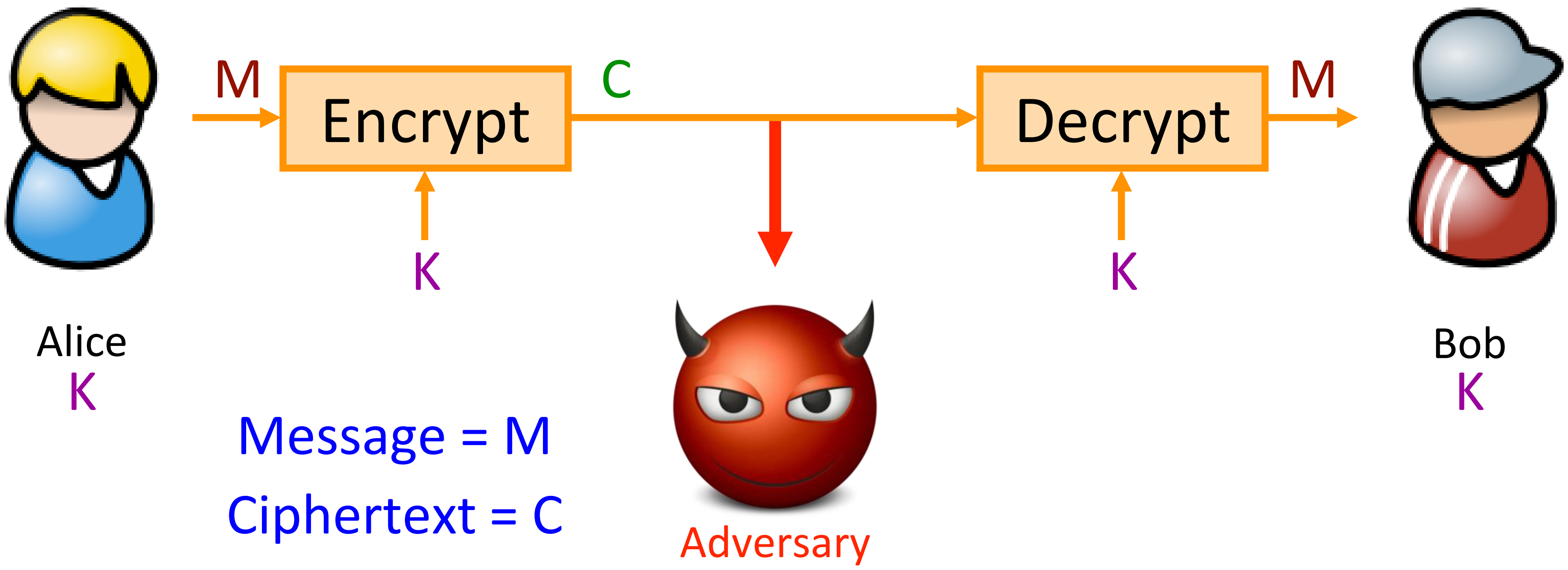
pk_B, sk_B

$\mathrm{Verify}(pk_B, m, \sigma) \rightarrow 0/1$

$\mathrm{Sign}(sk_B, m) \rightarrow \sigma$

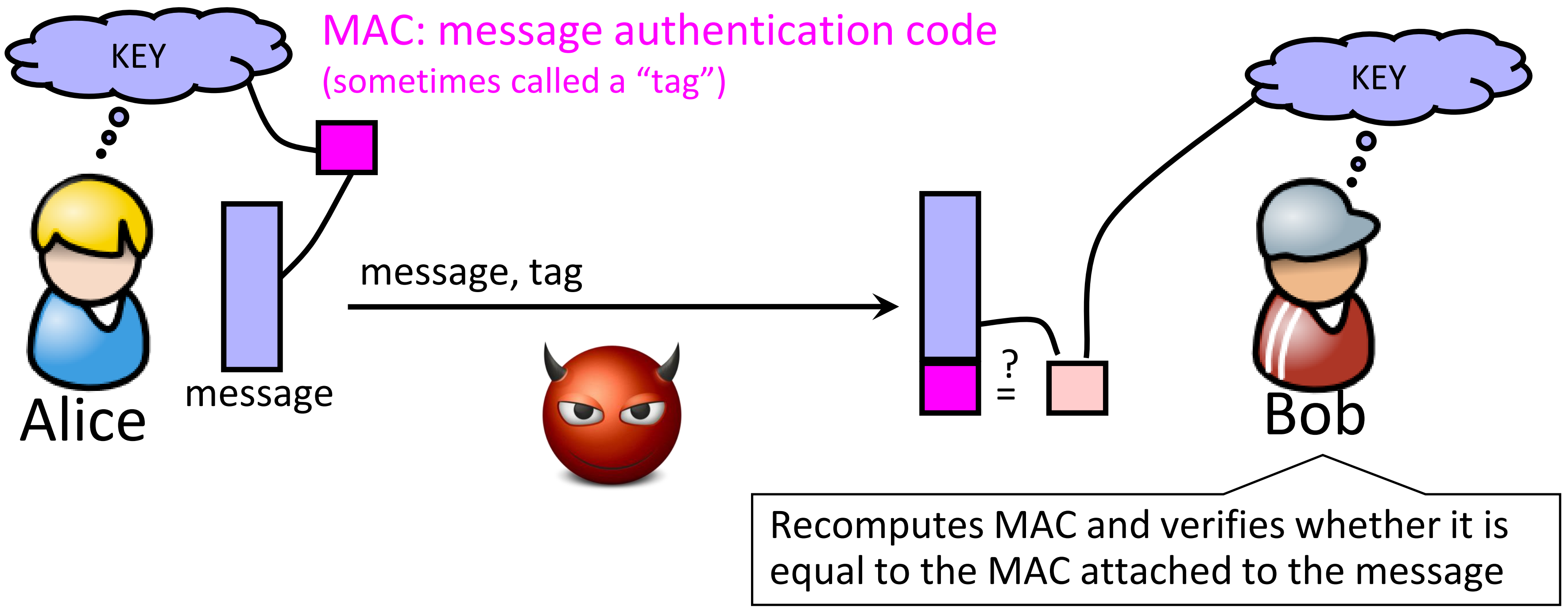
So Far: Achieving Confidentiality/Authenticity

Encryption schemes: A tool for protecting confidentiality.



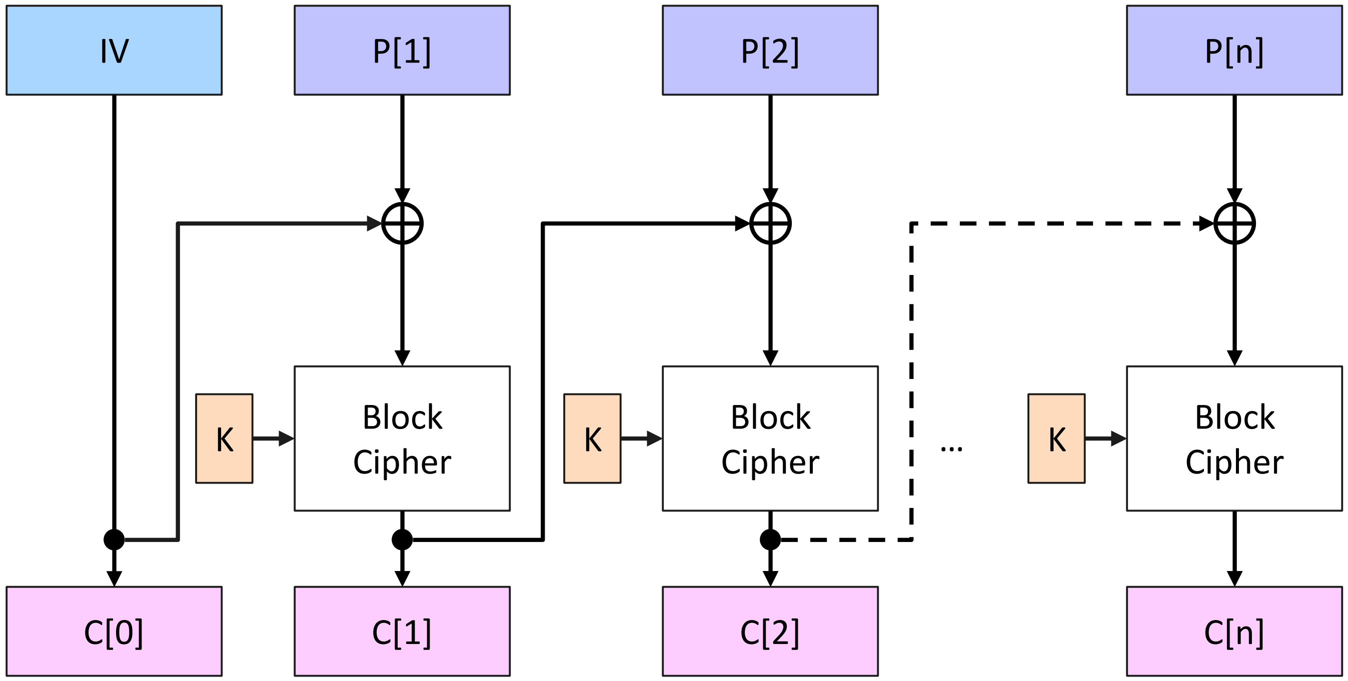
Now: Achieving Integrity

Message authentication schemes: A tool for protecting integrity.

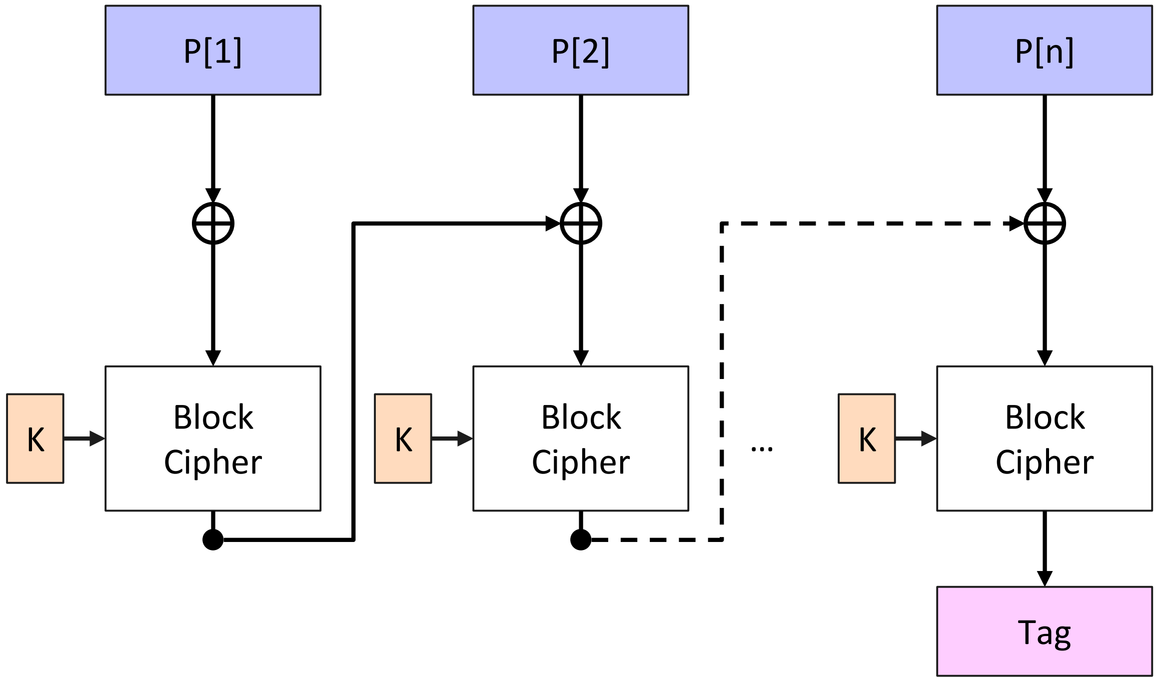


Integrity and authentication: only someone who knows KEY can compute correct MAC for a given message.

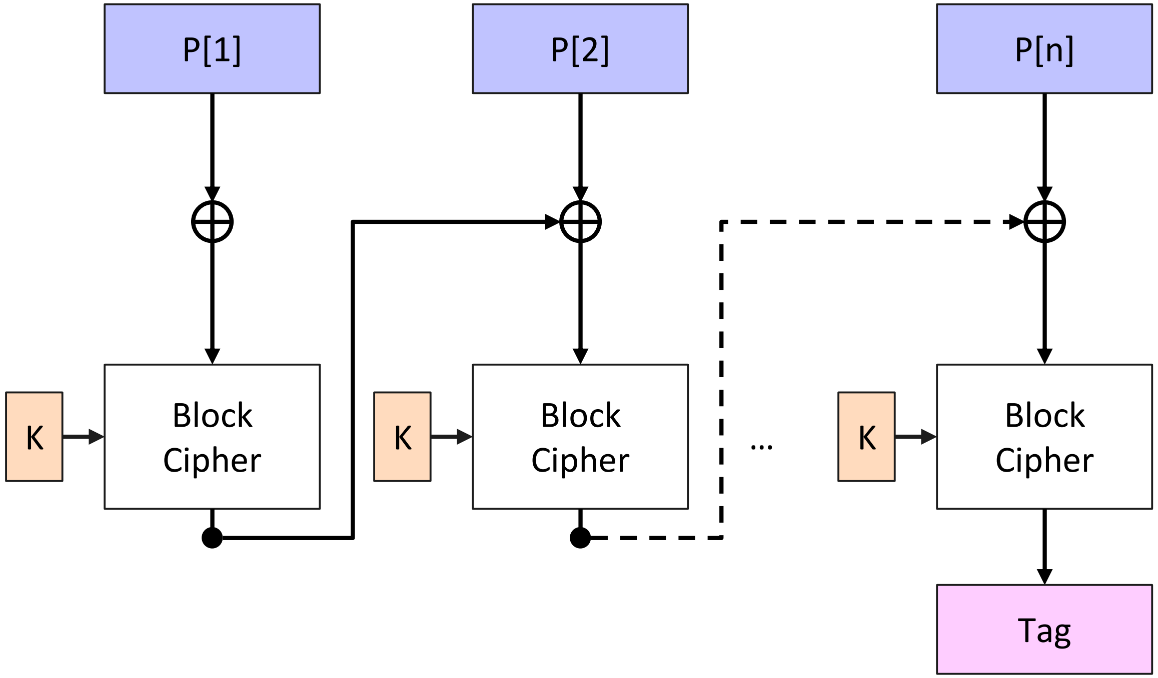
MAC from CBC Mode (CBC-MAC)



MAC from CBC Mode (CBC-MAC)

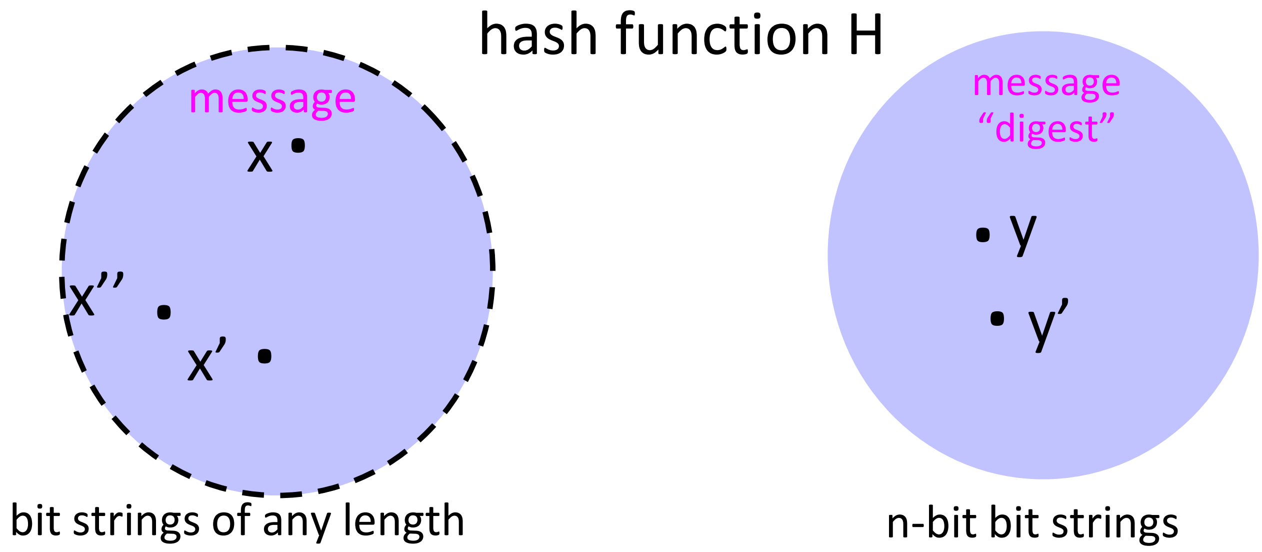


MAC from CBC Mode (CBC-MAC)



- Not secure when system may MAC messages of different lengths
- Adapt by concatenating message length to front of plaintext

Hash Functions: Main Idea



- Hash function H is a lossy compression function
  - Collision: h(x)=h(x') for distinct inputs x, x'
- Cryptographic hash function needs a few properties…

Hash Functions: Useful!

- Distributing software
- Checking integrity of files
- Hashtables
- Commitments
- Etc.

Property 1: One-Way

- Intuition: hash should be hard to invert
  - "Preimage resistance"
  - Let $h(x') = y \in \{0,1\}^n$ for a random x'
  - Given y, it should be hard to find any x such that h(x)=y
- How hard?
  - Brute-force: try every possible x, see if h(x)=y
  - SHA-2 (common hash function) has 256-bit output
    - Expect to try 2^{255} inputs before finding one that hashes to y.

Property 2: Collision Resistance

- Should be hard to find $x \neq x'$ such that $h(x)=h(x')$

Birthday Paradox

In a class with q students, what is the probability that two of them have the same birthday? [Assuming birthdays are uniform!]

- $\mathcal{S} = \{\text{Jan } 1, \dots, \text{Dec } 31\}$, $|\mathcal{S}| = 365$ [ignore leap year]

Birthday Paradox

In a class with q students, what is the probability that two of them have the same birthday? [Assuming birthdays are uniform!]

- $\mathcal{S} = \{\text{Jan } 1, \dots, \text{Dec } 31\}$, $|\mathcal{S}| = 365$ [ignore leap year]
- For $q = 23$: $0.500001 \ldots \leq p \leq 0.69315 \ldots$

Theorem. We have

$$1 - e^{-\frac{q(q-1)}{2|\mathcal{S}|}} \leq \mathrm{p}_{coll}(q, \mathcal{S}) \leq \frac{q(q-1)}{2|\mathcal{S}|}$$

Note: For $q = \sqrt{|\mathcal{S}|}$ we have $0.39 \leq \mathrm{p}_{coll}(q, \mathcal{S}) \leq 0.5$

Birthday Paradox

- Why is the birthday paradox important for collision resistance?
  - 2^{128} different 128-bit values
    - Pick one value at random. To exhaustively search for this value requires trying on average 2^{127} values.
    - **Expect "collision" after selecting approximately 2^{64} random values.**
    - **64 bits** of security against collision attacks, not 128 bits.

- Should be hard to find $x \neq x'$ such that $h(x)=h(x')$
- Birthday paradox means that brute-force collision search is only $O(2^{n/2})$, *not* $O(2^n)$
  - For SHA-2 with 256-bit output, this means $O(2^{128})$ vs. $O(2^{256})$

Property 3: Indifferentiability

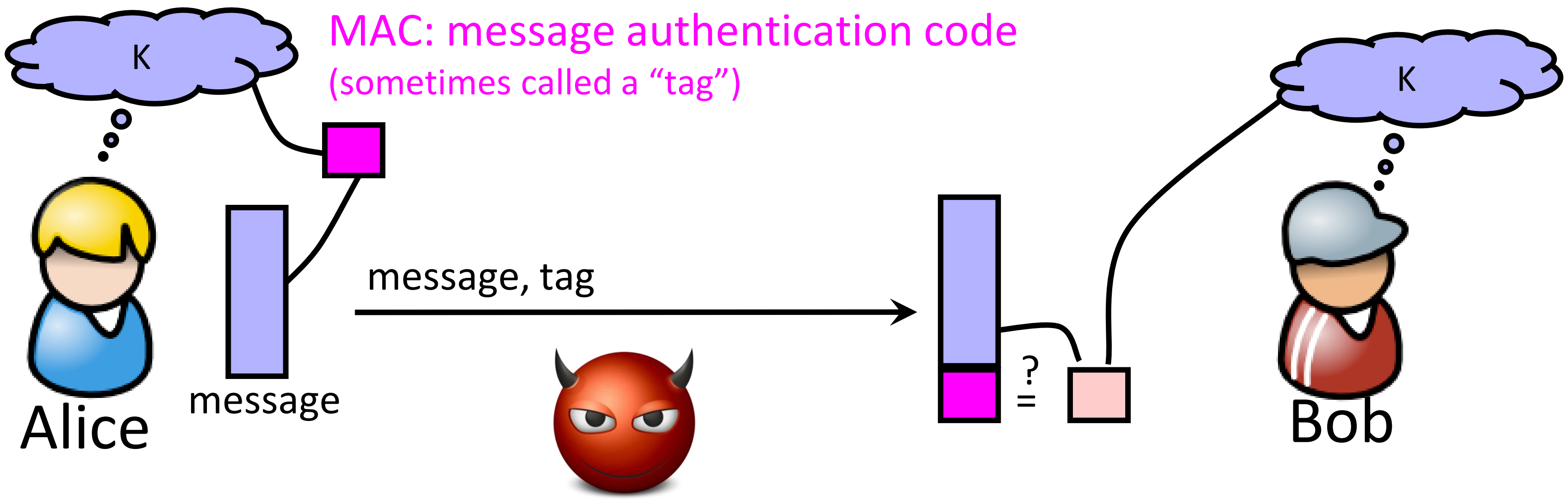
- Informal: Outputs of the hash function look “random” (in a certain ideal model)

Hashing vs. Encryption

- Hashing is one-way. There is no “un-hashing” (one-way)
  - A ciphertext can be decrypted with a decryption key
- Hashing is deterministic
  - Hash the same input twice => same hash value
  - Encrypt the same input twice => different ciphertexts

MAC via Hashing

Message authentication schemes: A tool for protecting integrity.



Tag = Hash(K || message)

Common Hash Functions

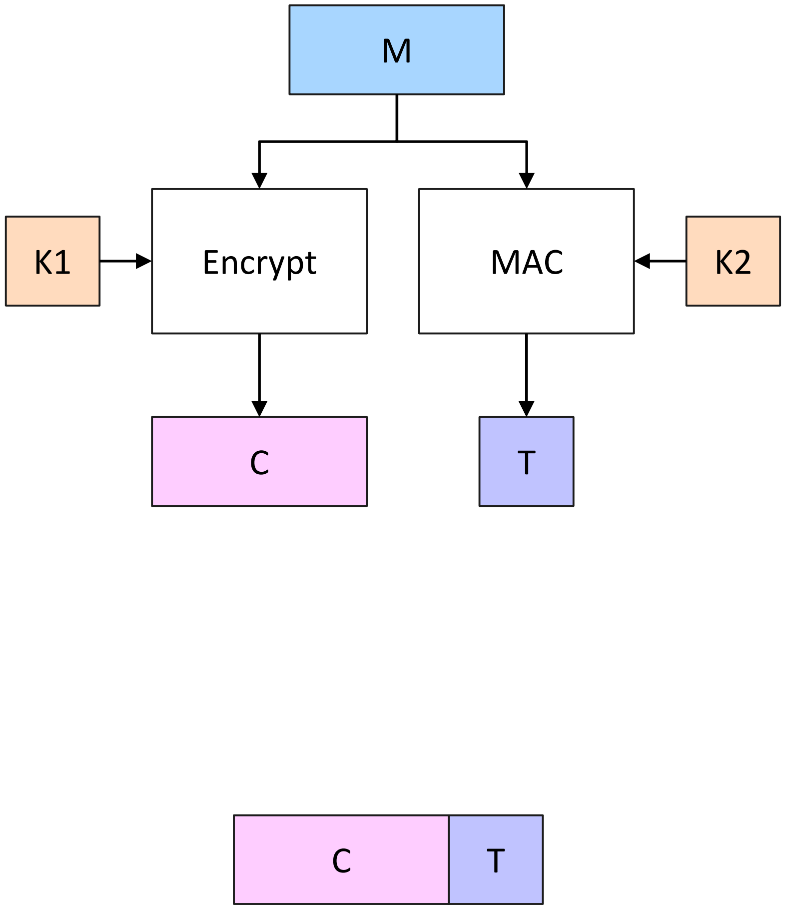
- **SHA-2: SHA-256, SHA-512, SHA-224, SHA-384**
- **SHA-3: standard released by NIST in August 2015**
- MD5 – Don’t use for security!
  - 128-bit output
  - Collision-resistance broken (summer of 2004)
- SHA-1 (Secure Hash Algorithm) – Don’t use for security!
  - 160-bit output
  - US government (NIST) standard as of 1993-95
  - Theoretically broken 2005; practical attack 2017!

Authenticated Encryption

- What if we want both confidentiality and integrity?
- Natural approach: combine encryption scheme and a MAC.

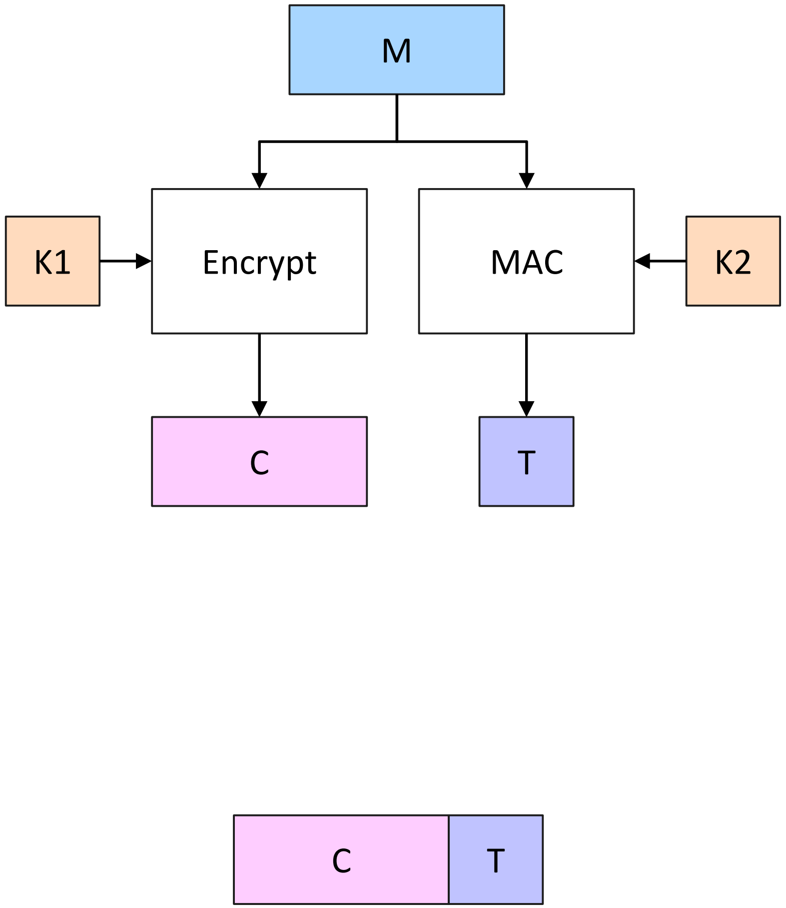
How to combine Encryption and MACs?

Encrypt-and-MAC

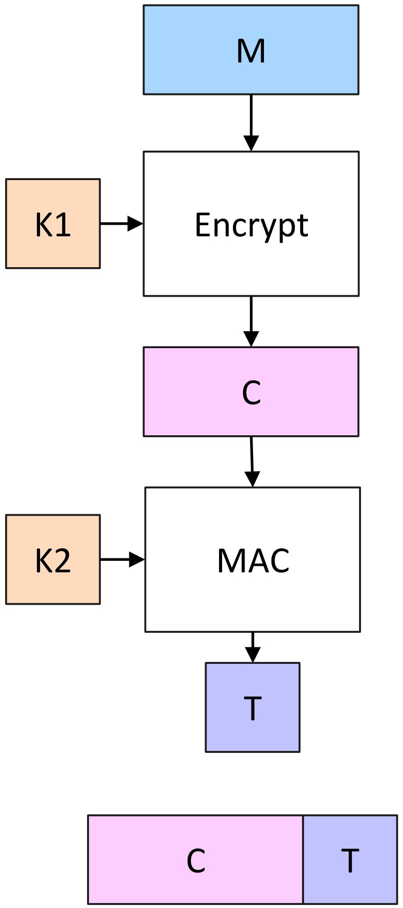


How to combine Encryption and MACs?

Encrypt-and-MAC



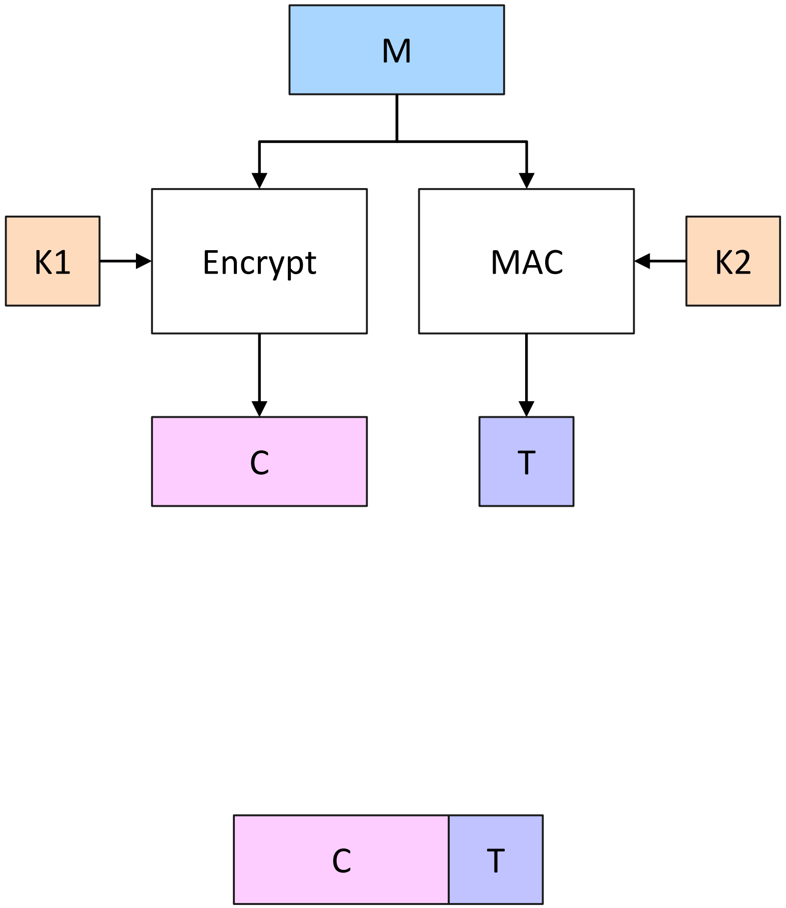
Encrypt-then-MAC



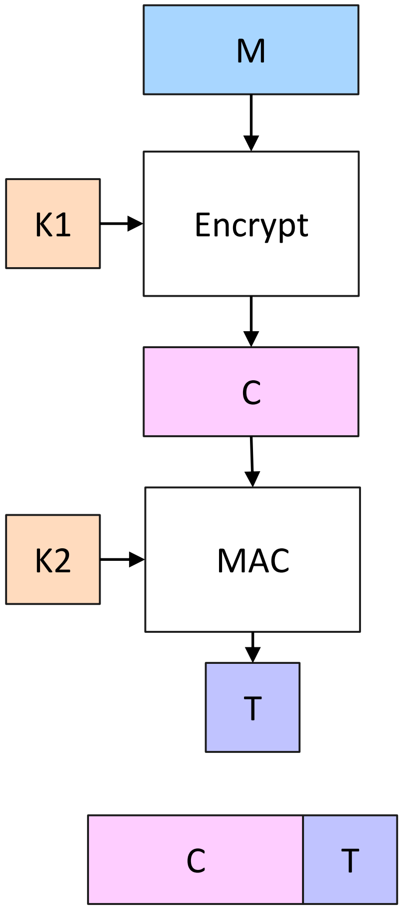
Gradescope!

How to combine Encryption and MACs?

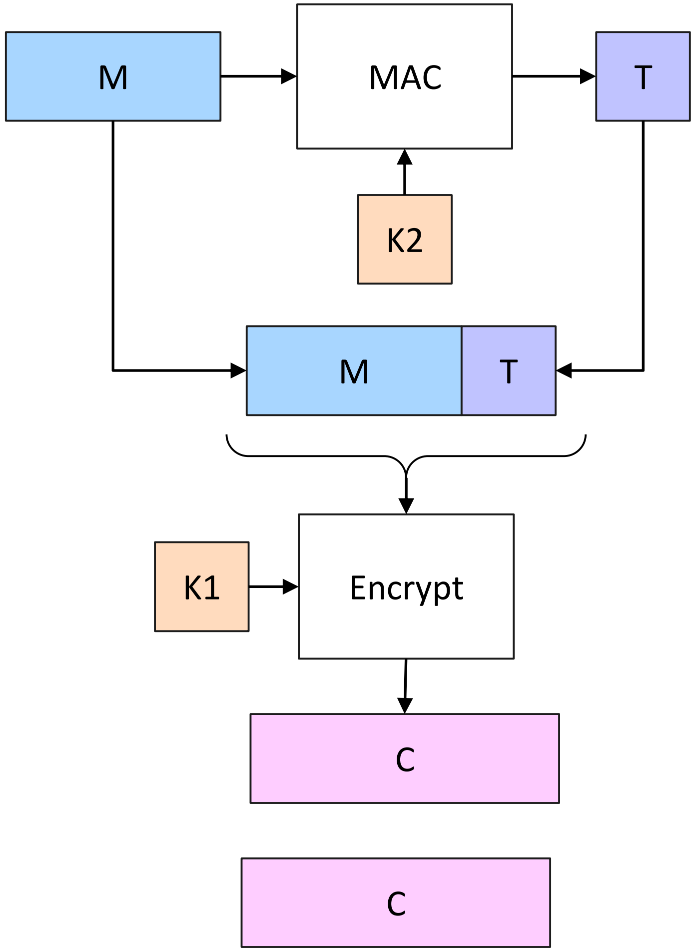
Encrypt-and-MAC



Encrypt-then-MAC

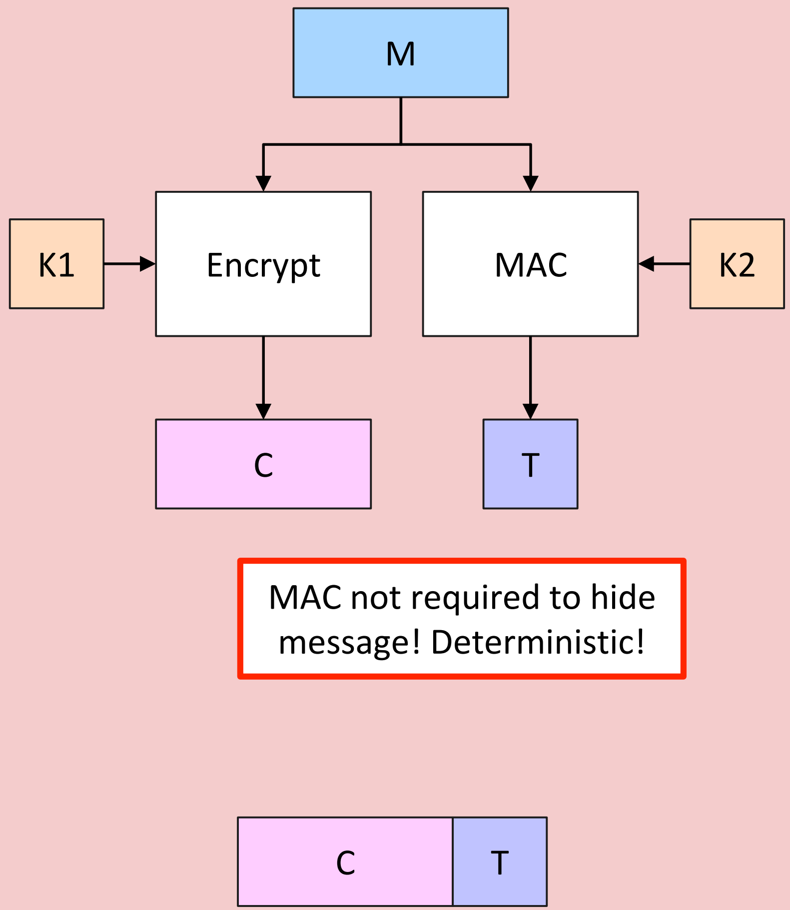


MAC-then-Encrypt

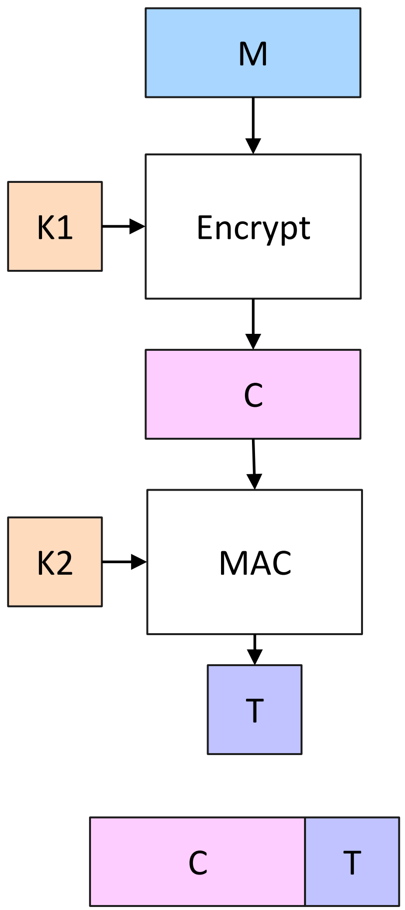


How to combine Encryption and MACs?

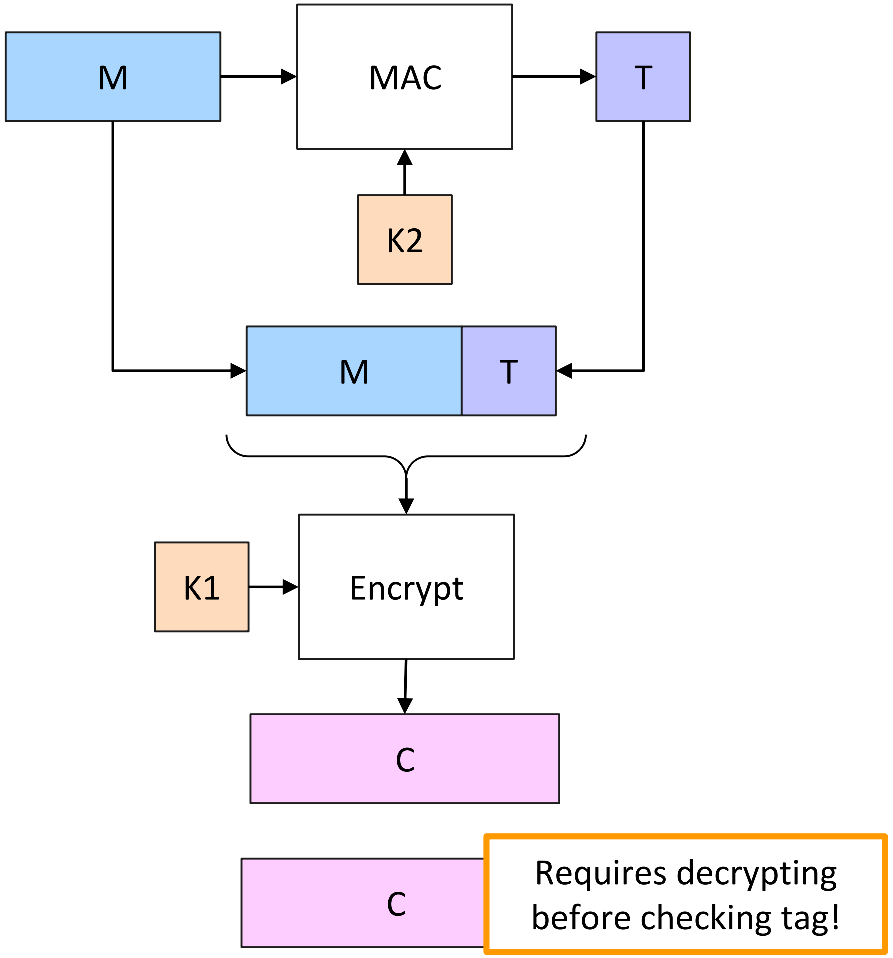
Encrypt-and-MAC



Encrypt-then-MAC



MAC-then-Encrypt



What do Quantum Computers mean for Cryptography?

What do Quantum Computers mean for Cryptography?

1. Implications for existing cryptography
    - Quantum algorithms exist to solve “hard” assumptions quickly
        - Shor’s algorithm can solve factoring and discrete logarithm
    - “Post-quantum” cryptography
        - Build asymmetric cryptography for classical computers based on assumptions that we think are “hard” even for quantum computers
        - “Lattice-based” cryptography

What do Quantum Computers mean for Cryptography?

1. Implications for existing cryptography
   - Quantum algorithms exist to solve “hard” assumptions quickly
     - Shor’s algorithm can solve factoring and discrete logarithm
   - “Post-quantum” cryptography
     - Build asymmetric cryptography for classical computers based on assumptions that we think are “hard” even for quantum computers
     - “Lattice-based” cryptography
2. Implications for future cryptography
   - Quantum computing offers new hardness assumptions and new functionality from which to build cryptography