Computer and Network Security: Integrity and Authentication

Kameswari Chebrolu

Outline

- **Modern Cryptography**
  - Overview
  - Confidentiality
    - Background: Definition, Crypto-analysis, One Time Pads
    - Symmetric key encryption, Block modes
    - Asymmetric key encryption
  - **Integrity (includes Authentication)**
    - Hashes, **MAC**, Digital signature

Recap: Integrity and Authentication

- Focus: Communication framework

Definition:

- Integrity (data): Has the data been modified in transit?
- Authentication: Am I talking with the right person?

- ~~Hashes/Message Digests: Integrity~~
- **Message Authentication Codes (MACs): Integrity and Authentication**
  - **Based on Shared key**
- Digital Signatures: Integrity and Authentication
  - Based on Asymmetric key algorithm

Message Authentication Codes (MACs)

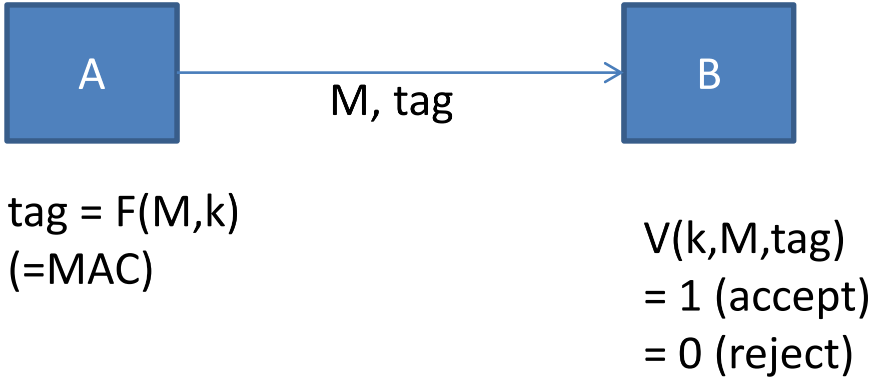
- Also referred to as keyed hash function
  - A secret key is needed for evaluation
- Provides both Integrity and Authentication
  - Only some one with identical key can verify hash
  - Does not provide confidentiality
  - Require similar properties as hash functions (pre-image resistance, weak/strong collision resistance)

MAC vs Hash

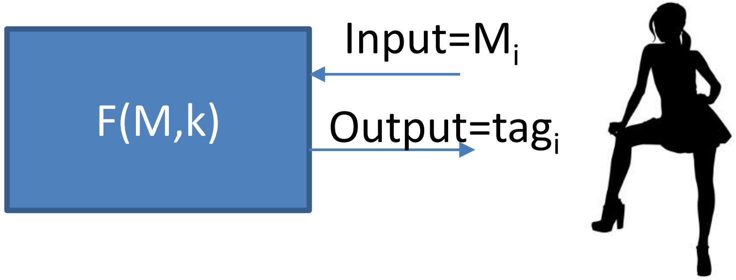
- A virus can modify a file and its hash also (recalculate) → cannot detect tampering
- Virus can modify file but cannot calculate new MAC since it does not know the secret key

Details

- M: message, k: secret key shared between A and B
- A sends message and tag
- B verifies received message with tag
  - Matches, accept (authentic + untampered)
  - No match, reject (tampered/unauthentic or corrupted)



Security Model



Attacker does not know k

Attacker can input *any* messages M_1, …, M_n of its choice and get corresponding tag

Attacker succeeds if it outputs a forgery;
i.e., (M, tag)

$M \neq M_i$ for all i

V(k,M,tag) = 1

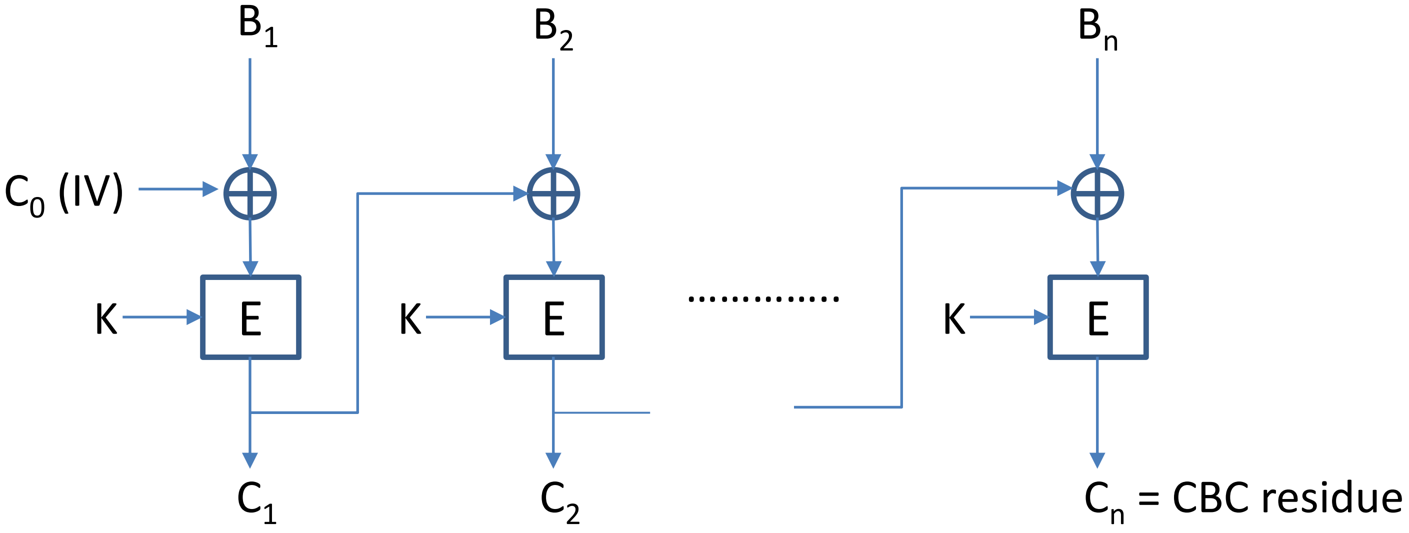
Want Pr[winning] ~ 0 (time bound)

Two Types (Popular)

- Block Cipher based (e.g. CBC mode)
- Hash based (but with key)
  - Faster
  - Code more readily available

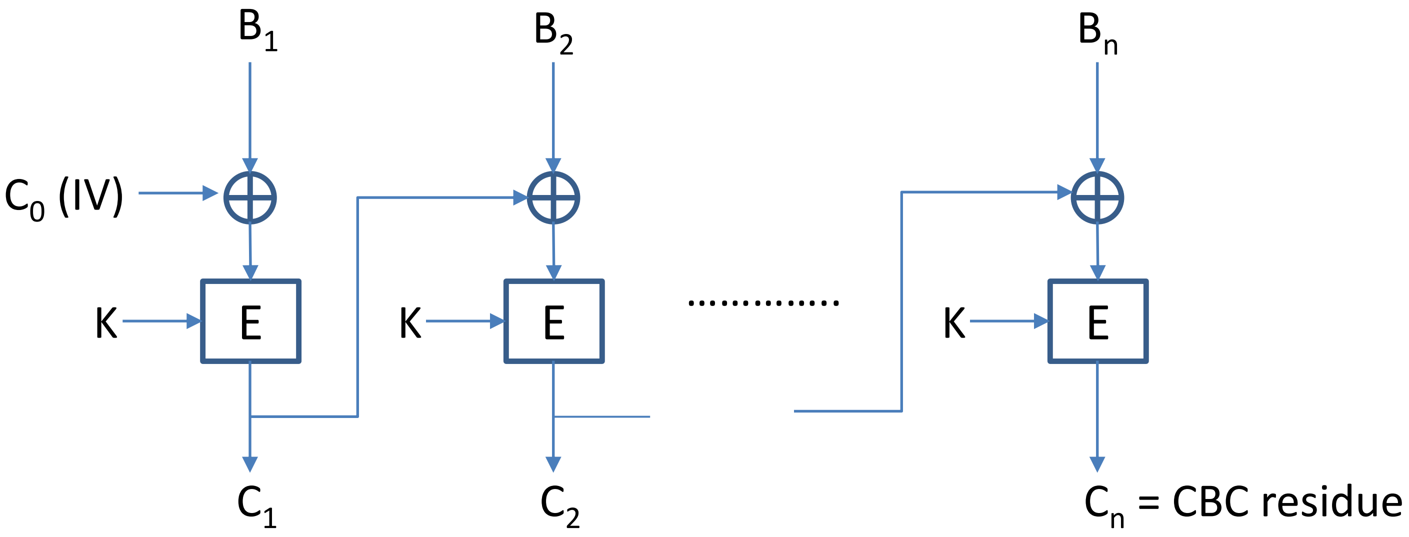
MAC Construction

- Use secret key algorithms to compute MAC
- Remember Cipher Block Chaining (CBC)?
- Last block ciphertext of CBC is CBC residue (=MAC)
- Send <M, CBC residue>



MAC Construction

- Send <M, CBC residue>
- Modify M, CBC residue invalid
  - Except 1 in 2^{64} times (CBC residue = 64 bits)
- Works well if M is not to be kept secret



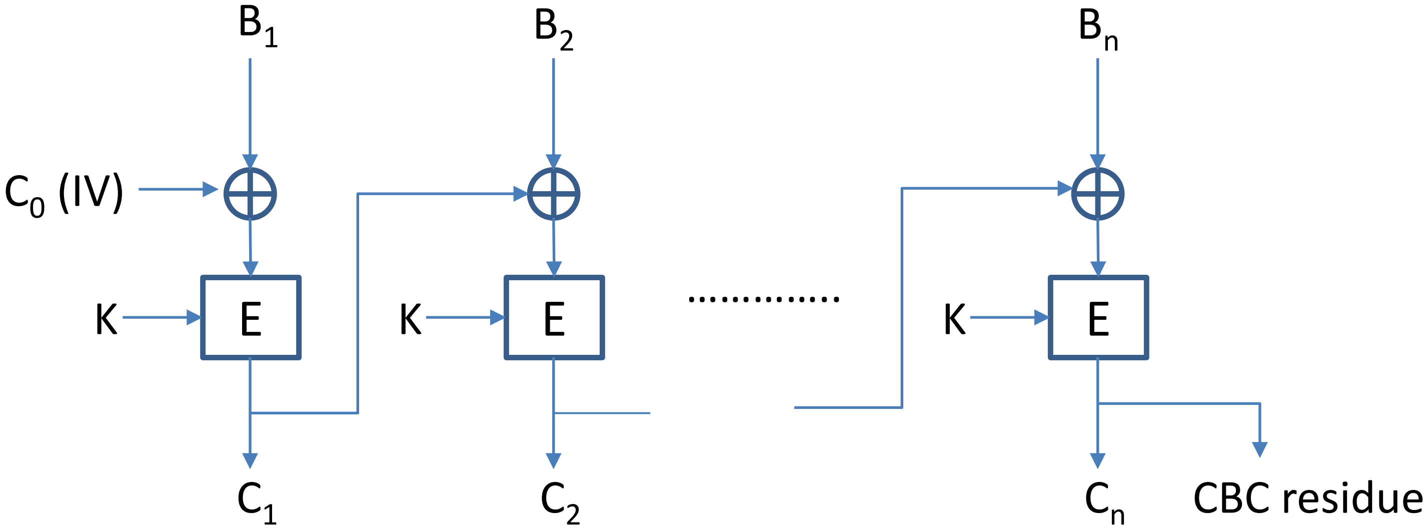
Subtleties

Message M

- Ensure confidentiality → CBC encyrpt M; Send C (Ciphertext)
  - Cannot detect tampering, especially by machines
- Ensure integrity + authentication → Send <M, CBC residue>
  - Message M is in the open
- How to ensure both confidentiality and integrity and authentication? Need attention to details!

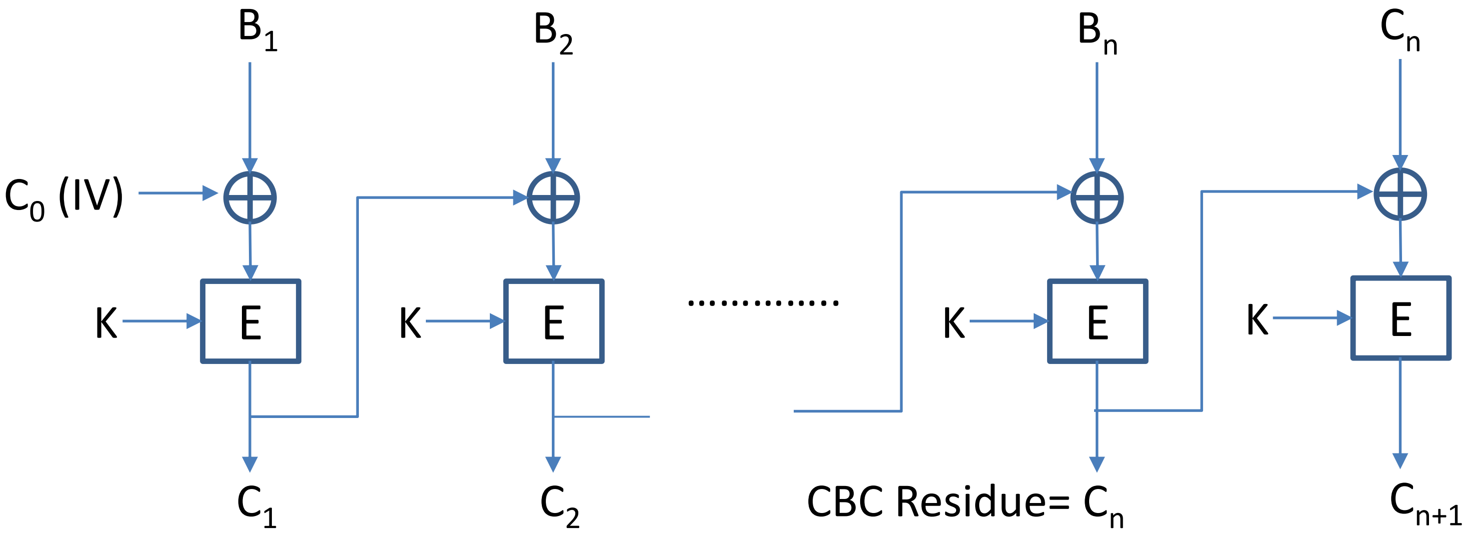
Version-1:

- Send <C, CBC residue>
- Does not work. Why?
  - Attacker can modify C and set CBC residue to last block of C
  - Some parts may decrypt to garbage in text but may be ok in some situations



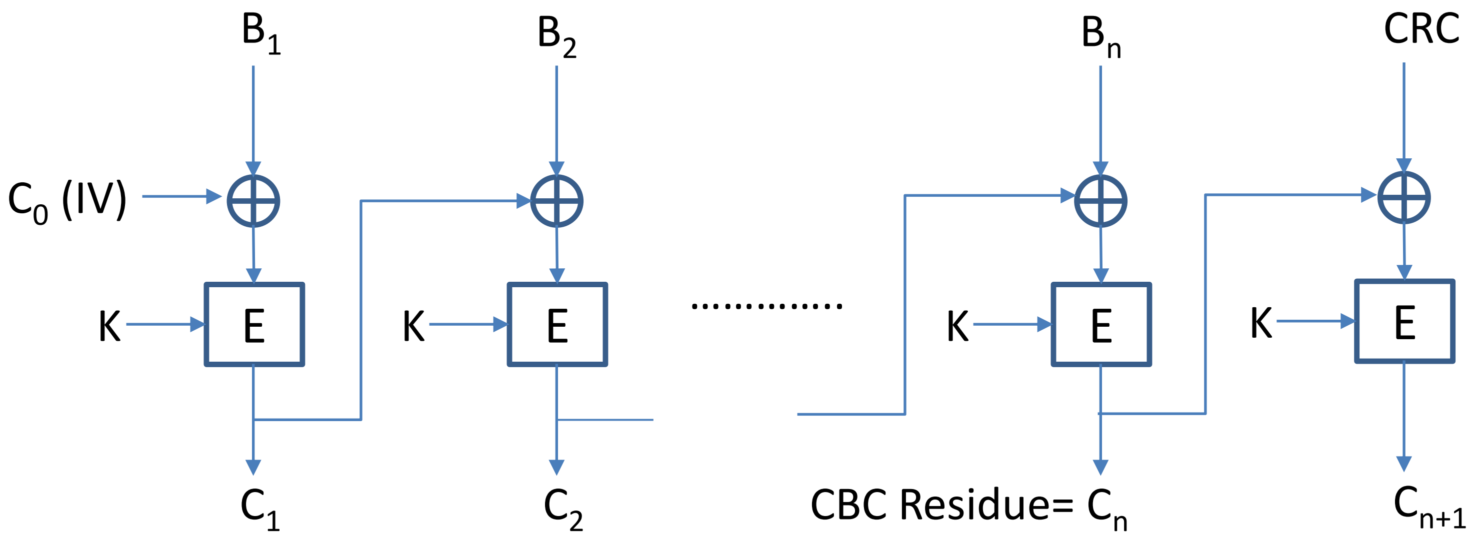
Version-2

- CBC encrypt <M|CBC residue>
  - | means concatenation
- Does not work. Why?
  - Last block always encrypts zero



Version-3

- CBC encrypt (M | CRC)
  - CRC: checksum on the message
  - May work but suspect
- Replace CRC with cryptographic hash of M
  - Requires two cryptographic passes (one for hash, one for CBC)
  - Stronger but has not received much attention or used much



Version-4

- Send <C, CBC residue>
- But use two keys
  - k1 for CBC residue of M
  - k2 for CBC encryption of M
  - K1 and k2 can be related to each other
- Requires two cryptographic passes

Two Types (Popular)

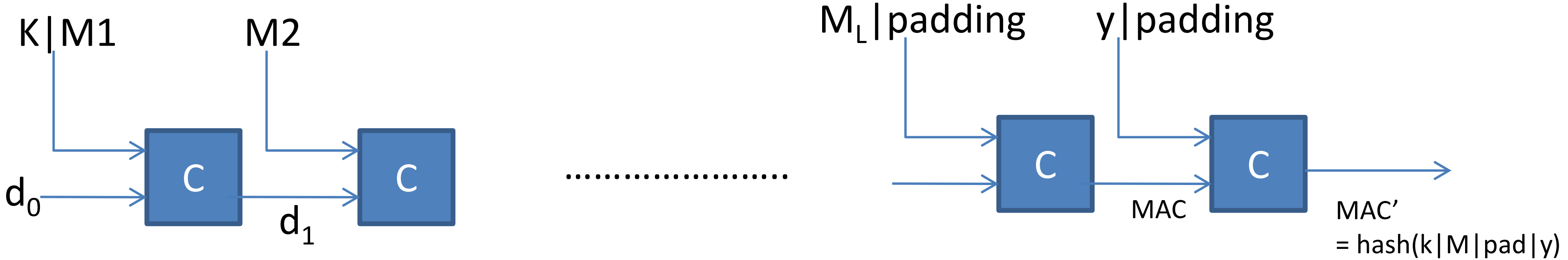
- ~~Block Cipher based (e.g. CBC mode)~~
- Hash based (but with key)
  - Faster
  - Code more readily available

MAC with a Hash

- Refresh: Mac for integrity and authentication
  - Send <M, CBC residue>
  - CBC residue based on secret key algorithm
- Secret key but no secret key algorithm (e.g DES, AES)
- Many subtleties!

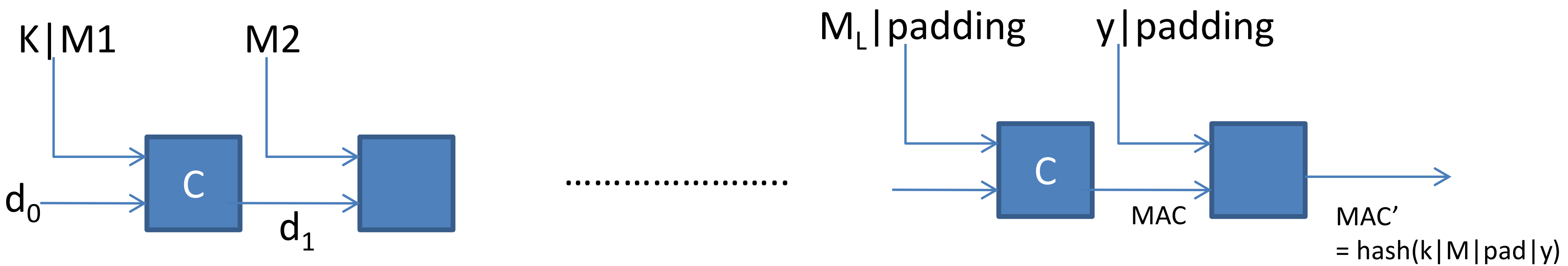
Version-1: Secret Prefix method

- Send hash(k|M) (=MAC)
- Insecure: If attacker knows M and hash(k|M); he can construct MAC of a longer message i.e. hash(k|M|y) for message M|y



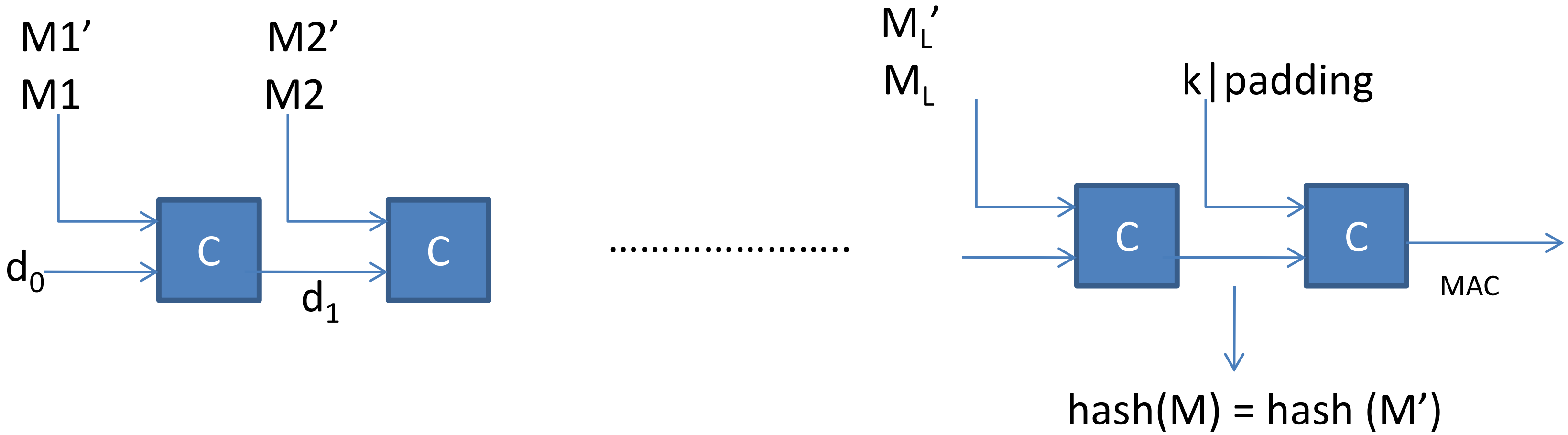
Version-2: Secret Prefix, half MAC

- Send half the bits of MAC =hash(k|M)
  - E.g. lower order 64 bits of a 128 bit digest
  - Cannot continue due to partial information



Version-3: Secret Suffix Method

- Send hash(M|k) (=MAC)
- Fixes problem with prefix method. Why?
- Insecure if hash is not collision resistant


M1'
M1
M2'
M2
d_0
C
d_1
C
....
M_L'
M_L
k|padding
C
C
MAC
hash(M) = hash (M')

Version-4: Envelope Method

- Send hash(k|M|k)
  - Appending front provides collision resistance
  - Appending back provides protection from extending message
- HMAC: More widely used (e.g. in IPsec, SSL)
  - Complex but roughly: hash(k | hash(k|M))

Integrity+Confidentiality

- Many possibilities
- More Popular
  - MAC-then-Encrypt: $E_k(M|MAC(M))$ (E.g. TLS)
  - Encrypt-and-MAC: $E_k(M)|MAC(M)$ (E.g. SSH)
  - **Encrypt-then-MAC: Ek(M)|MAC(Ek(M))** (E.g. IPsec)
- Encrypt-then-MAC: reaches the highest definition of security

Summary

- MACs based on ‘key’ provide both integrity and authentication
- Two types: block cipher and hashes
- Usage: Attention to detail important
- To provide both confidentiality and integrity, use “Encrypt-then-MAC”