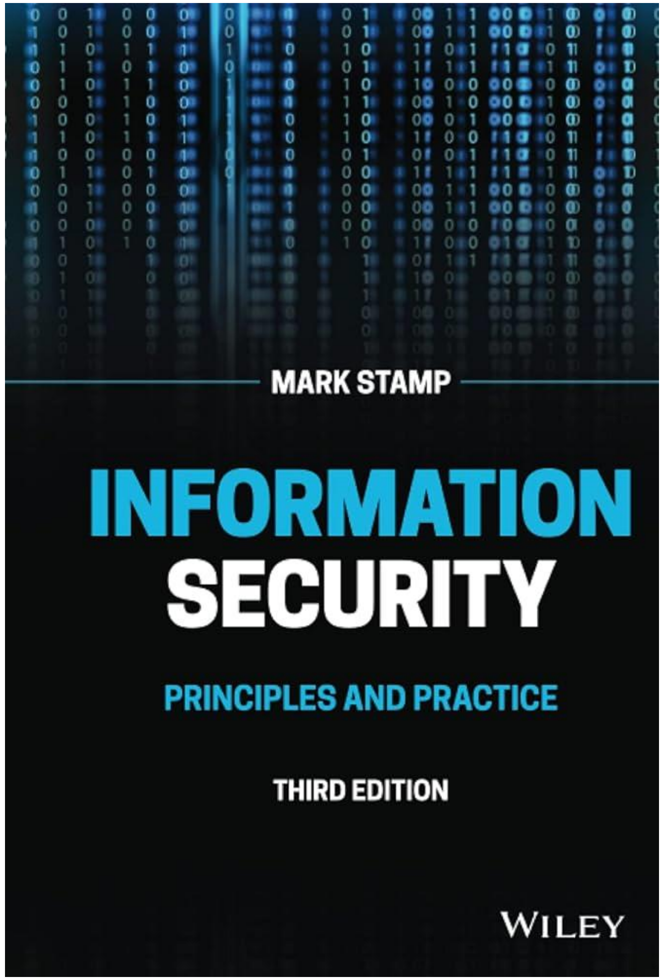
بسم الله الرحمن الرحیم

مبانی رایانش امن

جلسه ۷

مجتبی خلیلی
دانشکده برق و کامپیوتر
دانشگاه صنعتی اصفهان

- فصل سوم کتاب


MARK STAMP
INFORMATION
SECURITY
PRINCIPLES AND PRACTICE
THIRD EDITION
WILEY

MAC

- نیاز به اشتراک گذاری کلید دارد.

Alice

m, T

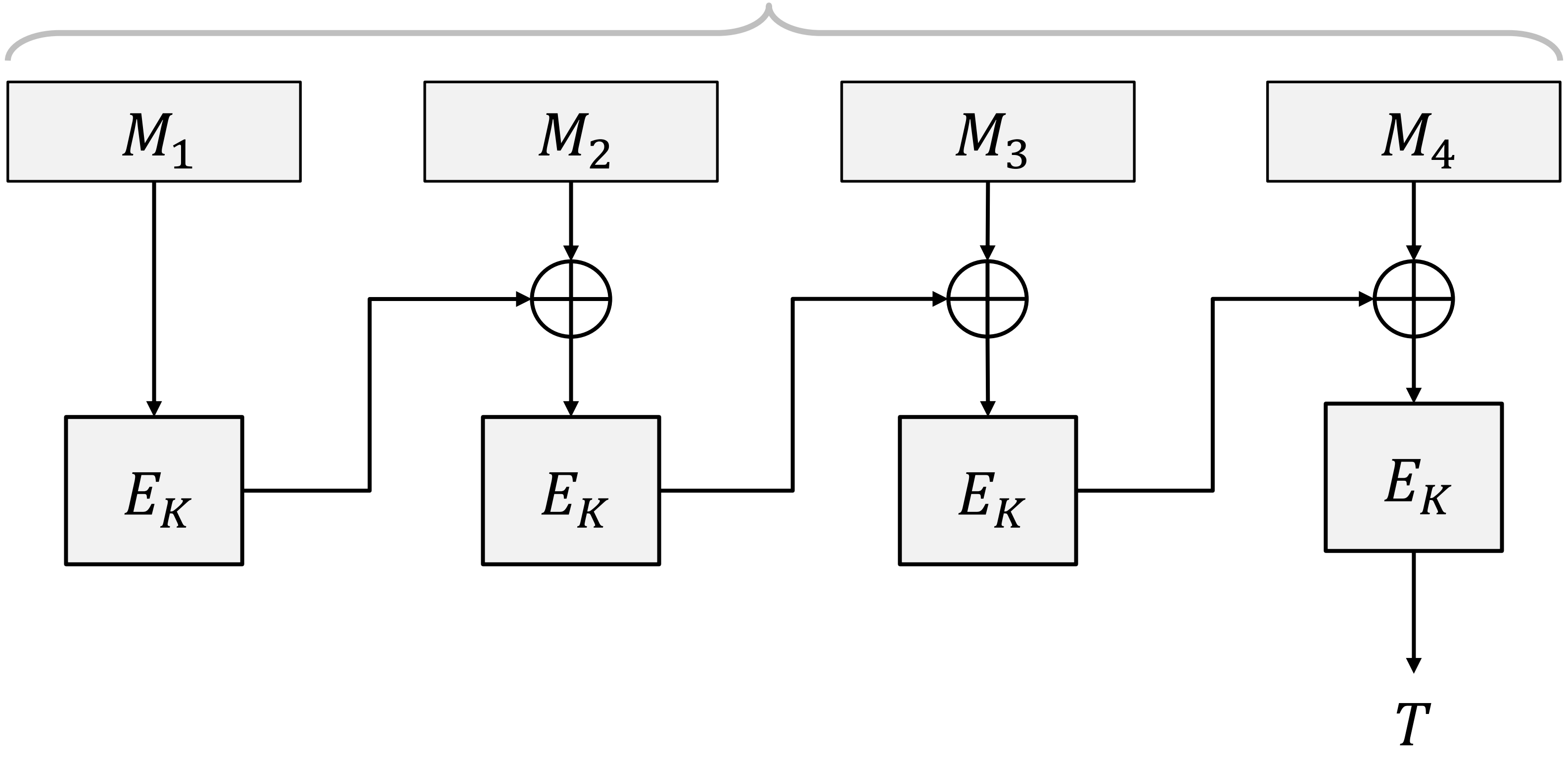
Bob

$T=MAC(k, m)$

$T \stackrel{?}{=} MAC(k, m)$

MAC

- Message Authentication Code (MAC)
  - Used for data **integrity**
  - Integrity **not** the same as confidentiality
- MAC is computed as **CBC residue**
  - That is, compute CBC encryption, saving only final ciphertext block, the MAC
  - The MAC serves as a cryptographic checksum for data


M_1
M_2
M_3
M_4
E_K
E_K
E_K
E_K
T

MAC Computation

- MAC computation (assuming N blocks)

$$C_0 = E(IV \oplus P_0, K),$$
$$C_1 = E(C_0 \oplus P_1, K),$$
$$C_2 = E(C_1 \oplus P_2, K), \ldots$$
$$C_{N-1} = E(C_{N-2} \oplus P_{N-1}, K) = MAC$$

- Send IV, $P_0, P_1, \ldots, P_{N-1}$ and MAC
- Receiver does same computation and verifies that result agrees with MAC
- Both sender and receiver must know K

Does a MAC work?

- Suppose Alice has 4 plaintext blocks
- Alice computes

  $\mathbf{C_0} = E(IV \oplus P_0, K)$, $\mathbf{C_1} = E(\mathbf{C_0} \oplus P_1, K)$,

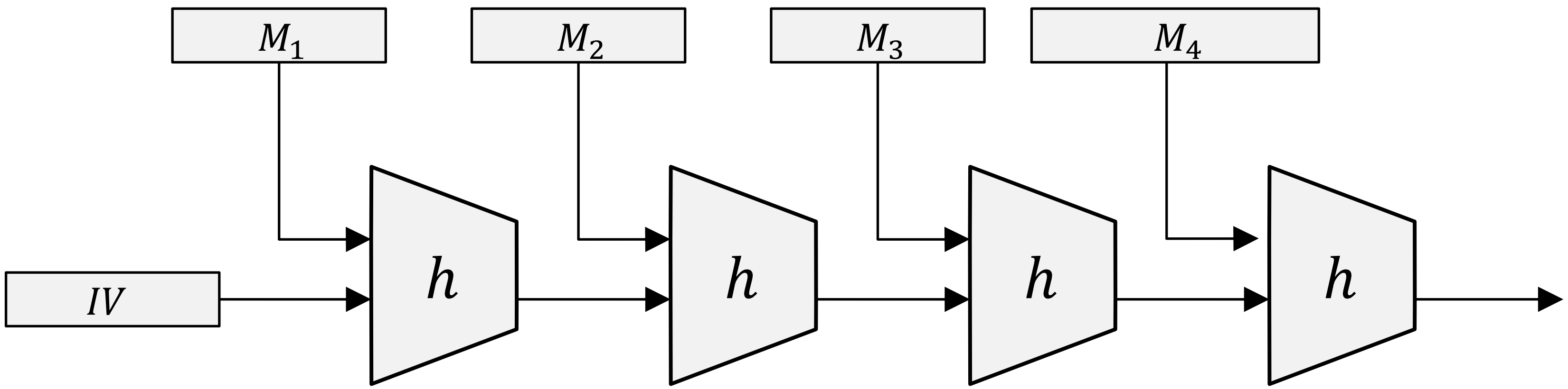
  $\mathbf{C_2} = E(\mathbf{C_1} \oplus P_2, K)$, $\mathbf{C_3} = E(\mathbf{C_2} \oplus P_3, K) = \mathbf{MAC}$
- Alice sends IV, P_0, P_1, P_2, P_3 and **MAC** to Bob
- Suppose Trudy changes P_1 to X
- Bob computes

  $\mathbf{C_0} = E(IV \oplus P_0, K)$, $\boldsymbol{C_1} = E(\mathbf{C_0} \oplus X, K)$,

  $\boldsymbol{C_2} = E(\boldsymbol{C_1} \oplus P_2, K)$, $\boldsymbol{C_3} = E(\boldsymbol{C_2} \oplus P_3, K) = \boldsymbol{MAC} \neq \mathbf{MAC}$
- It works since error propagates into MAC
- Cannot make $\boldsymbol{MAC}$ == **MAC** without key K

HMAC

- Can compute a MAC of the message M with key K using a "hashed MAC" or **HMAC**
- HMAC is a ***keyed*** hash
  - Why would we need a key?
- How to compute HMAC?
- Two obvious choices: h(K,M) and h(M,K)
- Which is better?


M_1
M_2
M_3
M_4
IV
h
h
h
h

HMAC

- Should we compute HMAC as $h(K,M)$?
- Hashes computed in blocks
  - $h(B_1,B_2) = F(F(A,B_1),B_2)$ for some F and constant A
  - Then $h(B_1,B_2) = F(h(B_1),B_2)$
- Let $M' = (M,X)$
  - Then $h(K,M') = F(h(K,M),X)$
  - Attacker can compute HMAC of M' without K
- Is $h(M,K)$ better?
  - Yes, but… if $h(M') = h(M)$ then we might have $h(M,K)=F(h(M),K)=F(h(M'),K)=h(M',K)$

Correct Way to HMAC

- Described in RFC 2104
- Let B be the block length of hash, in bytes
  - B = 64 for MD5 and SHA-1
- ipad = 0x36 repeated B times
- opad = 0x5C repeated B times
- Then

$$HMAC(M,K) = h(K \oplus opad, h(K \oplus ipad, M))$$

Confidentiality and Integrity

- Encrypt with one key, MAC with another key
- Why not use the same key?
  - Send last encrypted block (MAC) twice?
  - This cannot add any security!
- Using different keys to encrypt and compute MAC works, even if keys are related
  - But, twice as much work as encryption alone
  - Can do a little better — about 1.5 "encryptions"
- Confidentiality *and* integrity with same work as one encryption is a research topic

Quantum Computers and Symmetric Ciphers

- Assuming big quantum computers are built, are they a threat to symmetric ciphers?
- Best quantum algorithm for exhaustive search is due to Grover (1996)
- Grover's algorithm is square root faster
- For n bit symmetric key…
  - Conventional computer: Work factor 2^{n-1}
  - Grover's algorithm: Work factor about $2^{n/2}$

Quantum Computers and Symmetric Ciphers

- Assuming big quantum computers are built, are they a threat to symmetric ciphers?
- Not really a serious threat
- If we double the length of the key, work factor for Grover's algorithm is same
- For AES, most common to use 128-bit key
  - But, we can use 256-bit keys
  - Just a little slower with 256 than 128-bit key

Uses for Symmetric Crypto

- Confidentiality
  - Transmitting data over insecure channel
  - Secure storage on insecure media
- Integrity (MAC)
- Authentication protocols (later…)
- Anything you can do with a hash function (upcoming chapter…)