WILFRID LAURIER UNIVERSITY
LAURIER
Inspiring Lives.

CP 460 - Applied Cryptography

Message Authentication Codes (MACs)

Department of Physics and Computer Science
Faculty of Science, Waterloo

Abbas Yazdinejad, Ph.D.

Fall 2024

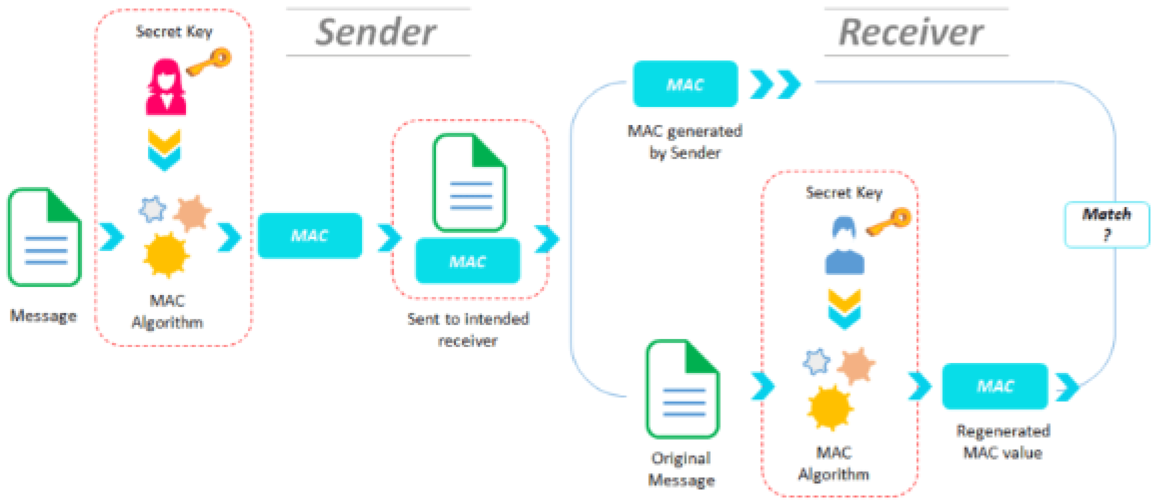
Content of this Chapter

- The principle behind MACs
- The security properties that can be achieved with MACs
- How MACs can be realized with hash functions and with block ciphers

Message Authentication Codes (MACs)

A **MAC** is a cryptographic tool used to ensure the **integrity** and **authenticity** of a message. It allows the recipient of a message to verify two things:

1. Integrity: The message has not been altered or tampered with during transmission.
2. Authenticity: The message was sent by the expected sender (shared a secret key).



Key Features of a MAC:

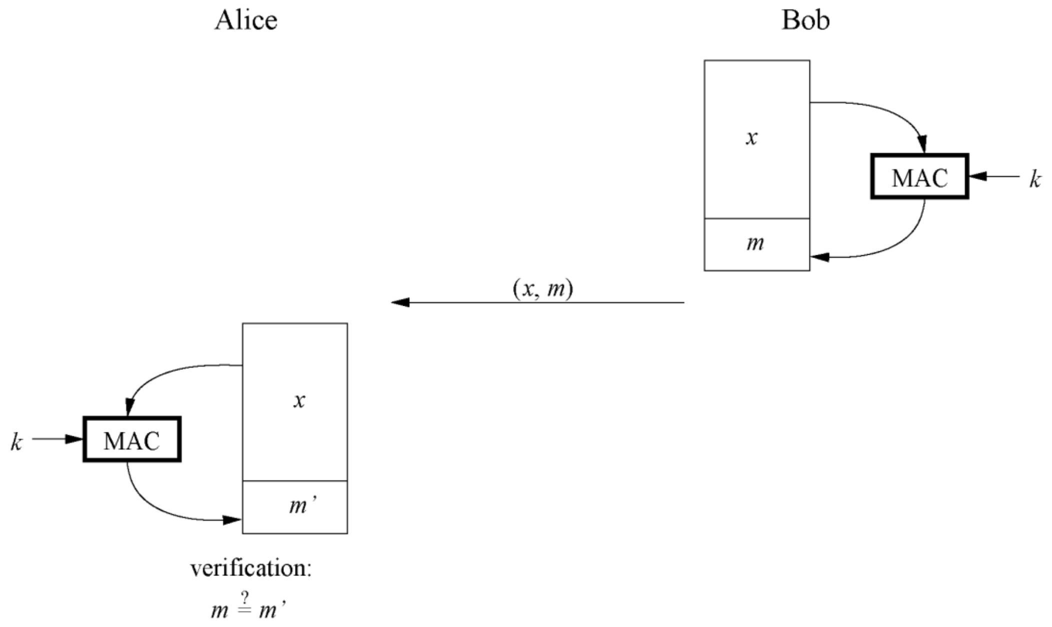
1. Symmetric Key: Both the sender and receiver share a secret key used to generate and verify the MAC.
2. Fixed-Length Output: The MAC produces a fixed-size "tag" regardless of the input message size.
3. Detects Tampering: If the message or the MAC is altered, the verification process will fail.
4. No Nonrepudiation: Unlike digital signatures, MACs do not provide proof of origin because the same key is used for both generation and verification.

Real-World Use Cases:

• **Securing API Communications**: MACs are used to ensure data integrity between systems.
• **Online Payments**: Verifying transaction data to prevent tampering.
• **Secure Protocols**: Used in protocols like TLS to authenticate messages

Principle of Message Authentication Codes

- Similar to digital signatures, MACs append an authentication tag to a message
- MACs use a symmetric key *k* for generation and verification
- Computation of a MAC: $m = \mathrm{MAC}_k(x)$



Bob (the sender) generates a **MAC** using her message x and a shared secret key k. She sends the message x along with the MAC m to **Alice** (the receiver). Bob uses the same key k to recompute the MAC from the received message. If Bob's computed MAC matches Alice's MAC, the message is verified as authentic and untampered. If not, Bob detects tampering or an unauthentic sender.

Properties of Message Authentication Codes

1. **Cryptographic checksum**
   A MAC generates a cryptographically secure authentication tag for a given message.
2. **Symmetric**
   MACs are based on secret symmetric keys. The signing and verifying parties must share a secret key.
3. **Arbitrary message size**
   MACs accept messages of arbitrary length.
4. **Fixed output length**
   MACs generate fixed-size authentication tags.
5. **Message integrity**
   MACs providemessage integrity: Any manipulations of a message during transit will be detected by the receiver.
6. **Message authentication**
   The receiving party is assured of the origin of the message.
7. **No nonrepudiation**
   Since MACs are based on symmetric principles, they do not provide nonrepudiation.

MACs from Hash Functions

Message Authentication Codes (MACs) from Hash Functions involve using cryptographic hash functions, such as SHA-1 or SHA-256, to create a secure authentication tag for messages. These are widely used due to their simplicity, efficiency, and strong cryptographic properties.

- MAC is realized with cryptographic hash functions (e.g., SHA-1)
- HMAC is such a MAC built from hash functions
- Basic idea: Key is hashed together with the message
- Two possible constructions:
  - secret prefix MAC: $m = \mathrm{MAC}_k(x) = h(k||x)$
  - secret suffix MAC: $m = \mathrm{MAC}_k(x) = h(x||k)$
- Attacks:
  - secret prefix MAC: Attack MAC for the message $x = (x_1, x_2, \ldots, x_n, x_{n+1})$*, where* x_{n+1} is an arbitrary additional block, can be constructed from *m without knowing the* secret key
  - secret suffix MAC: find collision x and x_O such that $h(x) = h(x_O)$, then $m = h(x||k) = h(x_O||k)$
- Idea: Combine secret prefix and suffix: HMAC (cf. next slide)

HMAC

A hash-based message authentication code

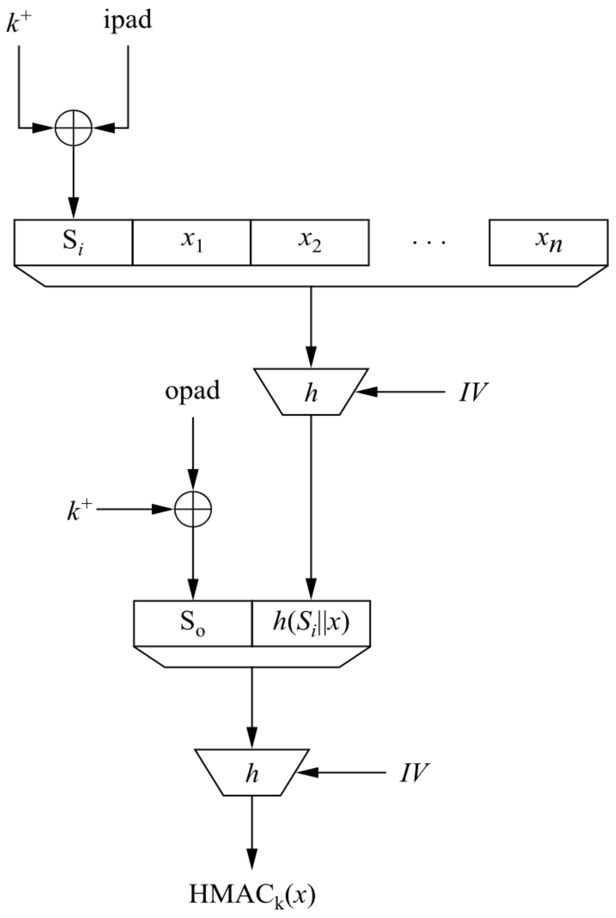
- Proposed by Mihir Bellare, Ran Canetti and Hugo Krawczyk in 1996
- Scheme consists of an inner and outer hash

h: Hash function

k^+: Key padded to the block size of the hash function

opad: Outer padding (fixed value)

ipad: Inner padding (fixed value)



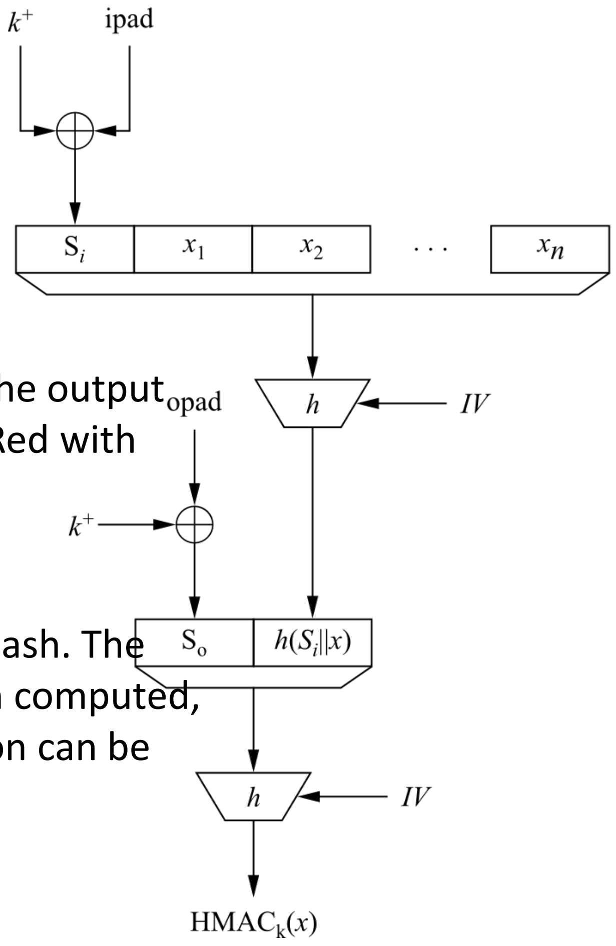
- k^+ is expanded key k
- expanded key k^+ is XORed with the inner pad
- ipad = 00110110,00110110, . . .,00110110
- opad = 01011100,01011100, . . .,01011100
- $\mathrm{HMAC}_k(x) = h[(k^+ \oplus \mathrm{opad}) \| h[(k^+ \oplus \mathrm{ipad}) \| x]]$

- HMAC is provable secure which means (informally speaking): The MAC can only be broken if a collision for the hash function can be found.

HMAC

The MAC computation starts with expanding the symmetric key *k* with zeros on the left such that the result k_+ is *b* bits in length, where *b* is the input block width of the hash function. The expanded key is XORed with the inner pad, which consists of the repetition of the bit pattern:

$$\text{ipad} = 00110110, 00110110, \ldots, 00110110$$



The second, outer hash is computed with the padded key together with the output of the first hash. Here, the key is again expanded with zeros and then XORed with the outer pad:

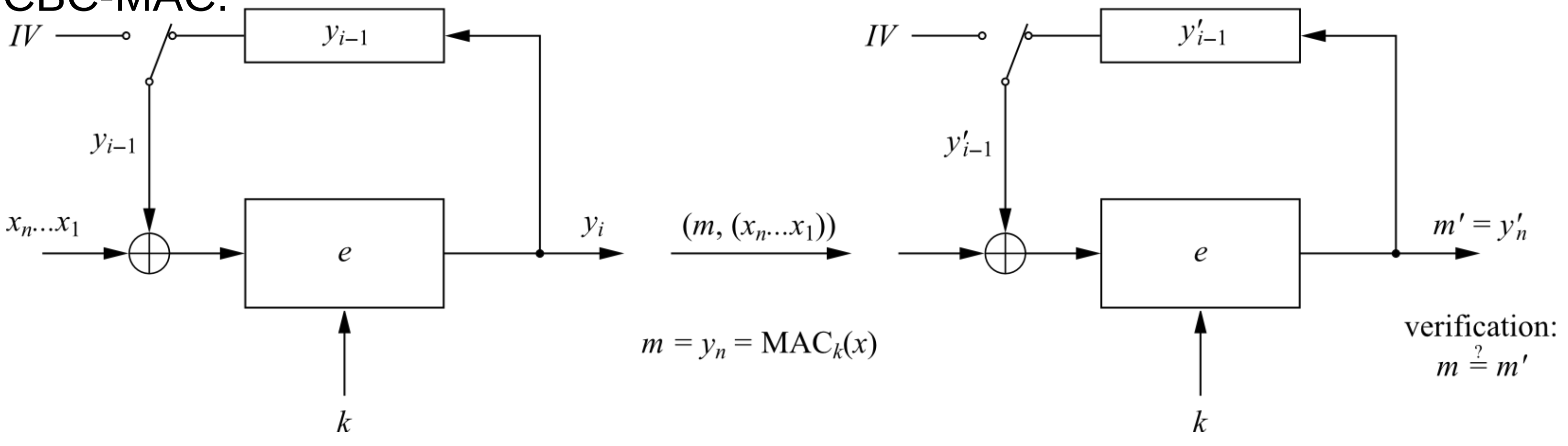
$$\text{opad} = 01011100, 01011100, \ldots, 01011100.$$

The result of the XOR operation forms the first input block for the outer hash. The other input is the output of the inner hash. After the outer hash has been computed, its output is the message authentication code of *x*. The HMAC construction can be expressed as:

$$HMAC_k(x) = h\,(k_+ \oplus \text{opad}) \,||\, h\,(k_+ \oplus \text{ipad}) \,||\, x$$

MACs from Block Ciphers

- An alternative method is to construct MACs from block ciphers
- Approach such as AES in cipher block chaining (CBC)
- MAC constructed from block ciphers (e.g. AES)
- Popular: Use AES in CBC mode
- CBC-MAC:



Block cipher-based MACs rely on symmetric encryption algorithms to process the input message into a fixed-length authentication tag. The most common approach is **CBC-MAC**, which utilizes the **Cipher Block Chaining (CBC)** mode of encryption.

CBC-MAC

- MAC Generation
  - Divide the message *x* into blocks x_i
  - Compute first iteration $y_1 = e_k(x_1 \oplus IV)$
  - Compute $y_i = e_k(x_i \oplus y_{i-1})$ for the next blocks
  - Final block is the MAC value: m =MAC$_k(x) = y_n$

- MAC Verification
  - Repeat MAC computation (*m*‘)
  - Compare results:In case $m' = m$, the message is verified as correct
  - In case $m' \neq m$, the message and/or the MAC value *m* have been altered during transmission

Security Properties of CBC-MAC

• **Integrity**: Any alteration to the message will result in a different MAC, which can be detected.
• **Authentication**: Ensures the message came from the party that holds the shared secret key.

Limitations:

1. **Variable-Length Messages**:
   If the message length varies, CBC-MAC may not be secure because attackers can exploit the predictable structure. A solution is to include the message length as part of the input or use a more advanced variant like CMAC.
2. **Replay Attacks**:
   CBC-MAC alone does not prevent replay attacks. Protocols should include unique identifiers or timestamps for protection.

Applications of Block Cipher-based MACs

1- **TLS and IPsec**: Securing communication channels.
2- **Payment Systems**: Ensuring the integrity and authenticity of financial messages.
3- **IoT Devices**: Lightweight MAC generation using block ciphers like AES.

Key Advantages of Block Cipher-based MACs

1- High performance in hardware.
2- Efficient for systems already using block ciphers (e.g., AES).

1- What are the primary purposes of a Message Authentication Code (MAC)?

2- How does a MAC ensure message integrity and authenticity?

3- Why do MACs rely on symmetric keys?

4- What is the difference between MACs and digital signatures?

The primary purposes of a MAC are:

• **Message Integrity**: To ensure that the message has not been tampered with during transmission.

• **Message Authentication**: To confirm that the message originates from a trusted sender who shares a secret key.

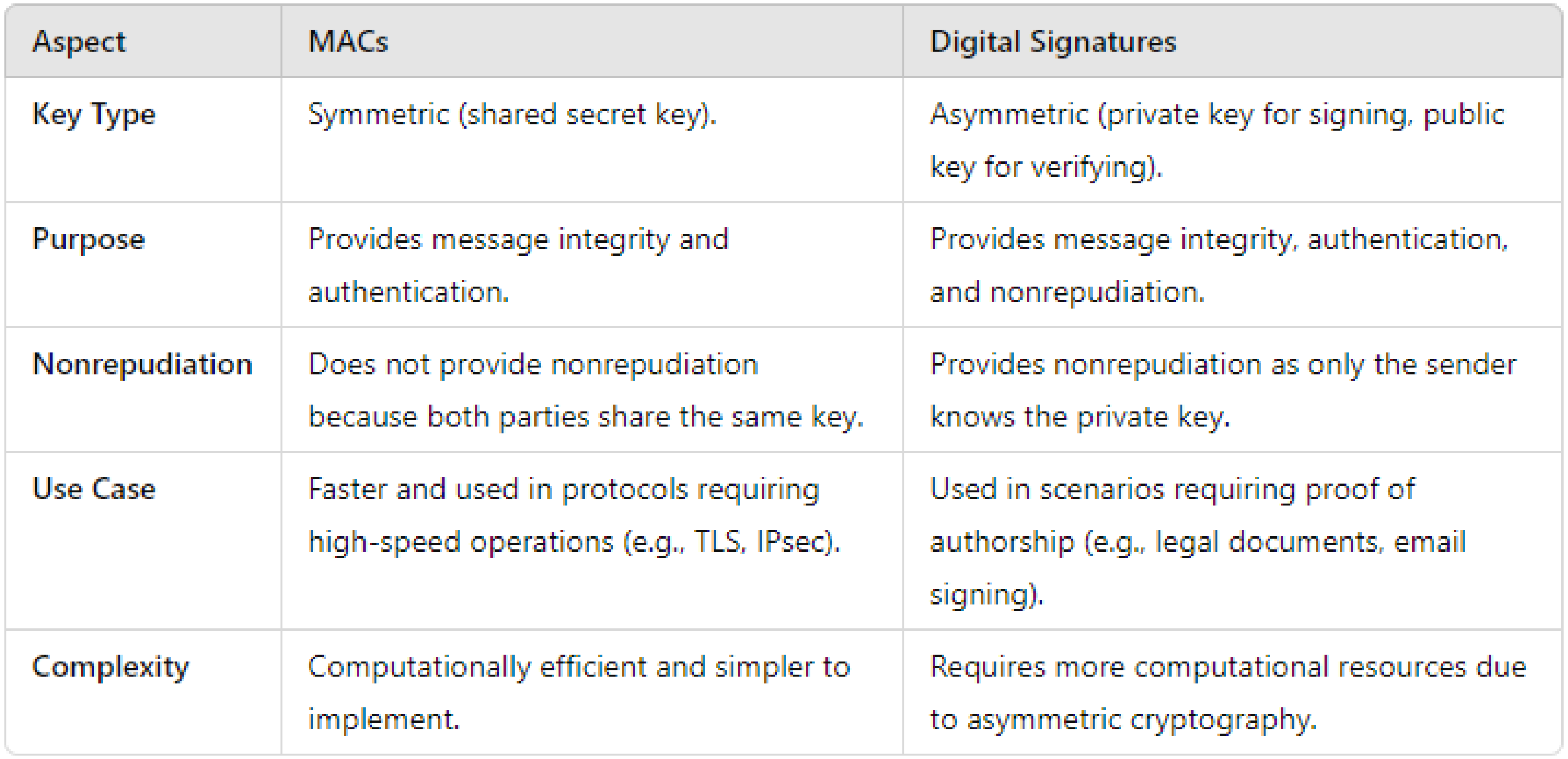
MACs rely on symmetric keys because:

1- The same key is used for both generating and verifying the MAC.

2- This shared secret ensures that only the parties who know the key can compute or verify the MAC, providing confidentiality and trust.

3- Symmetric key operations are computationally faster compared to public key operations, making MACs efficient.

Difference between MACs and digital signatures



Aspect	MACs	Digital Signatures
Key Type	Symmetric (shared secret key).	Asymmetric (private key for signing, public key for verifying).
Purpose	Provides message integrity and authentication.	Provides message integrity, authentication, and nonrepudiation.
Nonrepudiation	Does not provide nonrepudiation because both parties share the same key.	Provides nonrepudiation as only the sender knows the private key.
Use Case	Faster and used in protocols requiring high-speed operations (e.g., TLS, IPsec).	Used in scenarios requiring proof of authorship (e.g., legal documents, email signing).
Complexity	Computationally efficient and simpler to implement.	Requires more computational resources due to asymmetric cryptography.

Lessons Learned

- MACs provide two security services, *message integrity and message authentication,* using symmetric techniques. MACs are widely used in protocols.
- Both of these services also provided by digital signatures, but MACs are much faster as they are based on symmetric algorithms.
- MACs do not provide nonrepudiation.
- In practice, MACs are either based on block ciphers or on hash functions.
- HMAC is a popular and very secure MAC, used in many practical protocols such as TLS.

Thank
You