Introduction to Cryptography
ECE 597XX/697XX
Part 12
Message Authentication Codes (MACs)

Israel Koren

Content of this part

- 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
**Principle of MACs**

- 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:
  \[ m = MAC_k(x) \]

**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 provide message 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

- **MAC** is realized with cryptographic hash functions (e.g., SHA-1)
- **HMAC** is such a MAC built from a hash function
- Basic idea: Key is hashed together with the message
- Two possible constructions:
  - secret prefix MAC: \( m = MAC_k(x) = h(k||x) = h(k||x_1, x_2, \ldots, x_n) \)
  - secret suffix MAC: \( m = MAC_k(x) = h(x||k) = h(x_1, x_2, \ldots, x_n||k) \)
- 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
  - Oscar intercepts \( x = (x_1, x_2, \ldots, x_n) \) and \( m \)
  - Adds \( x_{n+1} \) and calculates \( m^O = h(m||x_{n+1}) \)
  - Sends \( (x_1, x_2, \ldots, x_n, x_{n+1}) \) and \( m^O \)

Secret suffix MAC

- \( m = MAC_k(x) = h(x||k) = h(x_1, x_2, \ldots, x_n||k) \)
- Attack:
  - find collision \( x \) and \( x^O \) such that \( h(x) = h(x^O) \), then \( m = h(x||k) = h(x^O||k) \)
  - can replace \( x \) by \( x^O \)
  - for a 160-bit about \( 2^{80} \) attempts are needed
HMAC

- Proposed by Bellare, Canetti and Krawczyk in 1996
- Avoids the above security weaknesses
- Scheme consists of an inner and outer hash

- $k'$ is expanded key $k$ with 0's on the left to match the size of a hash block
- expanded key $k'$ is XORed with inner pad
- $\text{ipad} = 00110110, 00110110, \ldots, 00110110$
- $\text{opad} = 01011100, 01011100, \ldots, 01011100$
- $\text{HMAC}_k(x) = h((k' \oplus \text{opad}) || h((k' \oplus \text{ipad}) || x))$

MACs from Block Ciphers

- MAC constructed from block ciphers (e.g., AES)
- Popular: Use AES in CBC (cipher block chaining) mode

- CBC-MAC:
**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: If $m' = m$, the message is verified as correct
  - If $m' \neq m$, the message and/or the MAC value $m$ have been altered during transmission

---

**Lessons Learned**

- **MACs** provide two security services, *message integrity* and *message authentication*, using symmetric ciphers. MACs are widely used in protocols.
- Both of these services are also provided by digital signatures, but MACs are much faster.
- MACs do not provide nonrepudiation.
- In practice, MACs are either based on block ciphers or on hash functions.
- HMAC is a popular MAC used in many practical protocols such as Transport Layer Security (TLS) – indicated by a small lock in the browser.