CS458: Introduction to Information Security

Notes 8: Message Authentication Codes (MACs)

Yousef M. Elmehdwi

Department of Computer Science

Illinois Institute of Technology

yelmehdwi@iit.edu

October 18, 2018

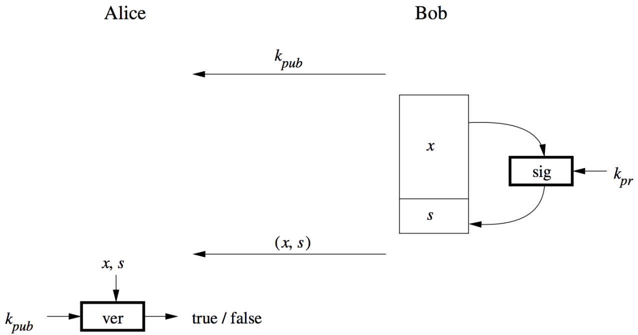
Slides: Modified from Christof Paar and Jan Pelzl

Outline

- 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

Introduction to Message Authentication Codes (MACs)

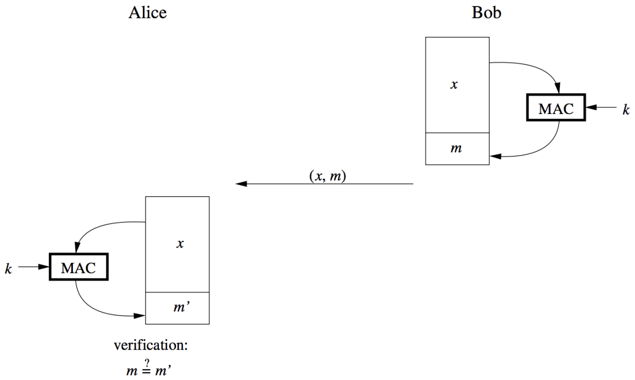
- MAC is a digital signature associated with a symmetric (one-key) signature scheme
- Terminology MACs are also called "cryptographic checksums"
- Recall motivation for digital signatures
  - "message authentication"
  - How to do that?



  - Let's try the same with symmetric algorithm

Principle of Message Authentication Codes

- MACs use a symmetric key k for generation and verification
- Computation of a MAC:
  - A MAC is generated by a function $m = MAC_k(x)$ that can be computed by anyone knowing the secret key k
- Bob computes $m = MAC_k(x)$ and sends (x, m) to Alice.
- Alice receives (x, m') and verifies that $m' = m$.



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

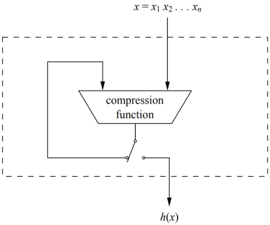
MACs from Hash Functions

- MAC is realized with cryptographic hash functions (e.g., SHA-1)
- HMAC is such a MAC built from hash functions
- **Basic idea**: Key is hashed together with the message

  $m = MAC_k(x) = h(k, x)$
- Two possible constructions:
  - secret prefix MAC: $m = MAC_k(x) = h(k || x)$
  - secret suffix MAC: $m = MAC_k(x) = h(x || h)$
- **Better solutions?**: Combine secret prefix and suffix: HMAC

Attack Against Secret Prefix MACs

- Assume $x = (x_1, x_2, \dots, x_n)$



- $m = MAC_k(x) = h(k||x_1||x_2||\dots||x_n)$ is computed using Merkle-Damgård hash function construction.
  - This iterated approach is used in the majority of today's hash functions
- Attack MAC for the message $x = (x_1, x_2, \dots, x_n, x_{n+1})$, where x_{n+1} is an arbitrary additional block, can be constructed from m without knowing the secret key.

Attack Against Secret Prefix MACs

Alice		**Oscar**		**Bob**
				$x = (x_1, \ldots, x_n)$ $m = h(k \| \| x_1, \ldots, x_n)$
		↯ intercept	$\xleftarrow{(x,m)}$	
		$x_O = (x_1, \ldots, x_n, x_{n+1})$ $m_O = h(m \| \| x_{n+1})$		
	$\xleftarrow{(x_O, m_O)}$			
$m' = h(k \| \| x_1, \ldots, x_n, x_{n+1})$ since $m' = m_O$ $\Rightarrow$ valid signature!				

- 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 $x = (x_1, x_2, \ldots, x_n, x_{n+1})$ and m_o
- Note: Attack does not work if padding with length information is being used.

Attack Against Secret Suffix MACs

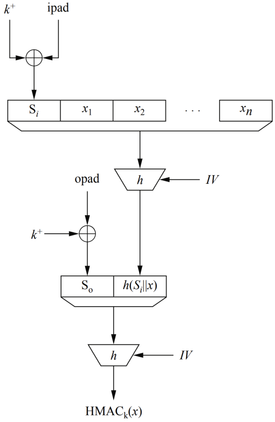
- Assume $x = (x_1, x_2, \ldots, x_n)$
- $m = MAC_k(x) = h(x||h) = h(x_1||x_2||\ldots||x_n||k)$
- **Attack**:
  - Assume Oscar can find collisions, x and x_o such that $h(x) = h(x_o)$, then $m = h(x||k) = h(x_o||k)$
  - Can replace x with x_o
- Q: Is this a problem, i.e., does Oscar gain anything?

  $\Rightarrow$ Compare brute-force effort with collision-finding effort:
  - Example: $h() \rightarrow$ SHA-1 ($160\ bit$ output)
    $|K| = 128\text{-}bit \rightarrow$ we expect attacker complexity of 2^{128}
  - but collision search takes $\approx \sqrt{2^{160}} = 2^{80}$ steps (birthday paradox)
  - $\Rightarrow$ cryptographically, make MAC attackable by birthday attack.

HMAC Construction

- Proposed by Bellare, Canetti and Krawczyk in 1996
- Avoids the above security weaknesses
- Widely used in practice, e.g., SSL/TLS
- **idea**:
  - Use two nested secret prefix MACs
  - Roughly, $h(k \,||\, h(k \,||\, x))$
  - i.e., Scheme consists of an inner and outer hash
- In reality: $HMAC_k(x) = h[(k^{+} \oplus opad) \,||\, h((k^{+} \oplus ipad) \,||\, x)]$
  - k^{+} is expanded key k
    - zero extended on the left to match hash block size ($k^{+} = 00...00 \,||\, k$)
  - Expanded key k^{+} is XORed with inner and outer pads
  - Padding:
    - Let B be the block length of hash in bytes.
    - $0x36$ repeated B times for $ipad$
    - $0x5c$ repeated B times for $opad$
      e.g., $B = 64$ for MD5 and SHA-1

HMAC Construction



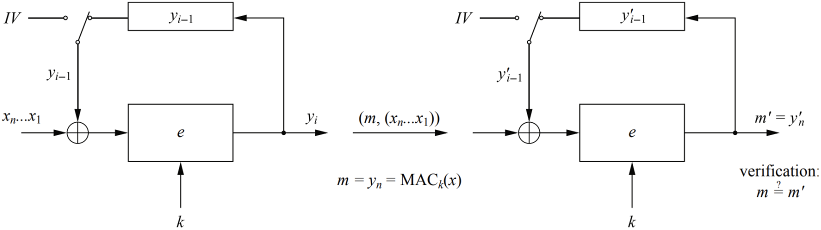
- Note: Message is only processed in inner hash!.
- HMAC is provable secure which means (informally speaking): The MAC can only be broken if a collision for the hash function can be found.

HMAC and Strong Collisions

- Birthday attacks don't make sense in HMAC scenario
  - Attacker would need to know k to generate candidate message/hash pairs

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

Summary: Protecting messages

- Encryption protects message confidentiality.
  - prevents unauthorized disclosure
- We also wish to protect message integrity and authenticity.
  - Integrity means that the message has not been altered.
    - detect unauthorized writing (i.e., modification of data)
  - Authenticity (Source Authentication) means that the message is genuine.
- Encryption alone does **not** provide integrity.
  - One-time pad, ECB cut-and-paste, etc.
- Q: When data integrity is more important than confidentiality?
  - Example: Inter-bank fund transfers
    Confidentiality may be nice, integrity is **critical**

Summary: Protecting integrity and authenticity

- Integrity and authenticity are protected using symmetric or asymmetric methods.
- A digital signature or a message authentication code (MAC) is a string s that binds an individual or other entity $\mathtt{A}$ with a message x.
  - The recipient of the message verifies that s is a valid signature of $\mathtt{A}$ for message x.
  - It should be hard for Eve to create a valid signature s' for a message x' without knowledge of $\mathtt{A}$'s secret information.

Summary: Non-non-repudiation

- Alice orders *100* shares of stock from Bob.
- Alice computes MAC using symmetric key.
- Stock drops, Alice claims she did not order.
- Q: Can Bob prove that Alice placed the order?
  - **No!** Bob also knows the symmetric key, so he could have forged the MAC.
- Problem: Bob knows Alice placed the order, but he can't prove it.

Summary: Non-repudiation

- **Can an asymmetric scheme help?**
- Alice orders *100* shares of stock from Bob.
- Alice signs order with her private key.
- Stock drops, Alice claims she did not order.
- Q: Can Bob prove that Alice placed the order?
  - **Yes!** Alice's private key used to sign the order - only Alice knows her private key.
  - Of course, this assumes Alice's private key has not been lost/stolen.

Summary: Non-repudiation

- Non-repudiation refers to a state where the author of a statement will not be able to successfully challenge the authorship of the statement.
- What is the relationship between authenticity and non-repudiation?
  - Authenticity: Alice interacts with Bob and convinces him that a message x truly came from her.
  - Non-repudiation: Same as above but now Bob can convince Charlie too.

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 non-repudiation.
- 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.