Message Authentication Codes

Ramki Thurimella
MAC

- Also called keyed hash function
- Used for Authentication, i.e. stopping Eve from tampering the message in transit
- MAC(K, m) is a fixed size value, referred to as the tag for m
MAC Uses

- Not suitable for digital signatures
- Because, in addition to Alice, Bob knows the key K
- For digital signatures to have legal value the property of *non-repudiation* is needed
  - *Repudiation* (legal) of a contract means a refusal to perform the duty or obligation owed to the other party
  - *Non-repudiation* (digital context) the purported maker of a statement will NOT be able to successfully challenge the validity of the statement or contract (i.e. in simple terms, if a person signs a contract, that person will NOT be able to be able to say in the future “I never signed that contract”)


CBC-MAC

- For a message $P_1, P_2, ..., P_k$
  
  $H_0 \leftarrow IV$
  
  $H_i \leftarrow E_K(P_i \oplus H_{i-1})$
  
  $MAC \leftarrow H_k$

- Sometimes only a PART of the final block is used as the MAC value

- IV is commonly fixed at 0

- If the same key $K$ is used for encryption for authentication, then the ciphertext for the last block and the MAC value would be the same
An Example Collision Attack

- Security is limited to half the block length
- Claim: if MAC(a) = MAC(b), then MAC(a || c) = MAC(b || c) for any single block c
  
  \textbf{Proof:}
  
  \[ \text{MAC}(a || c) = E_K(c \oplus \text{MAC}(a)) \] – by the definition of MAC
  
  = \[ E_K(c \oplus \text{MAC}(b)) \] – by our assumption
  
  = \[ \text{MAC}(b || c) \] – by the definition of MAC

- Collect a large number of messages until a collision, i.e. for some a, b, MAC(a) = MAC(b). (By Birthday Paradox, for a 128-bit block, this is expected to happen after \(2^{64}\) messages.)

- Get the attacker to authenticate MAC(a || c)
- Replace a || c with b|| c, and send the same MAC
- Solve Exercise 6.2
Collision Attack (cont.)

- This is NOT a generic attack, since the claim proved in the previous slide does hold for a random mapping.

- Another attack:
  - Assume $c$ is one block long and $\text{MAC}(a \ || \ c) = \text{MAC}(b \ || \ c)$
  - Claim: Under these assumptions, $\text{MAC}(a \ || \ d) = \text{MAC}(b \ || \ d)$ for any block $d$.

- Attack similar to the previously described one:
  - Collect enough message until a collision occurs for some $m_i$ and $m_j$ such both end in $c$.
  - Get the sender to authenticate $a \ || \ c$ and replace it by $a \ || \ d$. 
CBC-MAC

- Notwithstanding the attacks given before, CBC-MAC is acceptable if the block size is 256.
- CBC-MAC should be used as shown below (Otherwise, differing message lengths lead to weaknesses—Solve Exercise 6.3)
  - Construct $s = l || m$ where $l$ is the length of $m$ in fixed length format
  - Pad $s$ if necessary and apply CBC-MAC to $s$
  - Output the last ciphertext block; do NOT output any intermediate values
Cipher-Based MAC (CMAC)

- Since CBC-MAC and encryption share many computations, efficient implementations are possible
- **Recommended alternative: CMAC**
  - Standardized by NIST
  - Works exactly like CBC-MAC except treats last block differently
  - CMAC XORs one of two special values prior to encrypting the last block
  - Special values are derived from the key K
  - One of the two special values is chosen based on whether padding is required or not
HMAC

- Simple definitions like $h(K \ || \ m)$ or $h(K \ || \ m \ || \ K)$ leads to problems when iterative hash functions are used.
- For two hex constants $a$ and $b$, HMAC computes
  $$h(K \ a \ || \ h(K \ b \ || \ m))$$
- HMAC works with any iterative hash function.
- In fact, HMAC with SHA-1 is not as bad as SHA-1 alone.
- HMAC is also limited to $n/2$ bits of security.
- To achieve 128-bit security, use SHA-256.
GMAC

- Has been standardized by NIST recently
- Fundamentally different from
  - CBC-MAC
  - CMAC
  - HMAC
- Uses 3 parameters
  - Key
  - Nonce
  - Message
- Nonce could be implicit, e.g. message number, or sent explicitly along with the message
- GMAC doesn’t meet the definition of indistinguishability from the ideal MAC function as GMAC takes 3 inputs whereas the ideal MAC takes 2 inputs: m and k
- GMAC uses universal has functions under the hood, along with a block cipher in CTR mode
- For IV, GMAC uses a function of its nonce.
Like other MACs, GMAC provides only $n/2$ bits of security.

If 32-bits are desired, truncating the result to 32 bits might yield only 16-bit security—don’t use GMAC when short MACs are desired.

As mentioned earlier, leaving nonce source to the developer is ALWAYS risky.
Which MAC to Choose?

- Recommendation—HMAC-SHA-256
- Though GMAC is fast, it provides only 64 bits of security
  - Furthermore, GMAC depends on nonces
Using MAC Properly

- Eve could mount a replay attack
- Using the same key $K$ in both directions, similar problems arise
- Therefore, some *contextual* data must accompany MAC tags
- *The Horton Principle*
  
  *Authenticate what is meant, not what is said*

- MAC only authenticates bytes
- Include contextual data $d$ that includes:
  - Protocol identifier
  - Version numbers
  - Protocol message identifier
  - Sizes of various fields
  - Use XML to encode the above data