Designing Hash functions
and message authentication codes

Reviewing...
• We have seen how to authenticate messages:
  – Using symmetric encryption, in an heuristic fashion
  – Using public-key encryption in interactive protocols
  – Using digital signatures to authenticated short tags (secure hash values) of messages

Message Authentication Codes
• Message authentication codes are a very important authentication tool:
  – MACs are based on block ciphers or hash functions, and are very efficient to compute.
  – MACs provide authentication guarantees, unlike symmetric key encryption, which is heuristic.
• On the downside, MACs do not provide for non-repudiation, as they are symmetric-key primitives.
Block-cipher-based MAC

- Consider the CBC mode or operation:

\[ C_n = \text{Cipher}(M_n \oplus \text{opad}) \]

\[ C_{n+1} = \text{Cipher}(C_n \oplus \text{ipad}) \]

\[ \text{MAC}(K, M) = C_n \]

Hash-based MAC (HMAC)

- MACs can be built from hash functions, as well as block ciphers.
- The HMAC construction has been standardized by NIST.
- \( H : \{0, 1\}^* \rightarrow \{0, 1\}^8 \) is a hash function. (\( L \) is hash output size in bytes.)
- The standard defines an integer \( B \)— most inputs to the hash function in the HMAC construction have size \( B \) (in bytes).

HMAC Constants

- Outer pad \( \text{opad} \): \( B \) bytes long
  - NIST defines \( \text{opad} = 5C||5C||...||5C \)
    (8 copies of the byte w/ hex. repres. is 5C)
- Inner pad \( \text{ipad} \): Also \( B \) bytes long
  - NIST defines \( \text{ipad} = 36||36||...||36 \)
    (8 copies of the byte w/ hex. repres. is 36)
- \( K = \) secret key
- \( K_0 = 8\)-byte long key derived from \( K \)
- \( t = \) length of HMAC output in bytes
  - (application-specific, could be smaller than \( L \).)
HMAC Computation

- If $K$ is already $B$-bytes long, then
  \[ K_0 := K \]
- Else, if $K$ is less than $B$-bytes long:
  \[ K_0 := K || 00...00 \] (append zero bytes)
- While, if $K$ is longer than $B$-bytes long:
  \[ K_0 := H(K) || 00...00 \] Hash first, then append zeros.
  \( B \) is always larger than \( L \).
- \( \text{HMAC}(K, M) := \) t-lefmost bytes of
  \[ H(K_0 \oplus \text{opad}) || H(K_0 \oplus \text{ipad}) || \text{text} \]
What is known

- If CBC is the encryption mode:
  - CBC-MAC should **not** be combined with CBC encryption with the same key.

\[
\begin{align*}
M_{n-1} & \oplus \cdots \oplus M_2 & \oplus M_1 & \oplus \text{MAC}(K, M) \\
C_{n-2} & \oplus C_{n-1} & \oplus \text{MAC}(K, M) & \oplus \text{Enc}(K, 0)
\end{align*}
\]

CBC + Authentication

- Provided that the authentication and encryption uses different keys, it is safe to either:
  - Authenticate the plaintext using any MAC (including CBC-MAC with \(K_{\text{auth}}\)) and encrypt plaintext + MAC using CBC with \(K_{\text{enc}} \neq K_{\text{auth}}\).
    - Called Authenticate-then-Encrypt (AtE).
  - Encrypt the message, then compute a MAC on the ciphertext, sending ciphertext + MAC.
    - Called Encrypt-then-Authenticate (EtA).

The order of encryption and authentication

- As we have seen, both EtA and AtE are secure if
  - CBC is the encryption mode, and two keys are used.
    - If CBC is the mode for encryption (common case), then AtE is often preferred, because it allows saving the plaintext + authentication tag and discarding the encryption.
  - In other modes of operation (particularly OFB), AtE is not recommended. In these cases, the authentication tag should be computed on the encrypted message. Summarizing:
    - AtE preferred in CBC mode (and CCM).
    - EtA preferred in other modes.
Using the same key for encryption and authentication

• It is possible to use the same key for both encryption and authentication, if the MAC function is CBC-MAC and the encryption is in Counter mode and the MAC is CBC-MAC. In this case, the CBC-MAC should be computed before encryption.
• This mode is called CCM by NIST (pending approval).

Combining encryption and authentication

• The use of authenticated encryption requires two passes over the data, one computing encryption, the other authentication.
• Some proposed new modes of encryption are capable of combining encryption and authentication. They may eventually be adopted by standard organizations.

Summarization: Hash function uses

• To create a short tag which can be used to authenticate long messages with a single application of the signing algorithm, while providing non-malleability.
• To create a message authentication code (HMAC), enabling mutual authentication of messages with a symmetric key.
• To implement a key-generation function, permitting common secrets to be expanded into several keys.
Building hash functions

- Hash functions are often constructed using compression functions.
- A compression function is like a hash function that takes a slightly larger input than its output. For instance:
  - One-bit compression: \( H: \{0, 1\}^{t+1} \rightarrow \{0, 1\}^t \).
  - Half-length compression: \( H: \{0, 1\}^{2t} \rightarrow \{0, 1\}^t \).
- Given a secure compression function, one may use the Merkle-Damgård construction to get a secure hash function.

Generating multiple keys from shared secrets

- In what we have seen, it is often the case that a single shared secret (say, the result of a key agreement protocol) must be used to derive several values, for instance an encryption key, an authentication key and often also an IV.
- In order to expand a common value into a string or arbitrary size, standards define key generation functions (special purpose pseudo-random functions).
- These are often implemented by simple variations on hash functions.

Secure hash

- In this example, the compression function takes \( n + b \) bits into \( n \) bits.
- Each application of the compression function allows to process \( b \) bits of the message.