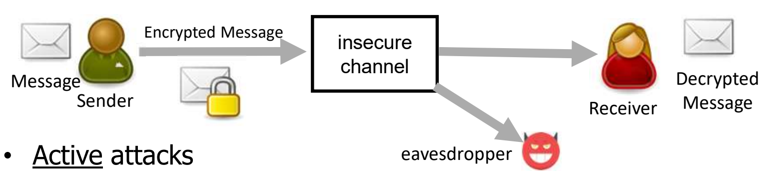
Lecture 8: Integrity

Stephen Huang

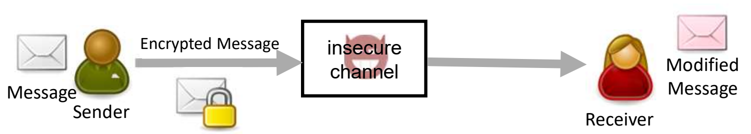
Content

1. Attacks Against Communication

- Protecting confidentiality against passive attacks.

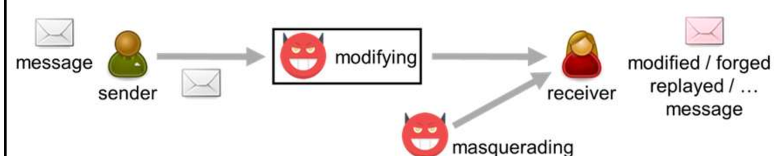


- Active attacks



- Data integrity: information cannot be modified in an unauthorized and undetected manner

Active Attacks in Communication Channel



- Content modification: changing the contents of a message
- Sequence modification: changing the sequence of messages, including deleting some of them
- Timing modification: delay or replay messages
- Masquerade (i.e., forgery): inserting messages of fraudulent source

Message Authentication

- An authenticator produces a value to be used to authenticate a message.
- The authenticator is used in a protocol that verifies the authenticity of a message (for not being altered).
- There are three classes of authenticators.
  1. **Message Encryption**: The ciphertext of the entire message serves as its authenticator.
  2. **Hash Function**: A function that maps a message of any length into a fixed-length hash value.
  3. **Message Authentication Code (MAC)**: A function of the message and a secret key that produces a fixed-length value.
- MAC is also known as cryptographic checksum.
- We will concentrate on the MAC in this lecture.

2. Message Authentication Code

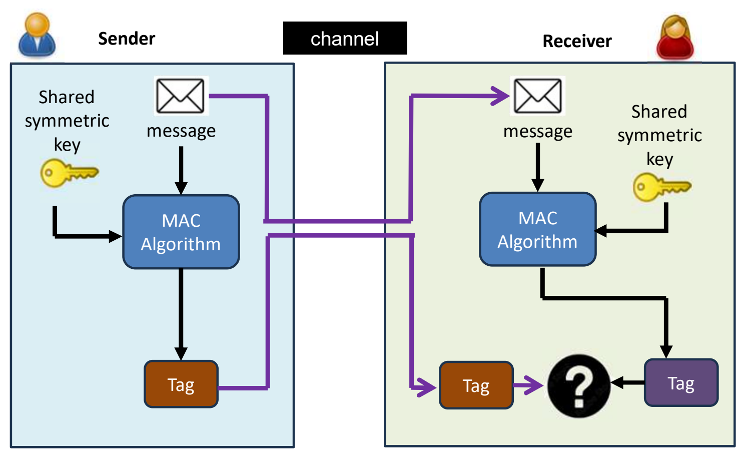
- Message authentication verifies that received messages come from the alleged source and have not been altered.
- Message authentication code MAC(K, M): takes a secret key K and an arbitrary-length input M and produces a tag T.
  - can be efficiently and deterministically computed given key K and message M
  - cannot be computed efficiently given only a message M without key K

MAC vs. Hashing

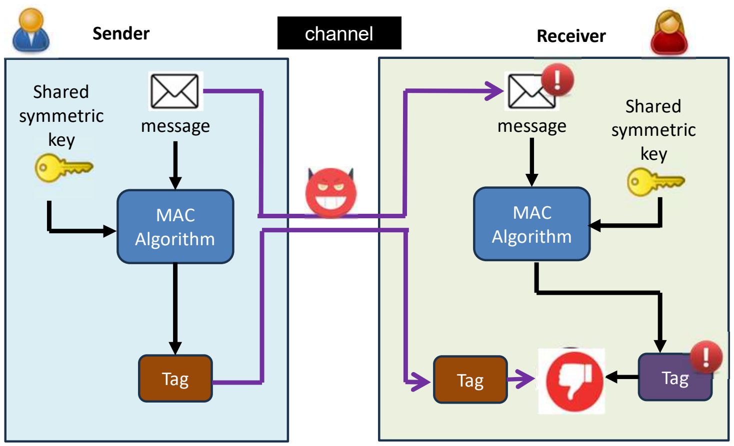
- Hashing is a one-way encryption process applied to the original plaintext to generate a fixed-size ciphertext, called a message digest or a hash.
  - Hash functions are used to ensure data integrity (has not been changed) only.
- MAC is an algorithm that takes a message M combined with a shared secret key K.
  - Also called a keyed hash function,
  - MACs are employed for data integrity (has not been changed) and authentication (came from the stated sender).

Some textbooks define authentication differently.

MAC



MAC Compromised



Message Authentication Code Properties

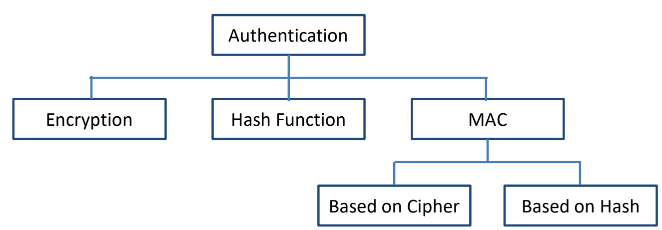
- MAC(K, M)
  - Looks like a pseudorandom function to the attacker.
  - It does not need to be invertible; the sender and receiver use it the same way.
- Correct tag proves
  - **Authenticity**: the message is from an entity that knows the secret key.
  - **Integrity**: the message has not been altered by an entity that does not know the key.
- Detecting timing or sequence attacks: include a timestamp or sequence number in the message.
- MAC can be independent of encryption: we can provide only integrity, confidentiality, or both.

Brute-Force Attacks

- Tag forging: What is the probability that a random tag matches a message? Success attack depends on the length of the tag (independent of the message length).
  - too short: high probability of an arbitrary tag matching a modified message (with an n-bit tag, the probability is 2^{-n})
  - too long: consumes bandwidth
- Key search
  - suppose that the key length is k bits,
  - finding the right key takes, on average, 2^{k-1} steps.

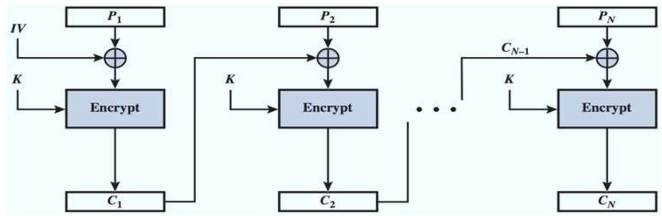
3. Message Authentication

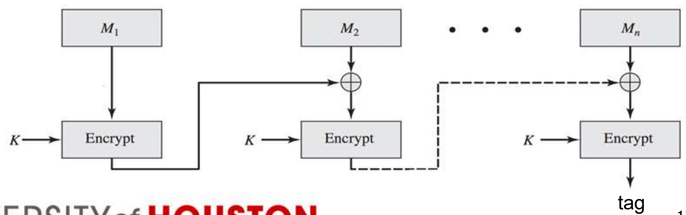
- Based on Block Ciphers
- Based on Hash Functions



MAC: Based on Block Ciphers

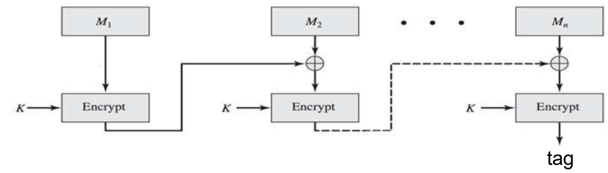
- Based on the CBC mode of operation





CBC-MAC

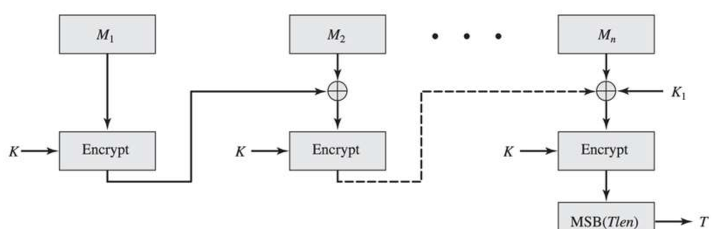
- Based on the Cipher Block Chaining (CBC) mode of operation
- Must use different keys for CBC encryption and CBC-MAC authentication.



- Not secure for variable-length messages
  - given a one-block message X and its tag T = MAC(K, X), attacker can create message X | (X ⊕ T), whose tag is also T.

Cipher-based MAC (CMAC)

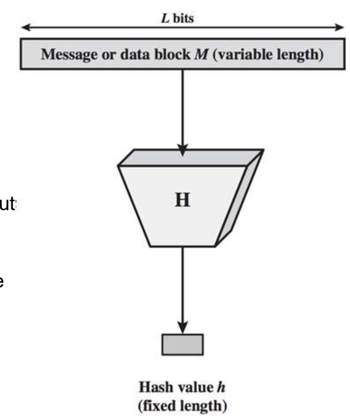
- Standardized in 2005 by NIST
- Thwarts forgery for variable-length messages
- Second key K_1 is derived from E(K, 0)



use as many bits Tlen for the tag as you want

Reminder: Cryptographic Hash Functions

- Hash function H: deterministically maps an input M to a fixed-length hash value H(M)
- Requirements
  - **one-way**: finding an input for which the output is a given hash value is hard (i.e., function is hard to invert)
  - **collision-resistant**: finding two input for which the hash values are the same is hard
  - **efficient**: computing the hash value of a given input is easy
  - **pseudorandom**: output meets standard tests for pseudo-randomness



HMAC

- **HMAC**, Hash-based Message Authentication Code, is a MAC using a cryptographic hash function and a secret key.
- Published in 1996 by Mihir Bellare, Ran Canetti, and Hugo Krawczyk.
- Provably secure if the hash function is pseudorandom (collision-resistance is not necessary).
- Works with any hash function
  - but it is more efficient with iterative hash functions.
- Widely used
  - part of the IPSec and SSL/TLS protocols.
  - standardized in NIST FIPS PUB 198 and RFC 2104.
- Formula:

$$\text{HMAC}(K, M) = H(K_{outer} \mid H(K_{inner} \mid M))$$

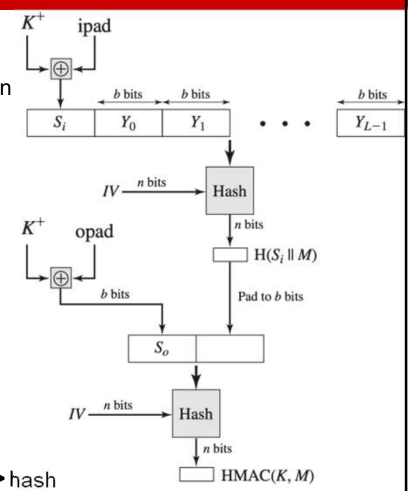
HMAC Design Objectives

- To use available hash functions without modifications ("black box").
- To allow easy replacement in anticipation of faster, more secure hash function ("Plug-and-Play").
- To preserve the performance of the hash function with no significant overhead.
- To use and handle keys in a simple way.
- The strength of the authentication mechanism is based on reasonable assumptions about the embedded hash function.

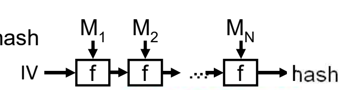
- H = embedded hash function (e.g., MD5, SHA-1, etc.)
- IV = initial value input to hash function
- M = message input to HMAC (including the padding specified in the embedded hash function)
- Y_i = i th block of M, $0 \le i \le (L - 1)$.
- L = number of blocks in M.
- b = number of bits in a block.
- n = length of hash code produced by embedded hash function.
- K = secret key; recommended length is $\ge n$; if key length is greater than b, the key is input to the hash function to produce an n-bit key.
- K^+ = K padded with zeros on the left so that the result is b bits in length.
- Ipad = `00110110`, Opad = `01011100`, repeated b/8 times.

HMAC Structure

- b: block size of the hash function
- IV: initial value of the hash function
- Inputs
  - $Y_0, \ldots, Y_{L-1}$: message blocks
  - K^+: key (padded with zeros)
  - ipad = `00110110` repeated
  - opad = `01011100` repeated
- Output HMAC(K, M) = $H(K^+ \oplus \text{opad} \mid H(K^+ \oplus \text{ipad} \mid M))$



Iterative hash function:



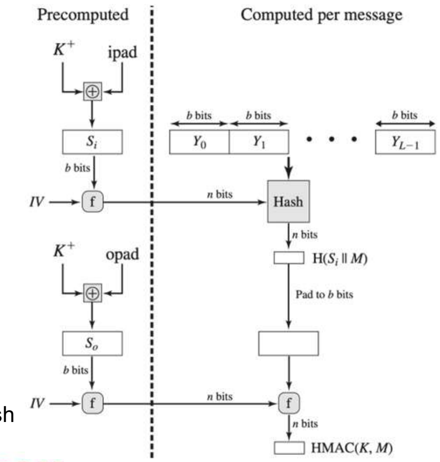
The Algorithm

$$\mathrm{HMAC}(K, M) = \mathrm{H}[K^{+} \oplus \mathrm{opad} \mid\mid \mathrm{H}[(K^{+} \oplus \mathrm{ipad}) \mid\mid M]]$$

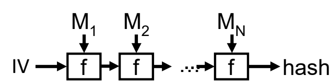
- Append zeros to the left end of K to create K^+.
- $S_i = K^+ \oplus$ ipad (b-bits), read i as inner
- Append M to S_i to produce (S_i ||M)
- Apply H to (S_i ||M).
- $S_o = K^+ \oplus$ opad (b-bits), read o as outer.
- Append H(S_i ||M) to S_o . To get S.
- Compute H(S).

HMAC Precomputation

- Precompute
  - f(IV, $K^+ \oplus$ ipad)
  - f(IV, $K^+ \oplus$ opad)
- For L input blocks, we need only L + 1 compressions



Iterative hash function:



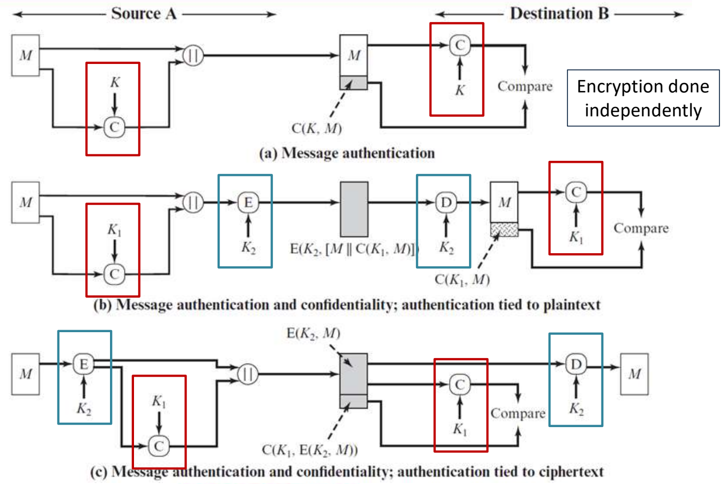
4. Authentication Encryption

- Motivation
  - widely-used cryptographic primitives are (almost always) secure
  - secure encryption + secure authentication ↛ secure combination
  - some security protocols have used cryptographic primitives in an insecure way (*e.g.*, WEP)
- Authenticated encryption: encryption systems that provide both <u>confidentiality</u> and <u>integrity</u>
  ~ block cipher modes that provide confidentiality and integrity

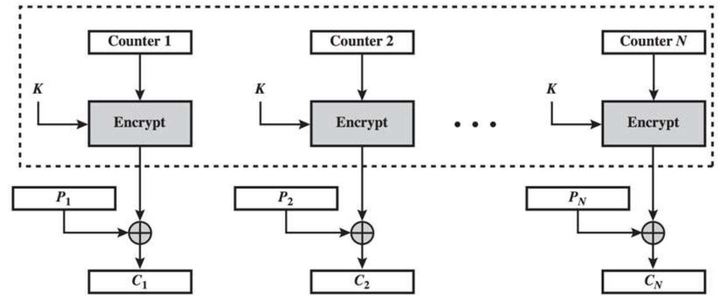
Approaches

- Different Approaches
  - <u>authentication</u> followed by <u>encryption</u> (*e.g.*, SSL/TLS)
  - <u>encryption</u> followed by <u>authentication</u> (*e.g.*, IPSec)
  - <u>independently</u> encrypt and authenticate (*e.g.*, SSH)

MAC + Encryption



Reminder: CTR Mode of Operation



Counter with CBC-MAC (CCM)

- Standardized by NIST in NIST SP 800-38C
  - standard was developed in 2004 for WPA2
  - defined for the AES block cipher
- Encryption: based on the Counter (CTR) block-cipher mode
  - converts a block cipher into a stream cipher
  - very efficient, blocks can be encrypted/decrypted in parallel
- Authentication: based on CBC-MAC message authentication
- Combination: first <u>authenticate</u>, then <u>encrypt</u>
  - compute CBC-MAC of the message, a nonce, and associated data (e.g., protocol headers) that may need integrity protection
  - encrypt message and authentication tag in CTR mode

Galois/Counter Mode (GCM)

- Standardized in 2007 by NIST.
- Encryption: based on the Counter (CTR) block-cipher mode
- Authentication: $GHASH_H(X)$
  - inputs: hash key H, message blocks $X_1, \ldots, X_m$ (in 128-bit blocks)
  - output: $(X_1 \cdot H^m) \oplus (X_2 \cdot H^{m-1}) \oplus \cdots \oplus (X_{m-1} \cdot H^2) \oplus (X_m \cdot H)$ where $\cdot$ is a special multiplication for 128-bit numbers
  - $H^m, H^{m-1}, \ldots, H^2$ can be precomputed, $\cdot$ can be performed in parallel
- Combination: first <u>encrypt</u>, then <u>authenticate</u>
  - authentication includes message length and associated data
- Overall, GCM mode is very efficient and parallelizable
  - widely used (e.g., in HTTPS)

Message Authentication Conclusion

- Message authentication code (MAC)
  - ensures authenticity and integrity
  - requires a secret key
- Based on block-cipher: CMAC
- Based on hash-function: HMAC
- Authenticated encryption: CCM and GCM

Next Topic

- Integrity
- Digital Signature, Key Distribution
- Wifi Security