CMSC 426
Principles of Computer Security

Lecture 12

Hashing and Public Key Infrastructure

Last Class We Covered

- Block cypher modes

- Asymmetric encryption
  - Diffie-Hellman
  - RSA
  - Math (for real this time)

Any Questions from Last Time?

Today's Topics

- Man in the Middle Attacks

- MAC
- Hashing
- HMAC

- Public Key Infrastructure
- Certificates
- Digital signatures

Man in the Middle Attack (MITM)

- Attacker secretly intercepts communication between parties
  - Communication is entirely relayed through the “middle man”

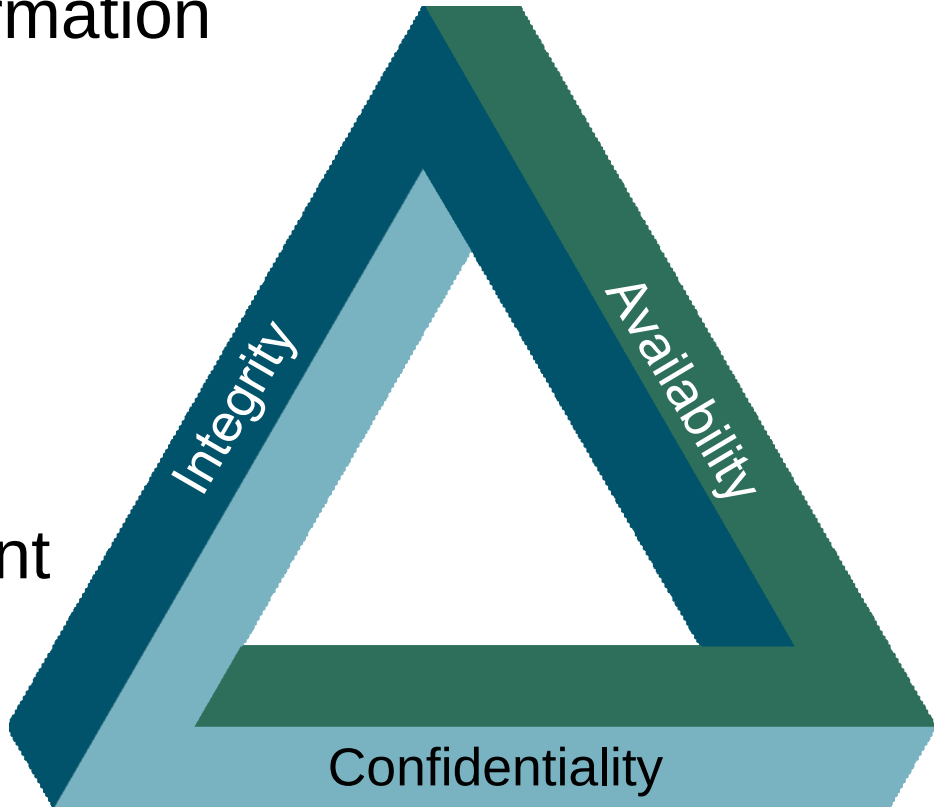
- Attacker can alter course of “conversation” in various ways
  - Not passing on one or more messages
  - Injected new messages
  - Altering intercepted messages

MITM and Diffie-Hellman/RSA

- Diffie-Hellman, attacker sits in the middle and negotiates a key with each side of the communication
  - To Alice, Eve pretends to be Bob; to Bob, Eve pretends to be Alice
  - As Alice and Bob communicate, Eve intercepts the messages, decrypts it, and re-encrypts it with her key
- RSA, attacker performs a similar action, but using her asymmetric public and private keys to re-encrypt
- In either case, Alice and Bob will not notice a difference

Violating the CIA Triad

- ~~Confidentiality~~
  - Users have no control over who sees information
  - Data is available to unauthorized persons
- ~~Integrity~~
  - Attacker can alter the contents of the data
- ~~Availability~~
  - Communication going through is dependent on the attacker forwarding it through
  - Even without modifying the information, can restrict means of communication



Violating Authentication

- Authentication means that users and data can be verified to be genuine (and therefore trusted)
- With the tools we currently know, MITM attacks will allow authentication to be violated as well
  - Simple example: in ECB mode, blocks can be rearranged
- Authentication is distinct from confidentiality
  - Possible to authenticate a message without encrypting it

Message Authentication Codes

Message Authentication

- A MAC algorithm is used to add a small “tag” to the end of a message
  - Message and tag transmitted
  - Receiver re-creates tag, and verifies that message has not been altered
- Even un-encrypted, message can be authenticated

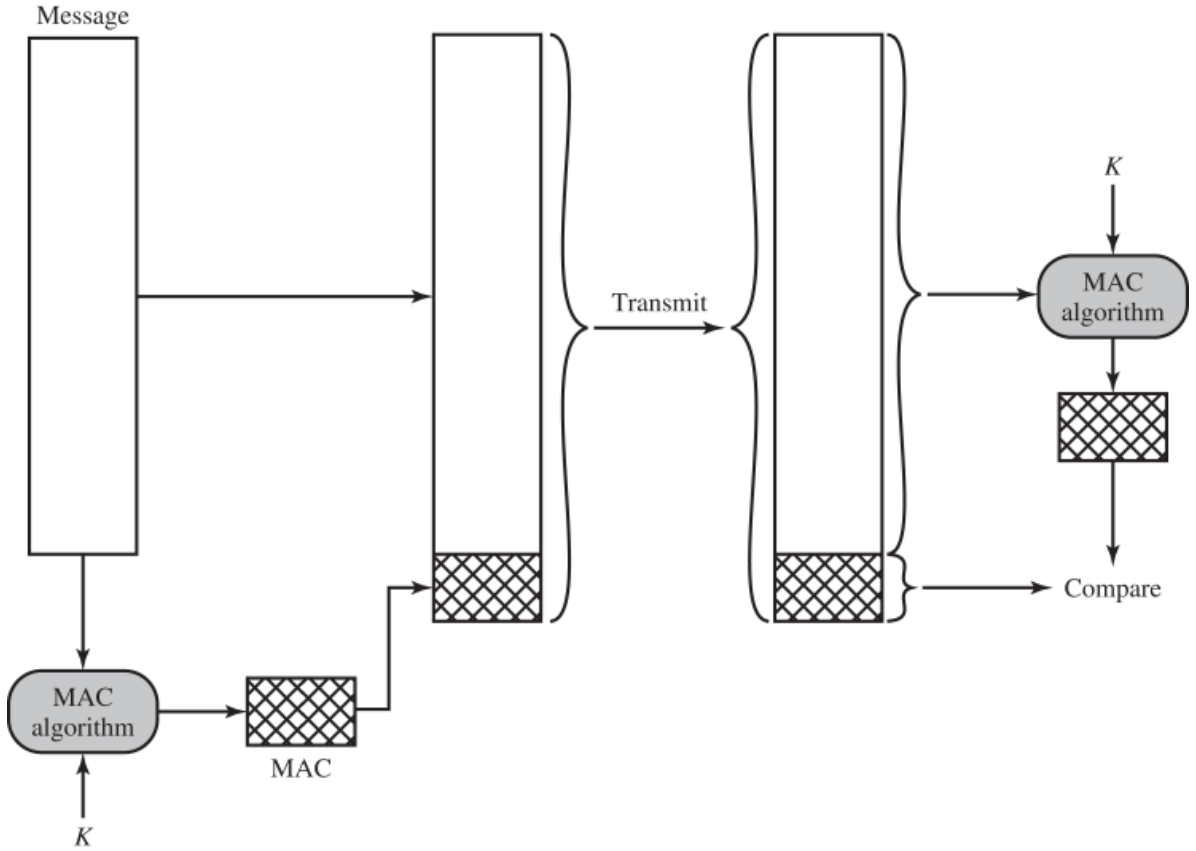


Image taken from Computer Security (Stallings & Brown)

Methods of Generating MACs

- There are a variety of ways to generate MACs
  - Includes using symmetric encryption, public-key encryption, or a secret value
  - Won't go into detail here; read the book if interested

- In order for authenticity to be assured, it must be assumed that only the sender and receiver share the encryption key or secret value
  - We'll discuss how this can be guaranteed later

Hashing

Secure Hash Functions

- Also known as “one-way” hash functions
  - Purpose is to create a “fingerprint” of an input

- To be useful for message authentication, hash functions must:
  1. Be able to be used on data of any size
  2. Must produce a fixed-length output
  3. Must be relatively easy/efficient to produce for any input
  4. Must be resistant with respect to:
     - Pre-image, weak collision, and strong collision

Notation

- H ( )
  - The hash function
- *M*
  - The message/input being hashed
- *h*
  - The resultant hash

- *h* = H(*M*)

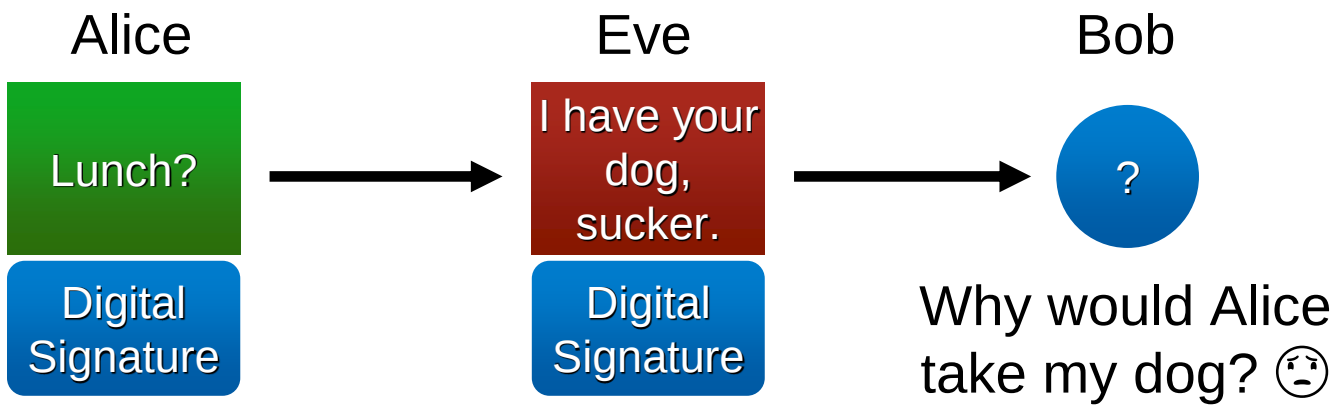
Pre-Image Resistance

- Should be resistant to construction of a “pre-image”
  - In other words, given an already existing *h*, it is not feasible to construct a message *M* such that H(*M*) = *h*
- This property is why they are called “one way” hashes
  - Hash can easily be generated from message, but original message cannot easily be reverse-engineered from resultant hash
- If a hash function is used on a key, this also means the key cannot be recovered by “reversing” the hash

Weak Collision Resistance

- Should be resistant to targeted collision
  - In other words, given an already existing *M*, it is not feasible to construct another message *N* such that H(*M*) = H(*N*)

- If a hash function was not resistant to weak collision, it would be possible for an attacker to replace a message with one with another meaning



Strong Collision Resistance

- Should be resistant to deliberate collision
  - In other words, should not be feasible to create two different messages *M* and *N* such that H(*M*) = H(*N*)

- This is the "stronger" collision resistance, because its requirements allow the attacker to create any two messages
  - With "weak" collision resistance, they had to match an existing one
  - Prevents an attack where Eve creates two messages with the same value, sends the first one to Bob, and then claims that the second message was the one she actually sent (*e.g.*, IOU $10 vs IOU $1000)

Brute Force Costs

- If a hash is ideal, then with an output of size n, it should cost:

- To construct a pre-image:
  - 2^n hash computations (attempts)
- To find a weak collision:
  - 2^n hash computations
- To find a strong collision:
  - $2^{n/2}$ hash computations
    - Lower because of the “birthday problem” – sometimes called a “birthday attack”

Hash Uses

- With encryption: append hash to plaintext *P* before encryption
  - E(*P* ⌢ H(*P*))

We're using *P* and *M* interchangeably here

- Keyed hash: using secret authentication key *K*
  - Authenticate message *M* by appending h_K = H (*M* ⌢ *K*)

- Digital signature: Alice encrypts the hash with her private key
  - *sig* = E(*A*, H(*M*))
  - We'll discuss this in more detail later today

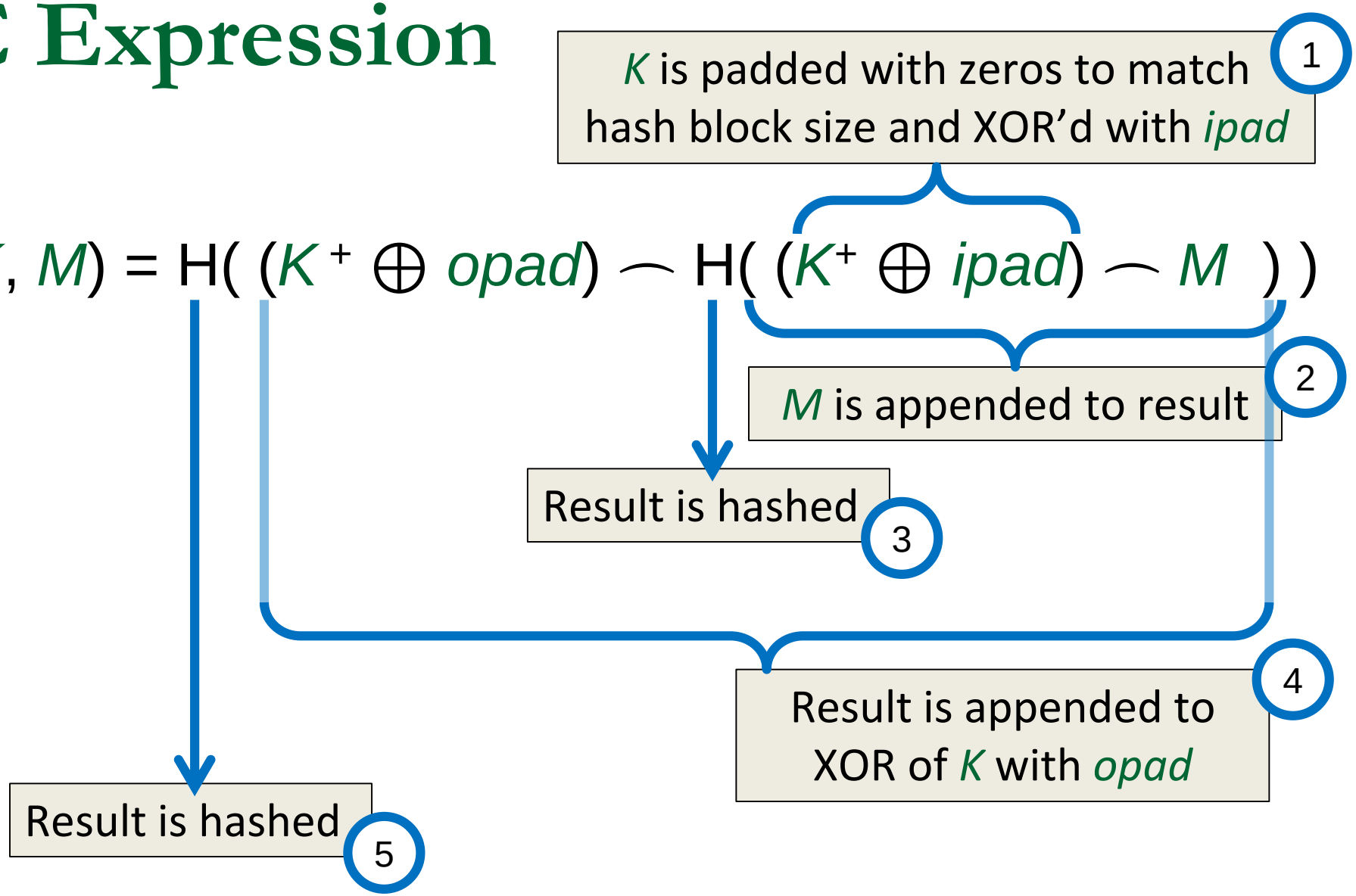
Hash-Based Message Authentication Codes

HMAC

- Improves on the security of basic keyed hash
- Security of HMAC depends solely on the security of the hash function
- HMACs use two “magic” numbers
  - *ipad* = 00110110 (36 in hex)
  - *opad* = 01011100 (5C in hex)
  - These do ~*~math~*~ things (they’re vital to the security proof)

HMAC Expression

- HMAC(K, M) = H(($K^+ \oplus opad$) ⌢ H(($K^+ \oplus ipad$) ⌢ M))



Using an HMAC

- Exactly as a keyed hash would be applied
- Alice and Bob share a secret key *K*
- Alice computes the HMAC of message *M* using key *K*
  - Sends *M* and the HMAC to Bob
- Bob computes the HMAC of the received message
  - Checks that it matches the HMAC sent by Alice
  - If it matches, all is well!

Public Key Infrastructure

What is Public Key Infrastructure?

- "The set of hardware, software, people, policies, and procedures needed to create, manage, store, distribute, and revoke digital certificates based on asymmetric cryptography."

- End goal is a secure, convenient way to acquire public keys
  - Keys are "bound" to identities via certificates

- Helps to alleviate the problems caused by man in the middle attacks

X.509 Certificates

- X.509 is simply the standard that defines the format that most public key certificates take
- Certificates attest to the link between a key and an entity
  - Also contain information about when the certificate is valid, exactly who issued the certificate, and if it has been revoked
- Issued by Certificate Authorities (CAs)
  - There are root CAs and intermediate CAs
  - An operating system/browser trusts a list of CAs by default

Certificate Contents

- The entity the certificate was issued to
- Their public key
- Whether they are a CA or not
  - This is how intermediate CAs are designated
- Serial number of certificate
- Validity window (when it became valid, when it expires)
- The entity that issued the certificate
- How a revoked certificate is designated
  - This part is kind of broken

Alleviating Man in the Middle Attacks

- By combining these different concepts, security can be increased to a point where MITM attacks aren't feasible
  - Unless a Certificate Authority is compromised, in which case ¯_(ツ)_/¯
- Use PKI to ensure public key belongs to person who claims it
  - Ensure authenticity, help with confidentiality
- Use HMAC (or MAC) to verify message is unaltered
  - Ensure integrity

Announcements

-
- Lab 3 will be coming out next week
  - Topic will be cryptanalysis, and no VM will be involved
- Exam grades will go up on Blackboard this weekend

Image Sources

- Penrose triangle (adapted from):
  - https://pixabay.com/en/optical-illusion-illusion-triangle-154081/