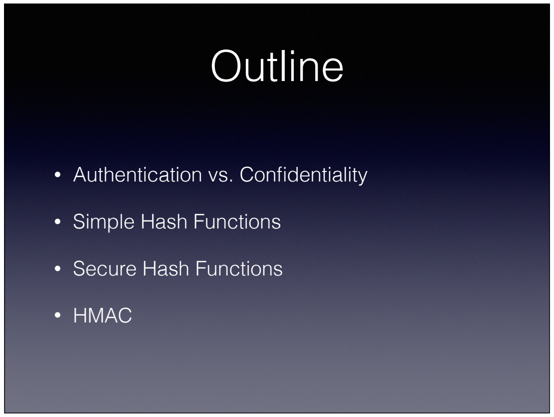
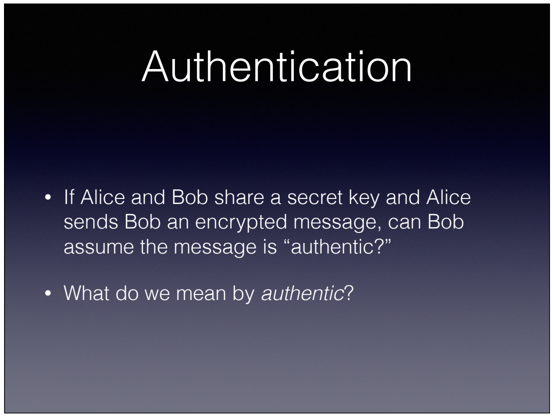
Cryptographic Hashes

CMSC 426/626 - Computer Security
Fall 2014



Outline

- Authentication vs. Confidentiality
- Simple Hash Functions
- Secure Hash Functions
- HMAC



Authentication

- If Alice and Bob share a secret key and Alice sends Bob an encrypted message, can Bob assume the message is “authentic?”
- What do we mean by *authentic*?

Consider

- A **block cipher in ECB** mode…

  An attacker could re-order blocks in the message without affecting Bob's ability to decrypt it.

- A **block cipher in CFB or CTR** mode, or a **stream cipher**…

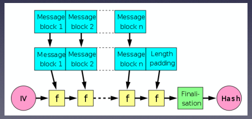
  If the plaintext is highly structured, an attacker can modify the plaintext without decrypting the message.

In fact, it is possible to *authenticate* a message without *encrypting* the message.

Authentication and Confidentiality are distinct.

Hash Functions

- Given a message *M* of arbitrary length, a hash function H produces a fixed size *digest* H*(M).*
- It should be "easy" to compute H*(M)* for any *M.*
- Hashes are an alternative to MACs (we'll cover those later)



(Merkle-Damgard Construction; from Wikipedia, public domain)

Uses for Hashes

- For authentication and integrity.
- **With encryption**: append hash to M before encrypting.
- **Keyed hash**: Alice and Bob share a secret authentication key K; Alice authenticates message M by appending

$$h_K = \mathrm{H}(M \,||\, K)$$

- **Digital signature**: Alice public-key encrypts the hash of M with her private key A_{priv}

$$s = \mathrm{E}[A_{priv}, \mathrm{H}(M)]$$

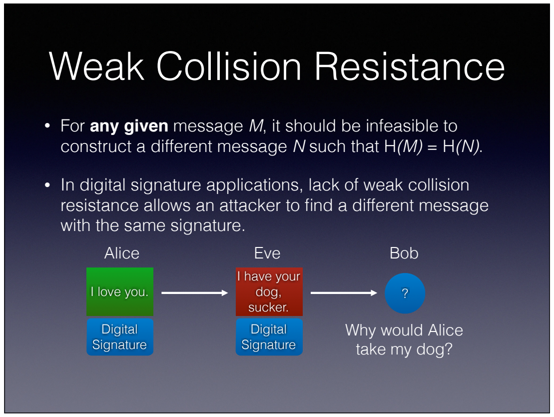
Questions

Suppose Alice and Bob are using a keyed hash scheme with shared key K_{AB}. Alice sends Bob the message M along with $\mathrm{H}(M \,||\, K_{AB})$.

1. How does Bob verify the message is really from Alice?
2. How does Bob verify that the message has not been altered?

Pre-Image Resistance

- For **any** given hash code h, it should be infeasible to construct an M such that $\mathrm{H}(M) = h$.
- In the keyed hash case, pre-image resistance prevents an attacker from recovering $M||K$, and thus K.



Weak Collision Resistance

- For **any given** message *M*, it should be infeasible to construct a different message *N* such that H*(M)* = H*(N)*.
- In digital signature applications, lack of weak collision resistance allows an attacker to find a different message with the same signature.

Alice

I love you.

Digital Signature

→

Eve

I have your dog, sucker.

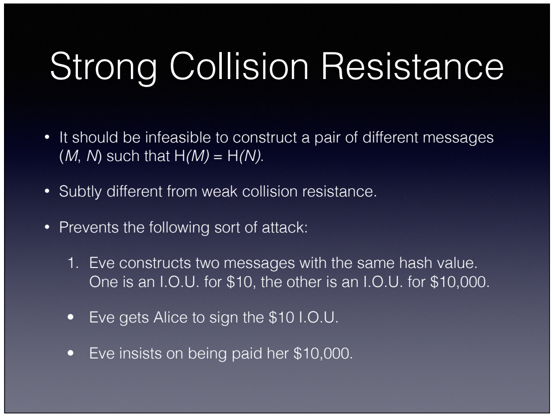
Digital Signature

→

Bob

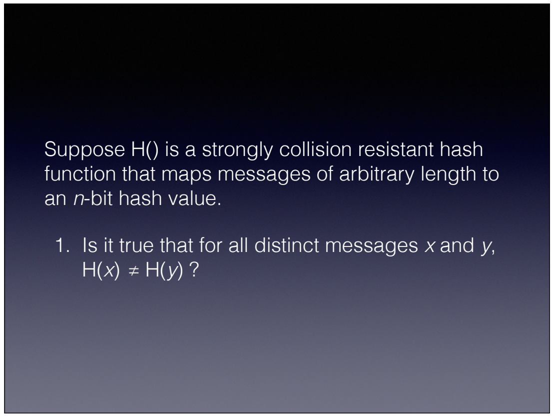
?

Why would Alice take my dog?



Strong Collision Resistance

- It should be infeasible to construct a pair of different messages (*M, N*) such that H*(M)* = H*(N)*.
- Subtly different from weak collision resistance.
- Prevents the following sort of attack:
  1. Eve constructs two messages with the same hash value. One is an I.O.U. for $10, the other is an I.O.U. for $10,000.
  - Eve gets Alice to sign the $10 I.O.U.
  - Eve insists on being paid her $10,000.



Suppose H() is a strongly collision resistant hash function that maps messages of arbitrary length to an *n*-bit hash value.

1. Is it true that for all distinct messages *x* and *y*, H(*x*) ≠ H(*y*) ?

Simple Hash Functions

- Break the M into b-bit blocks $M_1, M_2, \ldots, M_n$.

$$h = M_1 \oplus M_2 \oplus \cdots \oplus M_n$$

- A variation: let $r(x, n)$ denote the left circular shift of x by n bits

$$h = M_1 \oplus r(M_2, 1) \oplus \cdots \oplus r(M_n, n-1)$$

- There are 2^b possible hash codes, so if the message is modified or corrupted, there is probability 2^{-b} that the hash code h will be unchanged.

- Unfortunately, neither of these schemes is **collision resistant** (weak or strong).
- Suppose I construct the following messages:

$$M = M_1, M_2$$

$$N = N_1, N_2, M_1 \oplus M_2 \oplus N_1 \oplus N_2$$

$$N' = N_1, N_2, r(M_1 \oplus r(M_2, 1) \oplus N_1 \oplus r(N_2, 1), -2)$$

- If H is the first simple hash, then $H(M) = H(N)$.
- If H is the variation, then $H(M) = H(N')$

One More Example

- Another simple hash: let a message be represented by a list of integers

$$M = (a_1, a_2, \ldots, a_t)$$

- Let N be a positive integer and define $H(M)$ by

$$h = (a_1 + a_2 + \cdots + a_t) \bmod N$$

- **Is H pre-image resistant?**

Brute Force Costs

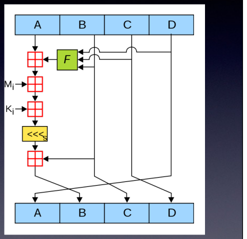
For a hash with digest of size *n*:

- Constructing a **pre-image**: 2^n hash computations
- Finding a **weak collision**: 2^n hash computations
- Finding a **strong collision**: $2^{n/2}$ hash computations (this is due to the birthday problem)

For example, the MD5 message digest is 128 bits, so it should take 2^{64} hash computations to find a strong collision pair.

MD5

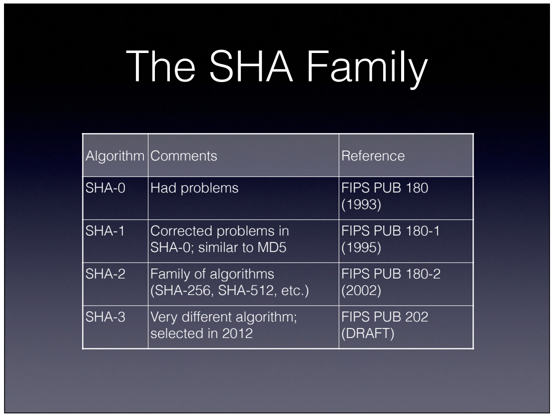
- Ron Rivest, 1992
- Operates on 32-bit words with addition mod 2^{32}
- Message processed in 512-bit “chunks” broken into 16 32-bit words.
- Basic function applied 64 times per chunk.



(from Wikipedia by Surachit; CC A-SA 3.0)

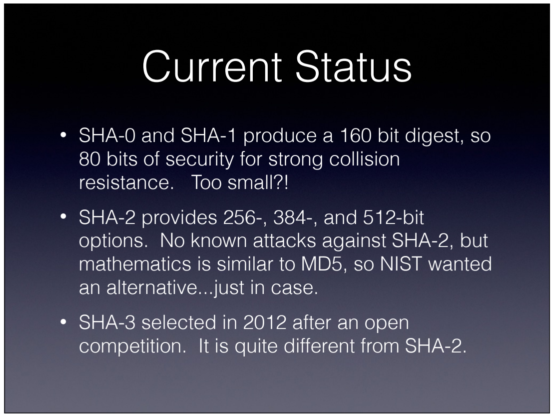
MD5 Attacks

- 2004 - Wang, Fang, Lai, and Yu demonstrate first practical collision
- 2005 - Lenstra, Wang, de Weger produce colliding X.509 certificates
- 2008 - “normal” certificate converted to intermediate CA certificate
- 2012 - Flame malware uses fraudulent MS code signing certificate; constructed using collision



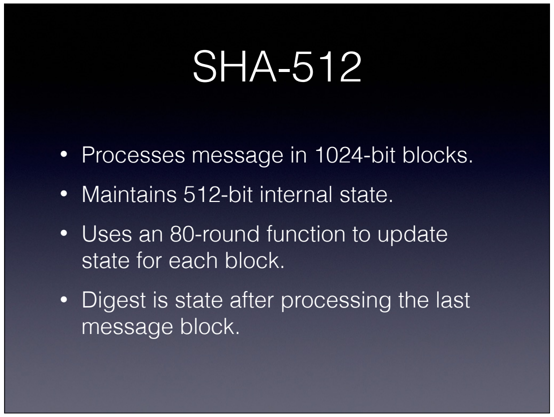
The SHA Family

Algorithm	Comments	Reference
SHA-0	Had problems	FIPS PUB 180 (1993)
SHA-1	Corrected problems in SHA-0; similar to MD5	FIPS PUB 180-1 (1995)
SHA-2	Family of algorithms (SHA-256, SHA-512, etc.)	FIPS PUB 180-2 (2002)
SHA-3	Very different algorithm; selected in 2012	FIPS PUB 202 (DRAFT)



Current Status

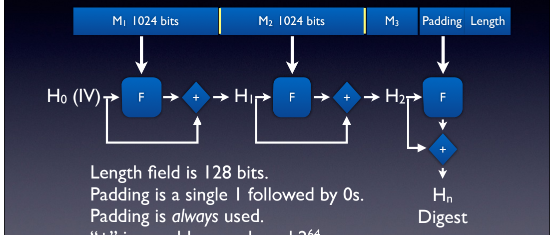
- SHA-0 and SHA-1 produce a 160 bit digest, so 80 bits of security for strong collision resistance. Too small?!
- SHA-2 provides 256-, 384-, and 512-bit options. No known attacks against SHA-2, but mathematics is similar to MD5, so NIST wanted an alternative...just in case.
- SHA-3 selected in 2012 after an open competition. It is quite different from SHA-2.



SHA-512

- Processes message in 1024-bit blocks.
- Maintains 512-bit internal state.
- Uses an 80-round function to update state for each block.
- Digest is state after processing the last message block.

SHA-512
Two full blocks and one partial block



Length field is 128 bits.
Padding is a single 1 followed by 0s.
Padding is *always* used.
"+" is word-by-word mod 2^{64}.

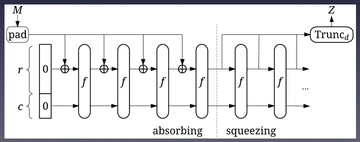
The F-function

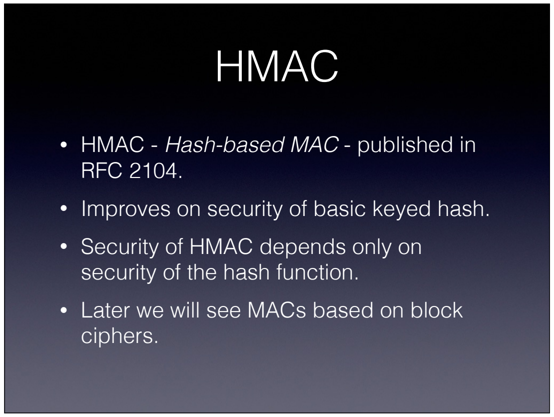
- The F-function consists of 80 rounds.
- Each round involves basic boolean operations (AND, OR, XOR, NOT).
- Each round incorporates a portion of the message block (W_t) and a constant (K_t).

The F-function provides good mixing.
Each digest bit is a function of every input bit.

SHA-3

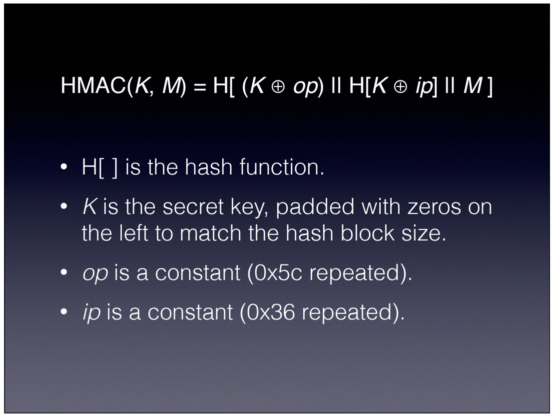
- "Sponge" construction
- *f*-function operates on 1600-bit state
- Message blocks xor-ed with state





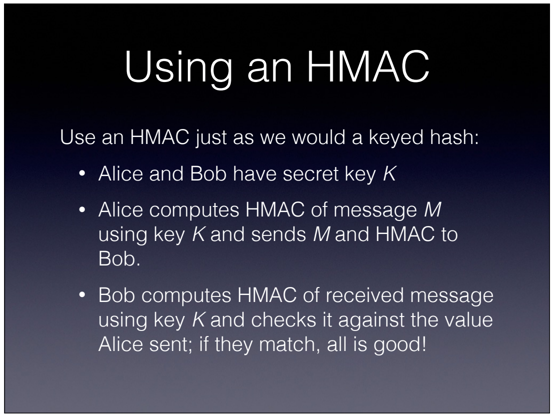
HMAC

- HMAC - *Hash-based MAC* - published in RFC 2104.
- Improves on security of basic keyed hash.
- Security of HMAC depends only on security of the hash function.
- Later we will see MACs based on block ciphers.



$\mathrm{HMAC}(K, M) = \mathrm{H}[\, (K \oplus op) \,||\, \mathrm{H}[K \oplus ip] \,||\, M \,]$

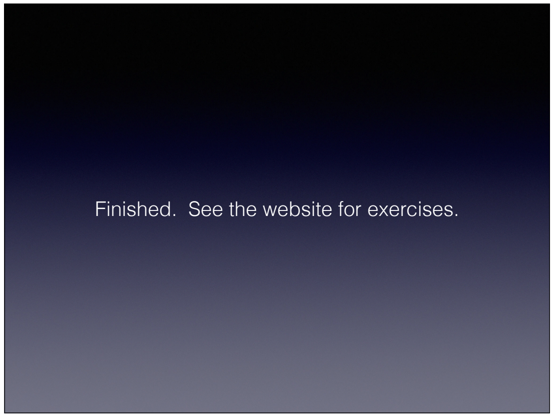
- H[ ] is the hash function.
- *K* is the secret key, padded with zeros on the left to match the hash block size.
- *op* is a constant (0x5c repeated).
- *ip* is a constant (0x36 repeated).



Using an HMAC

Use an HMAC just as we would a keyed hash:

- Alice and Bob have secret key *K*
- Alice computes HMAC of message *M* using key *K* and sends *M* and HMAC to Bob.
- Bob computes HMAC of received message using key *K* and checks it against the value Alice sent; if they match, all is good!



Finished. See the website for exercises.