CSE 484 / CSE M 584: Computer Security and Privacy

Cryptography: Hash Functions and MACs [continued]
Asymmetric Cryptography [start]

Spring 2015

Franziska (Franzi) Roesner
franzi@cs.washington.edu

Thanks to Dan Boneh, Dieter Gollmann, Dan Halperin, Yoshi Kohno, John Manferdelli, John Mitchell, Vitaly Shmatikov, Bennet Yee, and many others for sample slides and materials ...
Admin

- Checkpoint for lab #1 due today @5pm
  - Submit md5 hashes to Catalyst dropbox

- Homework #2 (on crypto) will be out soon

- Today: Finish hash functions, start public key crypto
- Wednesday: Finish public key crypto, crypto misc
- Friday: Finish crypto, start web security (if time)
Follow-up: CBC-MAC

- Not secure when system may MAC messages of different lengths.
- NIST recommends a derivative called CMAC [FYI only]
Back to Hash Functions
Hash Functions: Main Idea

- Hash function $H$ is a lossy compression function
  - Collision: $h(x)=h(x')$ for distinct inputs $x$, $x'$
- $H(x)$ should look “random”
  - Every bit (almost) equally likely to be 0 or 1
- Cryptographic hash function needs a few properties...
Property 3: Weak Collision Resistance

• Given randomly chosen $x$, hard to find $x'$ such that $h(x) = h(x')$
  – Attacker must find collision for a specific $x$. By contrast, to break collision resistance it is enough to find any collision.
  – Brute-force attack requires $O(2^n)$ time

• Weak collision resistance does not imply collision resistance.
Hashing vs. Encryption

• Hashing is one-way. There is no “un-hashing”
  – A ciphertext can be decrypted with a decryption key...
    hashes have no equivalent of “decryption”

• Hash(x) looks “random” but can be compared for equality with Hash(x’)
  – Hash the same input twice ➞ same hash value
  – Encrypt the same input twice ➞ different ciphertexts

• Cryptographic hashes are also known as “cryptographic checksums” or “message digests”
Application: Password Hashing

• Instead of user password, store $\text{hash(password)}$

• When user enters a password, compute its hash and compare with the entry in the password file
  – System does not store actual passwords!
  – Cannot go from hash to password!

• Why is hashing better than encryption here?

• Does hashing protect weak, easily guessable passwords?
Application: Software Integrity

Goal: Software manufacturer wants to ensure file is received by users without modification.

Idea: given goodFile and hash(goodFile), very hard to find badFile such that hash(goodFile)=hash(badFile)
Which Property Do We Need?

• UNIX passwords stored as hash(password)
  – One-wayness: hard to recover the/a valid password

• Integrity of software distribution
  – Weak collision resistance
  – But software images are not really random… may need full collision resistance if considering malicious developers

• Auction bidding
  – Alice wants to bid B, sends H(B), later reveals B
  – One-wayness: rival bidders should not recover B (this may mean that she needs to hash some randomness with B too)
  – Collision resistance: Alice should not be able to change her mind to bid B’ such that H(B)=H(B’)

Common Hash Functions

• MD5
  – 128-bit output
  – Designed by Ron Rivest, used very widely
  – Collision-resistance broken (summer of 2004)

• RIPEMD-160
  – 160-bit variant of MD5

• SHA-1 (Secure Hash Algorithm)
  – 160-bit output
  – US government (NIST) standard as of 1993-95
  – Also recently broken! (Theoretically -- not practical.)

• SHA-256, SHA-512, SHA-224, SHA-384

• SHA-3: Still in draft – not an official standard yet
Basic Structure of SHA-1 [FYI only]

- Split message into 512-bit blocks
- 160-bit buffer (5 registers) initialized with magic values
- Compression function:
  - Applied to each 512-bit block and current 160-bit buffer
  - This is the heart of SHA-1

Against padding attacks
Message length (K mod 2^64)

\[ L \times 512 \text{ bits} = N \times 32 \text{ bits} \]

Padding (1 to 512 bits)

160-bit buffer (5 registers) initialized with magic values
How Strong is SHA-1?

• Every bit of output depends on every bit of input
  – Very important property for collision-resistance
• Brute-force inversion requires $2^{160}$ ops, birthday attack on collision resistance requires $2^{80}$ ops
• Some weaknesses, e.g., collisions can be found in $2^{63}$ ops (2005)
Recall: Achieving Integrity

Message authentication schemes: A tool for protecting integrity.

**MAC:** message authentication code (sometimes called a “tag”)

Integrity and authentication: only someone who knows KEY can compute correct MAC for a given message.
HMAC

• Construct MAC from a cryptographic hash function
  – Invented by Bellare, Canetti, and Krawczyk (1996)
  – Used in SSL/TLS, mandatory for IPsec

• Why not encryption?
  – Hashing is faster than block ciphers in software
  – Can easily replace one hash function with another
  – There used to be US export restrictions on encryption
Structure of HMAC [FYI only]

- Secret key padded to block size
- Magic value (flips half of key bits)
- Another magic value (flips different key bits)
- Block size of embedded hash function
- Embedded hash function
- "Black box": can use this HMAC construction with any hash function
- hash(key, hash(key, message))
Authenticated Encryption

- What if we want **both** privacy and integrity?
- Natural approach: combine encryption scheme and a **MAC**.
- But be careful!
  - Obvious approach: Encrypt-and-MAC
  - Problem: MAC is deterministic! same plaintext → same MAC
Authenticated Encryption

• Instead: Encrypt then MAC.

• (Not as good: MAC-then-Encrypt)
Asymmetric (Public Key) Cryptography
Reminder: Symmetric Cryptography

• 1 secret key (or 2 or ...), shared between sender/receiver
• Repeat fast and simple operations lots of times (rounds) to mix up key and ciphertext
• Why do we think it is secure? (simplistic)
  – Lots of heuristic arguments
    • If we do lots and lots and lots of mixing, no simple formula (and reversible) describing the whole process (cryptographic weakness).
    • Mix in ways we think it’s hard to short-circuit all the rounds. Especially non-linear mixing, e.g., S-boxes.
  – Some math gives us confidence in these assumptions
Public Key Crypto: Basic Problem

Given: Everybody knows Bob’s public key
Only Bob knows the corresponding private key

Goals: 1. Alice wants to send a secret message to Bob
2. Bob wants to authenticate himself
Public Key Cryptography

• Everyone has 1 private key and 1 public key
  – Or 2 private and 2 public, when considering both encryption and authentication

• Mathematical relationship between private and public keys

• Why do we think it is secure? (simplistic)
  – Relies entirely on problems we believe are “hard”
Applications of Public Key Crypto

• Encryption for confidentiality
  – Anyone can encrypt a message
    • With symmetric crypto, must know secret key to encrypt
  – Only someone who knows private key can decrypt
  – Key management is simpler (or at least different)
    • Secret is stored only at one site: good for open environments

• Digital signatures for authentication
  – Can “sign” a message with your private key

• Session key establishment
  – Exchange messages to create a secret session key
  – Then switch to symmetric cryptography (why?)