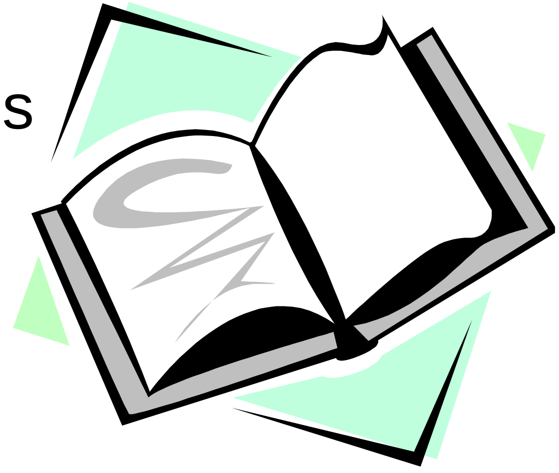
Cryptography
CS 555

Topic 14: CBC-MAC & Hash Functions

Outline and Readings

- Outline
  - CBC-MAC
  - Collision-resistant hash functions
  - Applications of MAC and hash functions



- Readings:
  - Katz and Lindell: : 4.5,4.6

Basic CBC-MAC (secure for fixed-length messages)

- Given a PRF $F:\{0,1\}^n \times \{0,1\}^n \rightarrow \{0,1\}^*$, fix a length function $\ell(n)$, basic CBC-MAC is
  - **Mac**$_k$(m) m is of length $\ell(n) \cdot n$
    - Divide m into $m_1, \ldots, m_\ell$
    - Set $t_0 := 0^n$
    - For $i=1$ to ℓ, set $t_i := F_k(t_{i-1} \oplus m_i)$
    - Output t_ℓ
  - **Vrfy**(k, m, t) on input m of length $\ell(n) \cdot n$, check whether $t =$ **Mac**$_k$(m)
- When F is a block cipher, this is similar to CBC encryption with IV= 0^n, and using last block as tag
- Why is this insecure for variable messages?

Security of Basic CBC-MAC

- The basic CBC-MAC is a fixed-length MAC that is existential unforgerable under an adaptive chosen-message attack assuming that F is PRF.

- CBC-MAC differs with CBC encryption
  - Fixed IV vs random IV
  - Outputting last block vs. all blocks
    - Outputting more than one ciphertext blocks is no longer a secure MAC. Why?

Secure MAC for Variable-length Msgs

- Several constructions are proven secure
  - Set $k_\ell := F_k(\ell)$, then compute basic CBC-MAC with k_ℓ
  - Prepend message with its length encoded as an n-bit string, then apply basic CBC-MAC
    - Append message length is insecure, why?
  - Uses two keys, compute basic CBC-MAC of m using k1 as t, then compute output tag $F_{k2}(t)$

Hash Functions

- A hash function maps/compresses messages of arbitrary lengths to a m-bit output
  - output known as the fingerprint or the message digest
- What is an example of hash functions?
  - Given a hash function that maps Strings to integers in [0,2^{32}-1]
- A hash function is a many-to-one function, so collisions must happen.
- Hash functions are used in a number of data structures
  - Good hash functions have few collisions
- Cryptographic hash functions are hash functions with additional security requirements

Security Requirements for Cryptographic Hash Functions

Given a function $h: X \to Y$, then we say that h is:

- preimage resistant (one-way):

  if given $y \in Y$ it is computationally infeasible to find a value $x \in X$ s.t. $h(x) = y$

- 2-nd preimage resistant (weak collision resistant):

  if given $x \in X$ it is computationally infeasible to find a value $x' \in X$, s.t. $x' \neq x$ and $h(x') = h(x)$

- collision resistant (strong collision resistant):

  if it is computationally infeasible to find two distinct values $x', x \in X$, s.t. $h(x') = h(x)$

Bruteforce Attacks on Hash Functions

- Attacking one-wayness
  - Goal: given $h: X \to Y$, $y \in Y$, find x such that h(x)=y
  - Algorithm:
    - pick a random value x in X, check if h(x)=y, if h(x)=y, returns x; otherwise iterate
    - after failing q iterations, return fail
  - The average-case success probability is

$$\varepsilon = 1 - \left(1 - \frac{1}{|Y|}\right)^q \approx \frac{q}{|Y|} \quad \text{when } q << |Y|$$

  - Let $|Y| = 2^m$, $q = 2^{m-1}$ then, ε is 1/sqrt(e) about 0.6

Bruteforce Attacks on Hash Functions

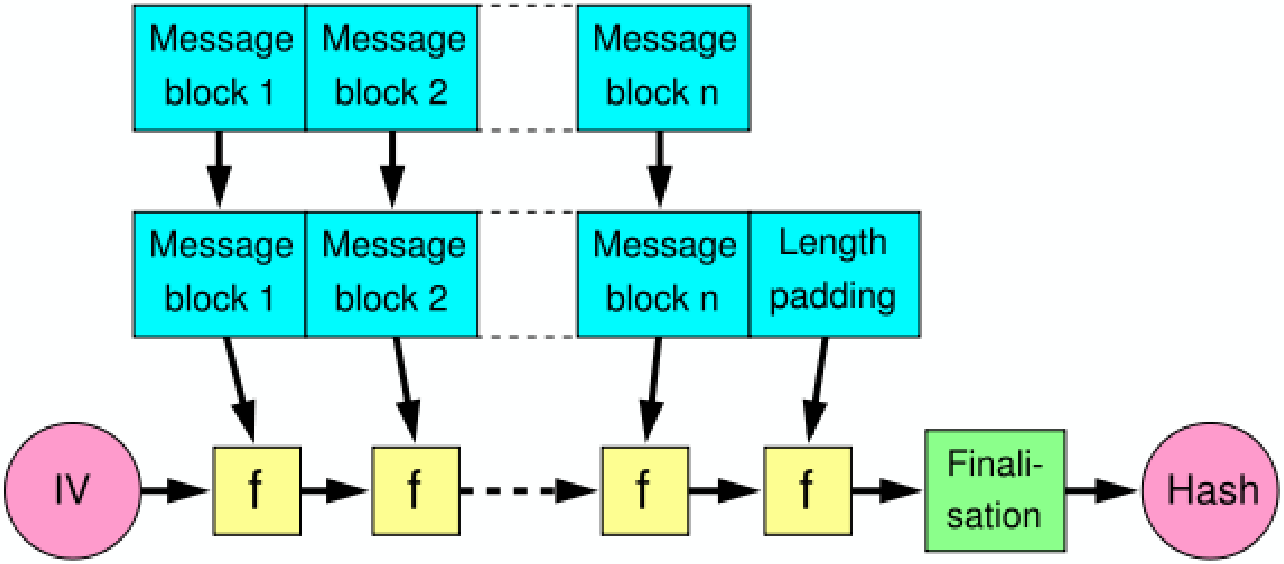
- Attacking collision resistance
  - Goal: given h, find x, x' such that h(x)=h(x')
  - Algorithm: pick a random set X_0 of q values in X
    for each $x \in X_0$, computes $y_x = h(x)$
    if $y_x = y_{x'}$ for some $x' \neq x$ then return (x,x') else fail
  - The average success probability is $1 - e^{-\frac{q(q-1)}{2|Y|}}$
  - Let $|Y| = 2^m$, to get ε to be close to 0.5, $q \approx 2^{m/2}$
  - This is known as the birthday attack.

Well Known Hash Functions

- MD5
  - output 128 bits
  - collision resistance completely broken by Prof. Xiaoyun Wang and others from in China in 2004
- SHA1
  - output 160 bits
  - no collision found yet, but method exist to find collisions in less than 2^80
  - considered insecure for collision resistance
  - one-wayness still holds
- SHA2 (SHA-224, SHA-256, SHA-384, SHA-512)
  - outputs 224, 256, 384, and 512 bits, respectively
  - No real security concerns yet

Merkle-Damgard Construction for Hash Functions

- Message is divided into fixed-size blocks and padded
- Uses a compression function f, which takes a chaining variable (of size of hash output) and a message block, and outputs the next chaining variable
- Final chaining variable is the hash value



Security of Merkle-Damgard Construction

- Finding a collision against the hash function implies finding a collision against the compression function
- A compression function that takes a chaining variable and a block of msg is often similar to a block cipher using the msg as round keys to encrypt the chaining variable
  - Finding collisions is similar to related-key attacks against block ciphers, something that is not very well-understood.

Related-Key Attacks Against Block Ciphers

- Attacker ensures two keys that satisfy that some relationship, e.g., k1⊕k2=p, and then use chosen plaintext attacks to obtain ciphertexts of msgs under both keys
- Recent paper claims that AES-256 with 9 rounds can be broken with 2^{39} ciphertexts and 2^{39} time
  - AES-256 uses 14 rounds
  - AES-128 and AES-192 are less vulnerable to related key attacks, because shorter keys force more permutation in generating sub-keys

NIST SHA-3 Competition

- NIST is having an ongoing competition for SHA-3, the next generation of standard hash algorithms
- 2007: Request for submissions of new hash functions
- 2008: Submissions deadline. Received 64 entries. Announced first-round selections of 51 candidates.
- 2009: After First SHA-3 candidate conference in Feb, announced 14 Second Round Candidates in July.
- 2010: After one year public review of the algorithms, hold second SHA-3 candidate conference in Aug. Announced 5 Third-round candidates in Dec.
- 2011: Public comment for final round
- 2012: Hold Final hash candidate conference. Draft standard, wait for comments, and submit recommendation.

Choosing the length of Hash outputs

- The Weakest Link Principle:
  – A system is only as secure as its weakest link.
- Hence all links in a system should have similar levels of security.
- Because of the birthday attack, the length of hash outputs in general should double the key length of block ciphers
  – SHA-224 matches the 112-bit strength of triple-DES
  – SHA-256, SHA-384, SHA-512 match the new key lengths (128,192,256) in AES

Application of Hash Function and MAC

- Using Passwords Over Insecure Channels

- One-time passwords
  - Each password is used only once
  - Defend against passive adversaries who eavesdrop and later attempt to impersonate

- Challenge response
  - Send a response related to both the password and a challenge

How to do One-Time Password

- Shared lists of one-time passwords
- Time-synchronized OTP
  - E.g., use $MAC_K(t)$, where K is shared secret, and t is current time
- Using a hash chain (Lamport)
  - h(s), h(h(s), h(h(h(s))), …, $h^{1000}(s)$
  - use these values as passwords in reverse order

Lamport's One-Time Password: Using a Hash Chain

- One-time setup:
  - A selects a value w, a hash function H(), and an integer t, computes $w_0 = H^t(w)$ and sends w_0 to B
  - B stores w_0
- Protocol: to identify to B for the i^{th} time, $1 \leq i \leq t$
  - A sends to B: A, i, $w_i = H^{t-i}(w)$
  - B checks $i = i_A$, $H(w_i) = w_{i-1}$
  - if both holds, $i_A = i_A+1$

Challenge-Response Protocols

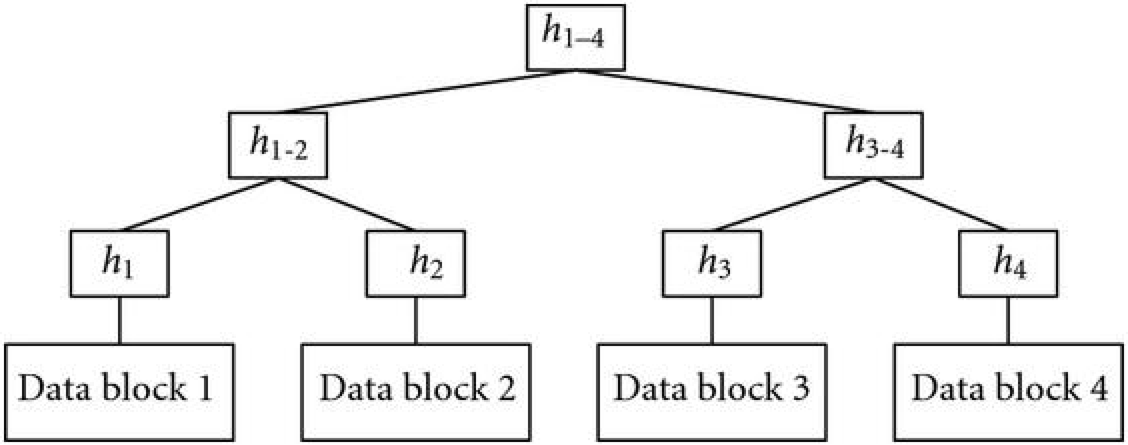
- Goal: one entity authenticates to other entity proving the knowledge of a secret, ‘challenge’
- Approach: Use time-variant parameters to prevent replay, interleaving attacks, provide uniqueness and timeliness
  - e.g., nonce (used only once), timestamps

Challenge-response based on symmetric-key crypto

- Unilateral authentication, timestamp-based
  - A to B: $MAC_K(t_A, B)$
- Unilateral authentication, nonce-based
  - B to A: r_B
  - A to B: $MAC_K(r_B, B)$
- Mutual authentication, nonce-based
  - B to A: r_B
  - A to B: r_A, $MAC_K(r_A, r_B, B)$
  - B to A: $MAC_K(r_B, r_A)$

Authenticated Data Structure with Merkle Hash Tree

- An Authenticated Data Structure enables an untrusted party to answer queries on behalf of data owner
  - Each query answer comes with a proof of correctness
  - Useful in data outsourcing, database as a service, cloud computing, etc.
- A merkle hash tree enables proof of size O(log n)



Other Usages of Cryptographic Hash Functions

- Software integrity
  - E.g., tripwire

- Timestamping
  - How to prove that you have discovered a secret on an earlier date without disclosing it?

Coming Attractions …

- HMAC
- CCA-Secure encryption
- Combining encryption with MAC

- Reading: Katz & Lindell: 4.7,4.8,4.9