Message Authentication and Hash Functions

CSCI 454/554

Security Requirements

- disclosure
- traffic analysis
- masquerade
- content modification
- sequence modification
- timing modification
- source repudiation
- destination repudiation
Message Authentication

- Message authentication is concerned with:
  - protecting the integrity of a message
  - validating identity of originator

- Authenticator and Authentication Protocol

- Producing authenticator (three alternative functions)
  - message encryption
  - message authentication code (MAC)
  - hash function

Message Encryption

- Message encryption by itself also provides a measure of authentication
  - symmetric and public-key encryption schemes

- If symmetric encryption is used then:
  - receiver knows sender must have created it
  - since only sender and receiver know key used
  - know content cannot have been altered
  - plaintext has suitable structure, eg. FCS (frame check sequence) or checksum, to detect any changes
**Message Encryption**

- if public-key encryption is used:
  - encryption provides no authentication of sender
  - since anyone potentially knows public-key
  - however if
    - sender **signs** message using their private-key
    - then encrypts with recipients public key
    - have both secrecy and authentication
  - again need to recognize corrupted messages
  - but at cost of two public-key uses on message

**Message Authentication Code (MAC)**

- generated by an algorithm that creates a small fixed-sized block
  - depending on both message and key
  - like encryption though *need not be reversible*
- appended to message as a **cryptographic checksum**
- receiver performs same computation on message and checks it matches the MAC
- provides assurance that message is unaltered and comes from sender
Message Authentication Code

- can also use encryption for secrecy
  - generally use separate keys for each
  - can compute MAC either before or after encryption
  - is generally regarded as better done before
- why use a MAC?
  - sometimes only authentication is needed
  - sometimes need authentication to persist longer than the encryption (eg. archival use)
- note that a MAC is not a digital signature
MAC Properties

- a MAC is a cryptographic checksum
  \[ \text{MAC} = C_K(M) \]
  - condenses a variable-length message \( M \)
  - using a secret key \( K \)
  - a fixed-sized authenticator
- is a many-to-one function
  - potentially several messages may have same MAC
  - but finding these needs to be very difficult

Requirements for MACs

- taking into account the types of attacks
- need the MAC to satisfy the following:
  1. knowing a message and MAC, is infeasible to find another message with same MAC
  2. MACs should be uniformly distributed
  3. MAC should depend equally on all bits of the message
Using Symmetric Ciphers for MACs

- can use any block cipher chaining mode and use final block as a MAC
- **Data Authentication Algorithm (DAA)** is a widely used MAC based on DES-CBC
  - using IV=0 and zero-pad of final block
  - encrypt message using DES in CBC mode
  - and send just the final block as the MAC
    - or the leftmost M bits (16 ≤ M ≤ 64) of final block

Hash Functions

- condenses arbitrary message to fixed size
- usually assume that the hash function is public and not keyed
  - vs. MAC is keyed
- hash used to detect changes to message
- can use in various ways with message
Digital Signatures

Hash Function Properties

- a Hash Function produces a fingerprint of some file/message/data
  - Hash code: $h = H(M)$
    - condenses a variable-length message $M$
    - a fixed-sized fingerprint
- assumed to be public
## Requirements for Hash Functions

1. can be applied to any sized message \( M \)
2. produces fixed-length output \( h \)
3. is easy to compute \( h = H(M) \) for any message \( M \)
4. given \( h \) is infeasible to find \( x \) s.t. \( H(x) = h \)
   - one-way property
5. given \( x \) is infeasible to find \( y \) s.t. \( H(y) = H(x) \)
   - weak collision resistance
6. is infeasible to find any \( x, y \) s.t. \( H(y) = H(x) \)
   - strong collision resistance

## How Hash Functions Works

- **General principle:**
  - the message is viewed as a sequence of \( b \)-bit blocks
  - the message is processed one block at a time in an iterative way to produce an \( n \)-bit hash code

- **Iterated hash function**
  - \( M = (M_1, M_2, M_3, \ldots M_n) \);
  - \( h_i = H(h_{i-1}, M_i) \);
  - \( H(M) = h_n \)
Different Hash Functions

- simple HF: based on XOR of message blocks
  - not secure since message can be easily changed with the same hash code
- use block ciphers as hash functions
  - using \( h_0 = 0 \) and zero-pad of final block
  - compute: \( h_i = E \left[ M_i, h_{i-1} \right] \)
  - resulting hash is not big enough (64-bit)
    - both due to direct birthday attack
    - and to “meet-in-the-middle” attack
  - other variants also susceptible to attack

Birthday Paradox
Birthday Attacks

- might think a 64-bit hash is secure
- but by Birthday Paradox is not
- birthday attack works thus:
  - Eve generates $2^{m/2}$ variations of a valid message all with essentially the same meaning
  - Eve also generates $2^{m/2}$ variations of a desired fraudulent message
  - two sets of messages are compared to find pair with same hash (probability $> 0.5$ by birthday paradox)
  - have user sign the valid message, then substitute the forgery which will have a valid signature
- conclusion is that need to use larger Hash Codes

Hash Functions & MAC Security

- brute-force attacks exploiting
  - strong collision resistance hash have cost $2^{m/2}$
    - 128-bit hash looks vulnerable, 160-bit better
  - MACs with known message-MAC pairs
    - can either attack keyspace (cf key search) or MAC
    - at least 128-bit MAC is needed for security
- cryptanalytic attacks exploit structure
  - typically focus on collisions in function h
Summary

- have considered:
  - message authentication usage
  - message encryption
  - MACs
  - hash functions
  - general approach & security