MACs and Digital Signatures

Prof. Clarkson
Spring 2016
Review: Encryption

• We can now protect confidentiality of messages against Dolev-Yao attacker
  – efficiently, thanks to hybrid of symmetric and asymmetric encryption
  – assuming existence of phonebook of public keys

• But what about integrity...?
Protection of integrity

• **Threat:** attacker who controls the network
  – Dolev-Yao model: attacker can read, modify, delete messages

• **Harm:** information contained in messages can be changed by attacker (violating integrity)

• **Vulnerability:** communication channel between sender and receiver can be controlled by other principals

• **Countermeasure:** message authentication codes (MACs)
  – beware: not the same "MAC" as mandatory access control nor media access control
MESSAGE AUTHENTICATION CODES
MAC algorithms

- Gen(len): generate a key of length len
- MAC(m; k): produce a tag for message m with key k
  - message may be arbitrary size
  - tag is typically fixed length
Security of MAC

• Must be hard to forge tag for a message without knowledge of key
  – message of attackers choice? vs.
  – message that attacker cannot control

• Even if in possession of multiple (message, tag) pairs for that key
Protocol to exchange MAC'd message

0. \( k = \text{Gen(len)} \)
1. \( A: t = \text{MAC}(m; k) \)
2. \( A \rightarrow B: m, t \)
3. \( B: \text{verify } t = \text{MAC}(m; k) \)

- Both principals use the same shared key: symmetric key cryptography
- Message is sent in plaintext: no protection of confidentiality
- Goal is to detect modification not prevent
- Both principals run same algorithm
  - unlike encryption scheme
  - though for some block ciphers Enc and Dec are effectively the same
Examples of MACs

• CBC-MAC
  – Parameterized on a block cipher
  – Core idea: encrypt message with block cipher in CBC mode, use very last ciphertext block as the tag

• HMAC
  – Parameterized on a hash function
  – Core idea: hash message together with key
  – Your everyday hash function isn’t good enough…
HASH FUNCTIONS
Hash functions

• Input: arbitrary size bit string
• Output: fixed size bit string
  – compression: many inputs map to same output, hence creating collision
  – for use with hash tables, diffusion: minimize collisions (and clustering)
Cryptographic hash functions

- Aka message digest
- Stronger requirements than (plain old) hash functions
- **Goal:** hash is compact representation of original like a fingerprint
  - Hard to find 2 people with same fingerprint
  - Whether you get to pick pairs of people, or whether you start with one person and find another
    - ...collision-resistant
  - Given person easy to get fingerprint
  - Given fingerprint hard to find person
    - ...one-way
Real world hash functions

• **MD5:** Ron Rivest (1991)
  - 128 bit output
  - Collision resistance broken 2004-8
  - Can now find collisions in seconds
  - Don't use it

• **SHA-1:** NSA (1995)
  - 160 bit output
  - Theoretical attacks that reduce strength to less than 80 bits
  - On its way out, yet many browsers continue to accept it
Real world hash functions

- **SHA-2**: NSA (2001)
  - Family of algorithms with output sizes \{224, 256, 384, 512\}
  - In principle, could one day be vulnerable to similar attacks as SHA-1

- **SHA-3**: public competition (won in 2012, standardized by NIST in 2015)
  - Same output sizes as SHA-2
  - Plus a variable-length output called SHAKE
Strength of hash functions

• **Birthday attack:** generic attack based on...
  – **Birthday paradox:** probability of two people in group sharing same birthday (a collision) is much higher than intuition might suggest
  – So collisions are easier to find than you might expect

• **Strength of hash function is thus** (at most) about half of output length
CONFIDENTIALITY & INTEGRITY
Encryption and integrity

IF I ENCRYPT MESSAGE

WOULDN'T CHANGES DECRYPT TO NONSENSE?
Encryption and integrity

**NO!**

- Plaintext block might be random number, and recipient has no way to detect change in random number
- Attacker might substitute ciphertext from another execution of same protocol
- In some block modes (e.g., CTR), it's easy to flip individual bits — change "admin=0" to "admin=1"
- In some block modes (e.g., CBC), it's easy to truncate blocks from beginning of message
- ...

So you can't get C+I solely from encryption
Authenticated encryption

• Newer block cipher modes designed to provide confidentiality and integrity
  – **OCB**: Offset Codebook Mode
  – **CCM**: Counter with CBC-MAC Mode
  – **GCM**: Galois Counter Mode

• Or, you could combine encryption schemes with MAC schemes...
Encrypt and MAC

0. k_E = Gen_E(len)
   k_M = Gen_M(len)

1. A: c = Enc(m; k_E)
   t = MAC(m; k_M)

2. A -> B: c, t

3. B: m' = Dec(c; k_E)
   t' = MAC(m'; k_M)
   if t = t'
      then output m'
   else abort
Encrypt and MAC

• **Pro:** can compute Enc and MAC in parallel
• **Con:** MAC must protect confidentiality
  (not actually a requirement we ever stipulated)

• Example: **ssh** (Secure Shell) protocol
  – recommends AES-128-CBC for encryption
  – recommends HMAC with SHA-2 for MAC
Aside: Key reuse

- Never use same key for both encryption and MAC schemes
- **Principle:** every key in system should have unique purpose
Encrypt then MAC

1. A: $c = \text{Enc}(m; k_E)$
   \[ t = \text{MAC}(c; k_M) \]
2. A $\rightarrow$ B: $c, t$
3. B: $t' = \text{MAC}(c; k_M)$
   if $t = t'$
      then output $\text{Dec}(c; k_E)$
   else abort
Encrypt then MAC

• **Pro:** provably most secure of three options
  [Bellare & Namprepre 2001]

• **Pro:** don't have to decrypt if MAC fails
  – resist DoS

• Example: IPsec (Internet Protocol Security)
  – recommends AES-CBC for encryption and HMAC-SHA1 for MAC, among others
  – or AES-GCM
MAC then encrypt

1. A: $t = \text{MAC}(m; k_M)$
   $c = \text{Enc}(m, t; k_E)$
2. A $\rightarrow$ B: $c$
3. B: $m', t' = \text{Dec}(c; k_E)$
   if $t' = \text{MAC}(m'; k_M)$
   then output $m'$
   else abort
MAC then encrypt

• **Pro:** provably next most secure
  – and just as secure as Encrypt-then-MAC for strong enough MAC schemes
  – HMAC and CBC-MAC are strong enough

• Example: SSL (Secure Sockets Layer)
  – Many options for encryption, e.g. AES-128-CBC
  – For MAC, standard is HMAC with many options for hash, e.g. SHA-256
**MACs**

- We can now protect integrity of messages against Dolev-Yao attacker
  - MAC algorithms use efficient symmetric-key cryptography
  - but what about quadratic key-sharing problem?

- Asymmetric cryptography for integrity...
DIGITAL SIGNATURES
Recall: Key pairs

- Instead of sharing a key between pairs of principals...
- ...every principal has a pair of keys
  - public key: published for the world to see
  - private key: kept secret and never shared
# Key pairs

<table>
<thead>
<tr>
<th>Encryption</th>
<th>Digital signatures</th>
</tr>
</thead>
<tbody>
<tr>
<td>Public key</td>
<td>Encryption key</td>
</tr>
<tr>
<td>Private key</td>
<td>Decryption key</td>
</tr>
</tbody>
</table>
Digital signature scheme

- **Sign(m; k):** sign message m with key k, producing signature s as output
- **Ver(m; s; K):** verify signature s on message m with key K
- **Gen(len):** generate a key pair (K,k) of length len
Protocol to exchange signed message

1. A: $s = \text{Sign}(m; k_A)$
2. A $\rightarrow$ B: $m, s$
3. B: accept if $\text{Ver}(m; s; K_A)$

- Message is sent in plaintext: no protection of confidentiality
- Goal is to detect modification not prevent
- Principals run different algorithms

...what if message is too long for asymmetric algorithms?
Signatures with hashing

1. A: $s = \text{Sign}(H(m); k_A)$
2. A -> B: m, s
3. B: accept if $\text{Ver}(H(m); s; K_A)$

So common a practice that I won't bother to write the hashing from now on
Security of digital signatures

• Must be hard to forge signature for a message without knowledge of key
  – message of attackers choice? vs.
  – message that attacker cannot control
    ...like handwritten signatures

• Even if in possession of multiple (message, signature) pairs for that key
  ...unlike handwritten signatures
Examples of digital signatures

• **DSA**: Digital Signature Algorithm [NIST 1991]
  – Used for decades without any serious attacks
  – Closely related to Elgamal encryption

• **RSA** [Rivest, Shamir, Adleman 1977]
  – Core ideas are the same as RSA encryption
  – Common mistake: RSA sign = encrypt with your private key
  – **Truth** (in real world, outside of textbooks):
    • there's a core RSA function R that works with either K or k
    • RSA encrypt = do some prep work on m then call R with K
    • RSA sign = do **different** prep work on m then call R with k
Upcoming events

• [today] A2 due
• [Mon] A3 out

Integrity without knowledge is weak and useless, and knowledge without integrity is dangerous and dreadful. – Samuel Johnson