MACs and Digital Signatures

Prof. Clarkson
Spring 2017
Review

• 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

• Today: **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
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 integrity solely from encryption
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
- “Secure MAC”? Must be hard to forge tag for a message without knowledge of key
Protocol to exchange MAC'd message

0. \( k = \text{Gen}(\text{len}) \)
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
Real-world 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
  - As of 2/23/17, “practical attack” on PDFs: https://shattered.io/
  - Industry has been deprecating SHA-1 over the couple years
    - E.g. MS IE 11 supposed to start rejecting SHA-1 last week
Real world hash functions

- **SHA-2**: NSA (2001)
  - Family of algorithms with output sizes \{224, 256, 385, 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**
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 pair terminology

<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

0. A: \((K_A, k_A) = \text{Gen}(\text{len})\)
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 $\rightarrow$ 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
  ...like handwritten signatures

• Even if in possession of multiple (message, signature) pairs for that key
  ...unlike handwritten signatures
REAL-WORLD DIGITAL SIGNATURES
RSA

• Core ideas are the same as RSA encryption
• Common mistake: “RSA sign = encrypt with 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 $k$
  – Prep work: recall “textbook RSA is insecure”
    • (For encryption: OAEP)
    • For signatures: PSS (probabilistic signature scheme)
**DSA**

**DSA**: Digital Signature Algorithm [Kravitz 1991]

- Standardized by NIST and made available royalty-free in 1991/1993
- Used for decades without any serious attacks
- Closely related to Elgamal encryption
EXTENSIONS
Blind signatures

[Chaum 1983]

• Purpose: signer doesn’t know what they are signing
• Two additional algorithms: Blind and Unblind
• Unblind(Sign(Blind(m); k)) = Sign(m; k)
• Uses: e-cash, e-voting
Undeniable signatures

[Chaum and van Antwerpen 1989]

• Purpose: signer doesn’t want the whole world to be able to verify

• Eliminate Ver algorithm; require signer to be available to verify signatures

• Two additional protocols between verifier and purported signer:
  – Confirm: convinces verifier that signature is valid
  – Disavow: convinces verifier that signature is invalid (i.e. signer did not sign that message)

• Either way, verifier can’t convince anyone else of that fact
Group signatures

[Chaum and van Heyst 1991]

• **Purpose**: one member of group signs anonymously on behalf of group
• **Introduces a group manager** who controls membership
• **Two new protocols**: Join and Revoke, to manage membership
• **One new algorithm**: Open, which manager can run to reveal who signed a message
David Chaum

- Invented e-cash
- Invented anonymous communication
- Many inventions in e-voting
- Founded IACR (1982)

b. 1955
Upcoming events

• [next Wed] A2 due

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