CS 5430

Secure Channel

Prof. Clarkson
Spring 2017

Review: Encryption, MACs

- We can protect confidentiality or integrity of a message against Dolev-Yao attacker

- Today:
  - What if we want to protect confidentiality and integrity?
  - What if we want to have a conversation not just a single message...?

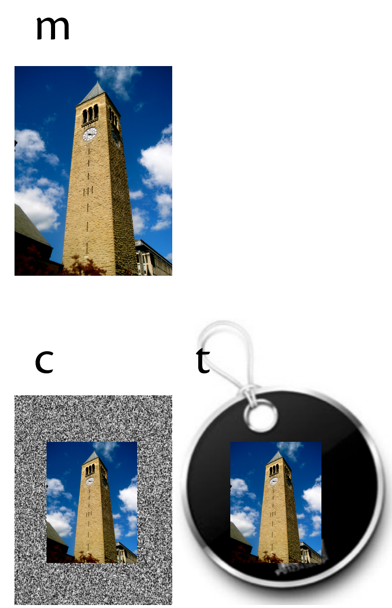
CONFIDENTIALITY & INTEGRITY

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

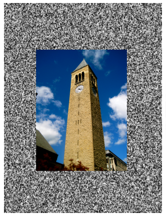
- Never use same key for both encryption and MAC schemes
- **Principle:** every key in system should have unique purpose

Encrypt then MAC

```
1. A: c = Enc(m; k_E)
      t = MAC(c; k_M)
2. A -> B: c, t
3. B: t' = MAC(c; k_M)
      if t = t'
        then output Dec(c; k_E)
        else abort
```

m

c



t

Encrypt then MAC

- **Pro:** provably most secure of three options [Bellare & Namprepre 2001]
- **Pro:** don't have to decrypt if MAC fails
  - resist DoS

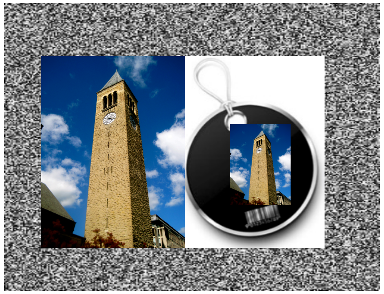
- Example: IPsec (Internet Protocol Security)
  - recommends AES-CBC for encryption and HMAC-SHA1 for MAC, among others
  - or AES-GCM

MAC then encrypt

```
1. A: t = MAC(m; k_M)
      c = Enc(m,t; k_E)
2. A -> B: c
3. B: m',t' = Dec(c; k_E)
      if t' = MAC(m'; k_M)
         then output m'
         else abort
```

m

c



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

Authenticated encryption

- Three combinations:
  - Enc and MAC
  - Enc then MAC
  - MAC then Enc
- Let's unify all with a pair of algorithms:
  - AuthEnc(m; ke; km): produce an authenticated ciphertext x of message m under encryption key ke and MAC key km
  - AuthDec(x; ke; km): recover the plaintext message m from authenticated ciphertext x, and verify that the MAC is valid, using ke and km
    - Abort if MAC is invalid

CONVERSATIONS

Protection of conversation

- **Threat:** attacker who controls the network
  - Dolev-Yao model: attacker can read, modify, delete messages
- **Harm:** conversation can be learned (violating confidentiality) or changed (violating integrity) by attacker
- **Vulnerability:** communication channel between sender and receiver can be controlled by other principals
- **Countermeasure:** all the crypto we've seen so far...

Secure channel

Channel:

- Bidirectional communication between two principals
- But their roles are not identical
  - Client and server, initiator and responder, etc.
  - We'll call them Alice and Bob
  - Same two principals might well have two parallel conversations in which they play different roles
- Communication might be...
  - spatial: over network
  - temporal: over storage
    - "Conversation with yourself"

Secure channel

Secure:

- The channel does not reveal anything about messages except for their timing and size (Confidentiality)
- If Alice sends a sequence of messages m1, m2, … then Bob receives a subsequence of that, and furthermore Bob knows which subsequence (Integrity)
  - And the same for Bob sending to Alice

Secure channel

Implications of security goals...

- No guarantee that any messages are ever received (subsequence could be empty) (no Availability goal)
- No attempt at anonymity
- No attempt to defend against traffic analysis
- Received messages:
  - are in order (or at least orderable)
  - are not modified
  - are attributable to the other principal

Secure channel

Pieces of the puzzle:

- Use authenticated encryption to protect confidentiality and integrity
  - Block cipher + mode
  - MAC
- Use message numbers to further protect integrity
- Use a key establishment protocol and key derivation function to create shared session keys

Secure channel

Pieces of the puzzle:

- Use authenticated encryption to protect confidentiality and integrity
  - Block cipher + mode
  - MAC
- Use message numbers to further protect integrity
- Use a key establishment protocol and key derivation function to create shared session keys

Message numbers

- Aka sequence numbers
- Every message that Alice sends is numbered
  - 1, 2, 3, ...
  - numbers increase monotonically
  - never reuse a number
- Bob keeps state to remember last message number he received
- Bob accepts only increasing message numbers
- And ditto all the above, for Bob sending to Alice
  - so each principal keeps two independent counters: messages sent, messages received

Message numbers

What if Bob detects a gap? e.g. 1, 2, 5

- Maybe Mallory deleted messages 3 and 4 from network
- Maybe Mallory detectably changed 3 and 4, causing Bob to discard them
- In either case, channel is under active attack
  - Absent availability goals, time to PANIC: abort protocol, produce appropriate information for later auditing, shut down channel

What if network non-maliciously dropped messages or will deliver them later?

- Let's assume underlying transport protocol guarantees that won't happen (e.g. TCP)

Message numbers

- Message number usually implemented as a fixed-size unsigned integer, e.g., 32 or 48 or 64 bits
- What if that `int` overflows and wraps back around to 0?
  - Message number **must** be unique within conversation to prevent Mallory from replaying old conversation
  - So conversation **must** stop at that point
  - Can start a new conversation with a new session key

Secure channel

Pieces of the puzzle:

- Use authenticated encryption to protect confidentiality and integrity
  - Block cipher + mode
  - MAC
- Use message numbers to further protect integrity
- Use a key establishment protocol and key derivation function to create shared session keys

Session keys

- For now, let's assume Alice and Bob already have a single shared session key k
  - Recall: *session key* is used for limited time then discarded
  - Here, the session duration is a single conversation
- But a single key isn't good enough...
  - Need a key for the block cipher
  - Need a key for the MAC
- And recall:
  - **Principle:** every key in system should have unique purpose
  - Implies: should not use same key for both Enc and MAC algorithms
  - Also implies: should not use same keys for
    - Alice **–>** Bob, vs.
    - Bob **–>** Alice

Key derivation

- Have one key: k
- Need four keys:
  1. kea: Encrypt Alice to Bob
  2. keb: Encrypt Bob to Alice
  3. kma: MAC Alice to Bob
  4. kmb: MAC Bob to Alice
- How to get four out of one: use a cryptographic hash function H to derive keys...
  1. kea = H(k, "Enc Alice to Bob")
  2. keb = H(k, "Enc Bob to Alice")
  3. kma = H(k, "MAC Alice to Bob")
  4. kmb = H(k, "MAC Bob to Alice")

Key derivation

- Why hash?
  - Destroys any structure in input
  - Produces a fixed-size output that can be truncated, as necessary, to produce key for underlying algorithm
  - Unlikely to ever cause any of four keys to collide
  - Even if one of four keys ever leaks, hard to invert hash to recover k and learn the other keys
- Small problem: maybe the output of H isn't compatible with the output of Gen
  - For most block ciphers and MACs, not a problem
    - they happily take any uniformly random sequence of bits of the right length as keys
  - For DES, it is a problem
    - has weak keys that Gen should reject
  - For many asymmetric algorithms, it would be a problem
    - keys have to satisfy certain algebraic properties

Secure channel

Pieces of the puzzle:

- Use authenticated encryption to protect confidentiality and integrity
  - Block cipher + mode
  - MAC
- Use message numbers to further protect integrity
- Use a key establishment protocol and key derivation function to create shared session keys

To send a message from A to B

```
1. A:
      increment sent_ctr;
      if sent_ctr overflows then abort;
      x = AuthEnc(sent_ctr, m; kea; kma)
2. A -> B: x
3. B:
      i,m = AuthDec(x; kea; kma);
      increment rcvd_ctr;
      if i != rcvd_ctr then abort;
      output m
```

To send a message from B to A

```
1. B:
      increment sent_ctr;
      if sent_ctr overflows then abort;
      x = AuthEnc(sent_ctr, m; keb; kmb)
2. B -> A: x
3. A:
      i,m = AuthDec(x; keb; kmb);
      increment rcvd_ctr;
      if i != rcvd_ctr then abort;
      output m
```

Secure channel

Pieces of the puzzle:

- Use authenticated encryption to protect confidentiality and integrity
  - Block cipher + mode
  - MAC
- Use message numbers to further protect integrity
- Use a key establishment protocol and key derivation function to create shared session keys

Session key generation

Back to this assumption:

For now, let's assume Alice and Bob already have a single shared session key k

We need a means for Alice and Bob to generate that key...

To be continued!

Upcoming events

- [Wed] A2 due, A3 out

Most conversations are simply monologues delivered in the presence of a witness. – Margaret Millar